



Car Park Control System

Software Engineering

Group 47

2012/4/28

Group Members:

<i>Jp092985</i>	<i>PING Yao</i>	<i>Jp092986</i>	<i>LIU Siyuan</i>
<i>Jp092987</i>	<i>LIU Zhijiao</i>	<i>Jp092989</i>	<i>LIU Yang</i>
<i>Jp092990</i>	<i>ZHANG Tianshu</i>	<i>Jp092991</i>	<i>GE Tong</i>
<i>Jp093165</i>	<i>ZHAO Chenyu</i>	<i>Jp093167</i>	<i>LIU Jiawei</i>

Content

INTRODUCTION	2
1. Brief introduction.....	2
2. Summary of Responsibilities and Achievements	2
REQUIREMENTS.....	6
1 Fact –finding Techniques	6
2 Requirements Capture.....	7
3 Defining the use case	8
ANALYSIS	12
1. Identify entity, boundary and control classes	12
2. Identify class relationships.....	12
3. Add constrains and attributes to conceptual diagram.....	12
4. Sequence diagram for each use case	13
5. Operation contracts	15
DESIGN.....	17
1. Design Principle.....	17
2. Architectural design	17
3. Communication diagrams for key operations	17
4. Design a class.....	19
5. Class diagrams	21
6. Re-usability of system components.....	22
IMPLEMENT	22
1. Architectural implementation.....	22
2. Integrate System.....	23
3. Implement a subsystem	24
4. Implement a class.....	25
5. Perform Unit Tests	25
TESTING.....	25
1. Plan test.....	25
2. design test.....	26
3. Implement Test.....	29
4. Perform Integration Test.....	29
5. Perform system test.....	29
6. Evaluate test	30
CONCLUSION.....	30
REFERENCE.....	30
APPENDIEX	31

INTRODUCTION

1. Brief introduction

The aim of the software engineering program is to design and build an easy-operating and demands-meeting car park control system. To finish the task efficiently, we divide the whole group into three subgroups in charge of different tasks during each period. In order to design the system that fulfills all needs, we capture requirements by interviews and questionnaire. Use cases are decided and models are described in UML.

Then we analyze all captured requirements and compose conceptual class diagram and sequence diagram which better indicate the operation of the system.

After analysis, we begin to design the system in detail in order to implement it in programming language. JAVA related details are discussed in this step.

We start to implement the system. Builds are designed and functions are realized in this step by composing the program.

We also test the system according to the plan. The whole system is tested after each component has passed the test. The whole system is also passed the test and meets all necessary requirements.

2. Summary of Responsibilities and Achievements

ID	Mission	Start time	Finish time	Duration	2012 March					2012 April			
					2/26	3/4	3/11	3/18	3/25	4/1	4/8	4/15	
1	Requirement	2012/2/22	2012/3/7	2w 1d	[Gantt bar]								
2	Analysis	2012/2/29	2012/3/14	2w 1d	[Gantt bar]								
3	Design	2012/3/7	2012/3/21	2w 1d	[Gantt bar]								
4	Implementation	2012/3/14	2012/4/4	3w 1d	[Gantt bar]								
5	Test	2012/3/28	2012/4/18	3w 1d	[Gantt bar]								
6	Deployment	2012/3/8	2012/4/25	7w	[Gantt bar]								

< Yao PING, jp092985 >

Summary of responsibilities		
Stage	Summary	Page
Requirements	Participate in designing questionnaire and analysis the result. Capture non-functional requirements. Discuss with team members to define the actors and use case model and relationship between them Write use case description of enter and payment.	2, 7-9
Analysis	Identify entity, boundary, and control classes. Identify attributes for each class.	12
Design	Do architectural design. Discuss with team about design a class.	17
Implementation	Implement sub systems. Participate in coding.	22-23
Testing	Testing perform integration.	29
Group Member Contribution		
Ping Yao is responsible for report writing and edits individual report into together. He could finish his work on time and never miss any meeting. Although sometimes he would make some mistakes in his reports, he had a great attitude to face this course work. Moreover, he can cooperate with other group members and always provides us with novel advice.		
Self-Appraisal Of Work		
Though this course work, I have a better understand about software engineering and enjoy cooperating with other excellent group members. At first I made many mistakes in my part because I didn't know this lesson well. But with the help of my group member, I corrected all the mistakes I made. What's more, I also learn how to manage my time appropriate and get well known about the importance of team work.		

< Siyuan LIU, jp0929986 >

Summary of responsibilities		
Stage	Summary	Page
Requirements	Complete the background of car park Discuss the interview Discuss the actors and use case Draw use case diagram	6, 8, 9
Analysis	Discuss classes of entity, boundary and control Discuss relationship and attributes of classes Write operation contracts Draw diagram of class	13, 15
Design	Draw completely class diagram Write the re-usability of the system	21-22
Implementation	Draw diagram of subsystem Code	24
Testing	Complete system testing	30
Group Member Contribution		
She is a team member, who draws a lot of diagram. She joins the discussion and sometimes can come up with some ideas. And she can cooperate with others. While she should prove her ability of writing so that write report better.		
Self-Appraisal Of Work		
I learn a lot from this coursework, like analysis and cooperation. In team meeting, I have some problem and my team mates tell me. So, sometimes I can provide some ideas. And I know it difficult to write something for me, so I draw a lot of diagram. And the cooperation is an important experience for me. And I should improve my writing ability.		

< Zhijiao LIU, jp092987 >

Summary of responsibilities		
Stage	Summary	Page
Requirements	Go to BUPT main campus and TB1 to do the interviews for fact-finding aspects. Finish the analysis of the functional requirements and priorities the uses cases. Discuss the parameters of each aspect and prepare for the use case and prototype.	7, 12
Analysis	Define classes of entity, boundary and control. Draw the conceptual class diagram with constrains. Identify attributes for each entity class.	12-13
Design	Do architecture design Design the classes of the system Identify attributes and operations	19-21
Implementation	Implement classes. Perform unit test. Do main coding part, especially the UI.	25
Testing	Writing Implement Test, including automatic test and manual test.	29
Group Member Contribution		
She is a responsible group leader, who devotes much effort on providing nice cooperating environment and building good inter-group relationships. She contributes a lot to implementation of the system, especially for the user interface. But her arrangements of individual work need some improvement for some members are overworked.		
Self-Appraisal Of Work		
I am a responsible group leader. I devote much effort on providing nice cooperating environment, because a good cooperation environment and relationships between members can help us finish the work efficiently. Also, I carefully plan future works, organize discussions, and motive group members in face of difficulties or delays. However, as a leader, I focus too much on the program, but not good at arranging equivalent individual work. That is what I need to improve in the future.		

< Yang LIU, jp092989 >

Summary of responsibilities		
Stage	Summary	Page

Requirements	Do background reading and search the existing examples of parking system for fact-finding aspects. Do the analysis from 4 aspects, including Entrance Management, Payment station, Exit Management, Main control, to capture the functional requirements. Discuss the parameters of the use case and prototype. Integrate and improve the requirement part.	6, 7, 12
Analysis	Define the actor and use case model and relationship between these according functional requirements. Discuss classes of entity boundary and control. Draw the diagram of payment station and the conceptual class diagram. Check and correct the operation contracts.	13, 15
Design	Work on the design principles. Define the attribute and relationship of the classes. Identify the operation and method for each class.	17, 20
Implementation	Implement classes. Perform unit test. Do main coding part, especially the control.	25
Testing	Writing Implement Test, including automatic test and manual test.	29
Group Member Contribution		
She is a responsible group member, who can finish the arranged task in a good quality on time. She contributes a lot to implementation of the system based on the discussion result after the analysis and design stage, especially for the architecture of the system and the control classes. But she pays less attention on the writing the report.		
Self-Appraisal Of Work		
I am a responsible group member. I can cooperate with the other group member and have some discussion on the project. At the earlier stage, I forgot knowledge learned in lesson, but after review of text book, I solve problems successfully. At the same time, I learn a lot on how to communicate and cooperate with the subgroup member in order to work out the final work efficiently.		

< Tianshu ZHANG, jp0929990 >

Summary of responsibilities		
Stage	Summary	Page
Requirements	Interview Dr. Marie-Luce Bourguet and analyse the result. Participate in group discussion on the function of the system. Discuss the actor and use case around the group and define the actor and use case model. Discuss the prioritize use case around group. Integrate and improve the requirement part.	6, 7, 10, 12
Analysis	Discuss classes of entity, boundary and control, relationship of classes and attributes of entity class. Write operation contracts Draw diagram of class	14,16
Design	Discus according analysis. Identifying Associations and Aggregations	17,21
Implementation	Writing Build 1.0 of Integrate system part. Code	23, 24
Testing	Writing system testing level part of Testing plan. Participate in perform unit tests work.	25, 26
Group Member Contribution		
She is a teammate, who is good at conclusion. She records our discussion and lists them. She can finish her job on time. While she should say more in the team discussion.		
Self-Appraisal Of Work		
I know I am good at English, so I write a lot. I hear others in team meeting and record something so that we can get a clear mind. I draw some diagram according to the result of discussion. I know		

I should have a change in analysis, so I work hard to follow others and get a lot.

< Tong GE, jp092991>

Summary of responsibilities		
Stage	Summary	Page
Requirements	Do background reading and be in charge of the parking policy description of the existing parking system in QM. Synthesize “Applying requirements finding techniques”. Define the actors and use case model and relationship between them. Draw the sequence diagrams of register and maintain. Prioritize the use cases.	6, 7, 9
Analysis	Discuss classes of entity, boundary and control Consider relationship of classes Write operation contracts Draw diagram of class	13, 16
Design	Impact of issue integrate and improve the analysis Identify operations Identify methods	18, 20
Implementation	Write build 2.0 of integrate system part	24
Testing	Write component testing level part of testing plan Write perform unit tests Write evaluate test part Integrate the whole report	25-26, 28, 2, 30

Group Member Contribution

She is responsible for the completion of the work. She participates positively in every team meeting and always provides useful ideas. Although she is very busy doing other works, she can always complete the allocated tasks in time.

Self-Appraisal Of Work

During the software development process, I manage to do everything allocated to me in time and successfully. But because of the unfamiliar of definition, I have made a big mistake that results in the rework of the whole team. By doing such a work, I am more familiar with the lecture slides and I also understand a person should learn to be responsible for what you are in charge of. In addition, I practice a lot about how to arrange time and to cooperate with teammates.

< Chenyu ZHAO, jp093165>

Summary of responsibilities		
Stage	Summary	Page
Requirements	Design the observation and go to car park to do it for fact-finding technique. Capture and quantify the non-functional requirements Discuss and define the actor and use case model. Draw the use case diagram.	7, 8, 9, 11
Analysis	Define classes of entity, boundary and control. Draw the conceptual class diagram. Draw sequence diagram. Write the operation contracts.	13, 15
Design	Do architecture design Design the classes of the system Identify attributes and operations Draw communication diagram	17, 19
Implementation	Draw architectural implementation diagram Code	22
Testing	Make test design and test matrix	27-28

Group Member Contribution

He is a comprehensive and hard-working group mate, who is good at problem analysis and designing. He contributes a lot to design the system. Additionally, he finds the unreasonable part of the work and corrects it. Also, he draws a lot of diagrams and writes some report. But he needs some improvement in coding.

Self-Appraisal Of Work

I am a comprehensive group mate. I devote much effort on problem analysis and software design, discussing and giving good ideas to the group. I also can find the unreasonable part of the work and corrects it whether I work it or not. I really appreciated this experience with my group mates. I have learned much about coding and some wonderful ideas from them.

< Jiawei LIU, JP093167 >

Summary of responsibilities		
Stage	Summary	Page
Requirements	Design questionnaire, issue them and analysis the results of the questionnaire. Cooperate with other teammates to discuss functional requirements. Do the non-functional requirements. Define the actors and use case model and relationship between them. Design and draw use case diagrams. Write the use case description of equipment repair and update data.	7, 8-9,11
Analysis	Identify entity, boundary and control class. Draw class diagram. Design and draw use sequence diagrams of payment and maintain. Write operator contracts of payment and maintain.	14, 16
Design	Design and draw most communication diagrams Identify operations and methods.	17-18
Implementation	Implement classes. Perform unit test. Do coding.	25
Testing	Design test, identify and describe Test cases for each build Identify and structure Test Procedures specifying how to perform the test cases.	28-29
Group Member Contribution		
Liu Jiawei is responsible for the report team and do what teammates expect him to do. He participates each team meeting and provides ideas to improve coursework. He can cooperate with different teammates, organizing and managing time to complete tasks. Consciously subject to arrangement and praised for a job well done.		
Self-Appraisal Of Work		
During the whole coursework process, I am a member in a good team which everyone has a cordial working relationship. I manage other two teammates to finish our parts of work and complete almost tasks with them. I also faced some troubles irretrievable so that we have to redo the whole project. But most errors is because of the negligence and unfamiliar to the slides. It makes me realize the importance of preparation work.		

REQUIREMENTS

1 Fact –finding Techniques

To deeper understand the requirements, we select three fact-finding techniques: background reading, interview and questionnaires.

1.1 Background Reading

We gather information about existing systems and policies to study from successful cases and know more about the client we serve.

- Company report:

Queen Mary University is visited by lots of people every day, and the current campus car parking system cannot meet the demand. Thus a new one is needed.

- Policy manuals:

In Queen Mary Campus, parking is offered to staff, visitors, contractors and those making deliveries. Every user must obtain and prominently display a valid QMUL parking permit.

- Job description:

A car park with 30 parking spaces, one entrance, one payment station and one exit is planned. The entrance and the exit are barrier controlled. It's staff-only on weekdays

and also available to public at weekends, public holidays, and summer time.

- Documentation of existing system:

This kind of system is already invented, thus our project can be realized.

1.2 Interviewing

To know users' needs in detail, we interviewed QM staff, BUPT drivers, and operators. We conclude that Car parks located near work places are preferred. Registering only with ID number is more secure and convenient. The most critical concern of the users is usability and reliability. It's acceptable to only accept coins and no change given. Detailed contents are in Appendix 4.

1.3 Questionnaire

In order to know practical requirements of users, we design a questionnaire issued on the internet to understand what they need and what they expect for the parking system.

According to the result of the questionnaire, we can make conclusions that:

- End users want to avoid personal info, so just ID to be registered is enough.
- Usability and reliability are the most important elements for park operator.
- Staff and public want to use it in simple way.

2 Requirements Capture

2.1 List Candidate Requirements

The system is to provide help of car parking management within the QMU campus.

- Operator updates tariffs and opening times to public and does maintenance.
- Staffs can park cars at any time by scanning campus cards after registration. Parking fees will be charged automatically from their salary monthly.
- Public can only use the car park at specific days. They are charged according to the tariff table and only coins of certain denominations are accepted.
- The system should be easy to use. The registering of staff should be simple.

2.2 Understand the system context

Purpose: the system is used for management of car park.

Scope: the system can control the entrance and exit barriers, identify the campus cards and tickets, and process public payment. The charging of staffs is beyond the scope.

2.3 Assumptions

- We assume the barrier can open for enough time for only one car to pass.
- We assume that staffs have unique campus cards that can be used to recognize their identities at the parking entrance.
- We assume that every staff has only one car can be identified by the ID number.
- We assume that the car must enter the car park if a ticket is taken or a campus card is scanned, and same to the exit.
- We assume there is device at payment station to check the denomination of coins.
- We assume that cars wouldn't remain in the park after processing the payment.

2.4 Functional Requirements

2.4.1 Entrance Management

- A registered staff can open the entrance barrier by scanning the campus card.
 - A public customer can open the entrance barrier by taking an issued ticket.
 - The system can identify and record who and when enters the car park.
 - The entrance screen should show the information including tariffs, availability to public, and numbers of available spaces to users.
 - The system should keep the entrance barrier closed when the park is full.
 - The system should judge if blank tickets are used up and inform the main control.
- Addition: The system may be able to recognize the number plate automatically.

2.4.2 Payment station management

- The payment screen should show the parking hours and amount of charge.
- The university staff's payment condition can be changed and related information can be recorded into their bills.
- The public users can complete their payment with tickets and their payment condition can be changed.
- The system should judge if the cash box is full, and informed the main control.
- There is device at the exit to ensure that unpaid tickets would not be taken in.

Addition: The system can print receipts for customer, give change, process the notes and credit/debit cards and store all transaction information.

2.4.3 Exit management

- A registered staff can open the exit barrier by scanning the campus card.
- A public customer can open the entrance barrier by inserting a paid ticket.

2.4.4 Main control

- The operator can do staff profile management (register).
- The operator can update the system information (tariffs, opening time, etc.)
- The operator can obtain the instant information from the entrance and payment station.
- The system can generate monthly bills to the registered staff.
- The system can control the printed information on each screen.

Addition: The car park can be enlarged and spaces number can be modified.

Many payment stations can co-exist. Usage report can be generated for QMUL.

2.5 Non-functional Requirements

1) Product requirements

a) Efficiency

i. Performance: The system can provide automatic service of parking and calculating charge. The response time of the whole system should be no more than 1s.

ii. Space: The software needs no more than 20MB of the disk space.

b) Reliability: The data should not lose if failure happens. The whole system's error rate is lower than 1%. The system will recover from a failure after restart.

c) Portability: The hardware environment, software environment and organization of system environment are all portable in different operating systems.

d) Usability: Operators can easily know how to use it by simple training. Users can use the system without training. Users can still do normal operations in the abnormal environment.

2) Organizational requirements

a) Delivery: The first part of the documentation should be finished by March 23rd. The whole system and the final report should be finished by April 27th.

b) Implementation: The system can work by both visualization operation and java language programming.

c) Standards: A Readme file should explain clearly how to install, compile and run the code. The software should be developed as a standalone Java app with Java GUI.

3) External requirements

a) Interoperability: Different networks, systems and applications can work together.

b) Ethical: Multilanguage supported and language courtesy

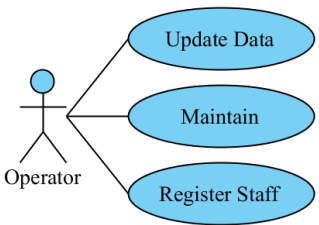
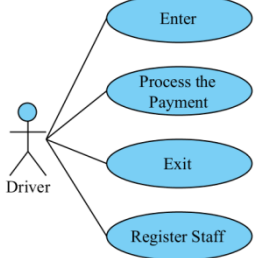

c) Legislative

i. Safety: The administrator cannot change records.

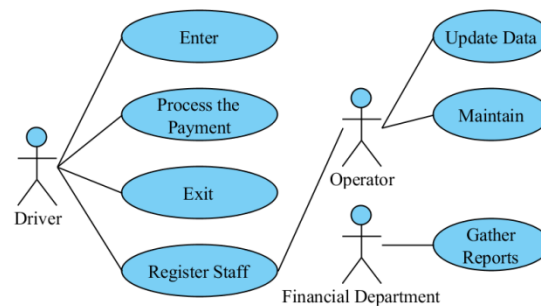
ii. Privacy: The system would never leak users' information.

3 Defining the use case

3.1 Finding actors and use case for them

		
<p>Operator: A worker who is hired to perform jobs in the garage.</p>	<p>Driver: Anyone who use the car park to park cars.</p>	<p>Financial Department: An external system that gathers parking bills and charge employees.</p>

3.2 Use cases diagram



3.3 Use case description

Use case: Enter

—Pre-conditions:

For public: At weekends, holidays, summer time, blank tickets are available.

The car park is not full.

—Description:

The public presses the button at the entrance device. The device issues a ticket. The public takes the ticket. The system enters one of the random public accounts and record the ticket number and entrance time in this account. The system sets “unpaid” condition to this account. The entrance barrier opens. The number of available parking spaces reduces one. The barrier closes after enough time for the car passing.

The staff scans the campus card at the entrance device. The system checks the ID to find whether it is registered. If it is, the system enters the corresponding staff account and records the entrance time in this account. The system sets “unpaid” condition to the account. The entrance barrier opens. The number of available parking spaces reduces one. The barrier closes after enough time for the car passing.

—Post-condition:

The entrance time is recorded in corresponding account.

The corresponding account is set unpaid.

The number of available parking spaces is reduced by one.

The ticket number is recorded in a random public account (for public entering).

The public get the ticket (for public entering).

The driver enters the car park.

—Exceptions:

If the campus card is not registered, the entrance barrier would not open.

Use case: Process the Payment

—Pre-conditions:

The campus card or ticket is unpaid.

Cash box is not full (for public).

—Description:

The public scans the ticket at payment station device. The system enters the corresponding account and calculates parking hours and amount of money to be paid and saves them into the account. The screen displays them. The public put appropriate amount of required kinds of coins into the cash box. The counter of cash box increases the corresponding amount to update data. System changes the state of

account to 'paid'.

The staff scans the campus card at payment station device. The system enters corresponding account and calculates parking hours and amount of money to be paid. The information is recorded under corresponding ID number in the system. The system sets the state of account 'paid'.

—Post-conditions:

The money is put into the cash box (for public).

The state of the account is set 'paid'.

The parking charge is recorded into the system under corresponding ID.

—Exception:

If money isn't of the required denomination or kinds, the box doesn't take it in.

Use case: Exit

—Pre-conditions:

The payment of campus card (for staff) or ticket (for public) has been processed.

—Description:

The public tries to insert the ticket at the exit. The system checks the ticket number and enters the corresponding account to check whether the ticket is paid. The system accepts the ticket if it's paid. The exit barrier opens. The number of available parking spaces adds one. The barrier closes after the car has passed.

The staff scans the campus card at the exit device. The system checks the campus card and enters the corresponding account to check whether it's paid. The exit barrier opens if the account state is paid. The number of available parking spaces adds one. The barrier closes after the car has passed.

—Post-condition:

The ticket is returned back to the system (for public exit).

The number of available parking spaces adds one.

The car leaves the car park.

The random public account is released for reusing.

—Exceptions:

If the ticket is unpaid, the system would not take the ticket back, and the barrier would not open.

If the staff account state is unpaid, the barrier would not open.

Use case: Register Staff

—Pre-conditions:

The staff hasn't been registered.

The staff provides his/her ID number to the operator.

—Description:

The staff provides the ID number to the operator. The operator inputs the ID number into system. The system searches the ID number in university database. If it can be found, the staff is registered into the parking system.

—Post-conditions:

The staff is registered to the parking system.

—Exception:

If the staff has been registered, the register behavior is rejected.

If the ID number cannot be searched in the university database, the staff cannot be registered in the parking system.

Use case: Update Data

—Pre-conditions:

The present condition of the car park has changed.

—Description:

The operator is asked to update data according to the condition change (opening time to public, tariffs). The operator modifies the data in the system. The new condition would be used immediately and shows to drivers.

—Post-conditions:

The changes of the current car park state are updated by the operators.

The public user can know the current state of the car park in time.

—Exception: None

Use case: Maintain

—Pre-conditions:

System needs maintenance and the warning in the system has been issued.

The operator is informed by UI.

—Description:

Absence of blank tickets, saturation of cash box, or other abnormal cases causes the system stops working. The operator is informed by the system and solves the problem. The system continues to work after the problem is solved (blank tickets are filled up or cash box is cleared, etc.). The system records this maintenance.

—Post-conditions:

Blank tickets are added to system.

Cash box is empty and can be stored money again.

System error has been handled and it operates normally again.

—Exception: None

Use case: Gather Reports

—Pre-conditions:

The system has generated reports.

The time comes to the end of a month.

—Description:

The financial department asks for monthly report from the system. The system sends the bill of each registered staff to financial department. The financial department checks the bill and deducts parking charges from staffs' salary.

—Post-condition:

The account information of registered staffs is sent to the financial department.

Parking charge of this month is deducted from staff's salary.

—Exceptions: None

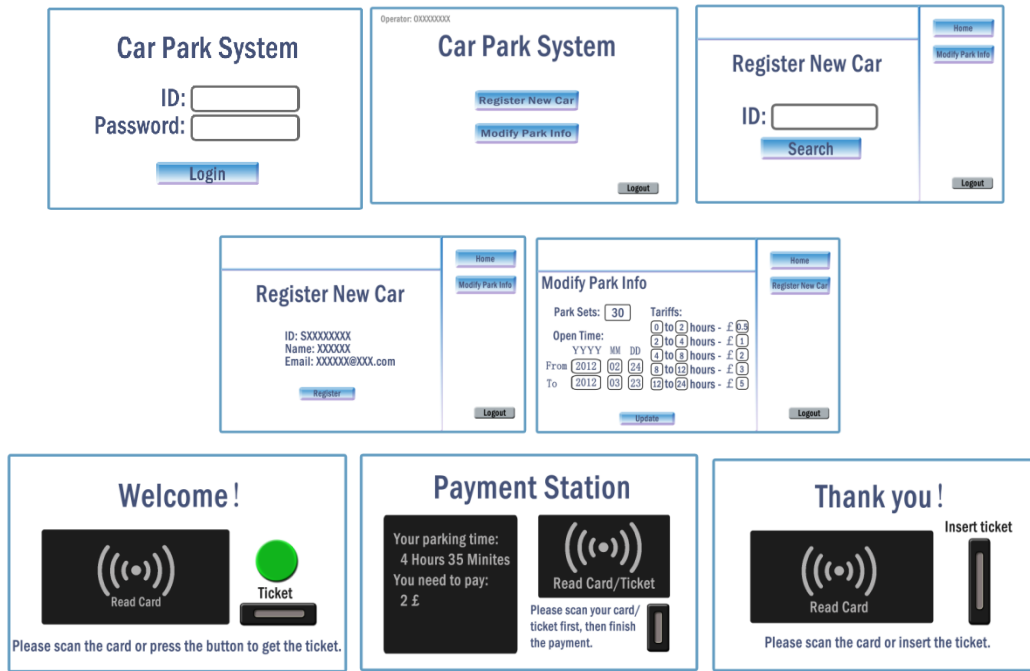
3.4 Prioritize the used case

First priority: Enter; Process the Payment; Exit

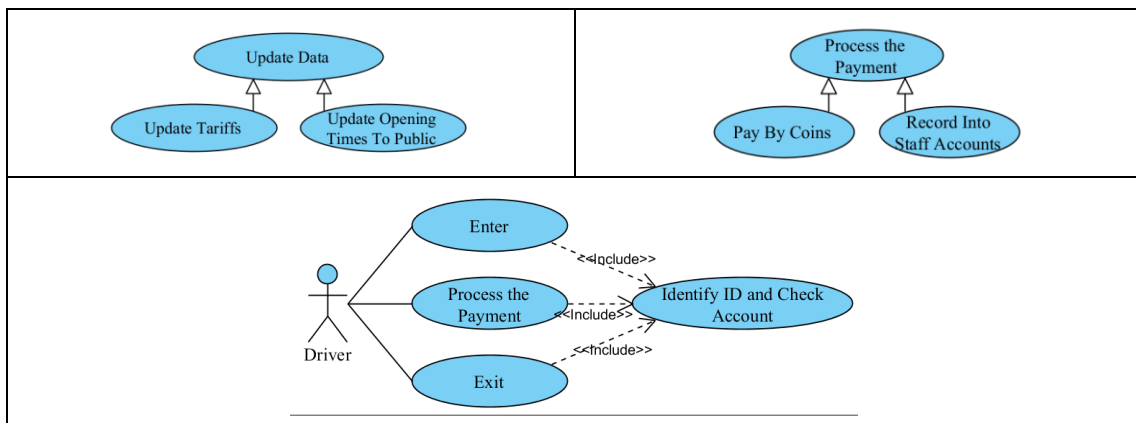
Second priority: Register Staff; Update data; Maintenance

Third priority: Gather Reports

3.5. Prototype user interface



3.6 Refine the Use Case Model



ANALYSIS

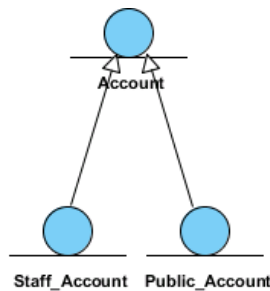
1. Identify entity, boundary and control classes

Boundary class: Operator_UI, Enter_UI, Payment_UI, Exit_UI, FinanDep_UI

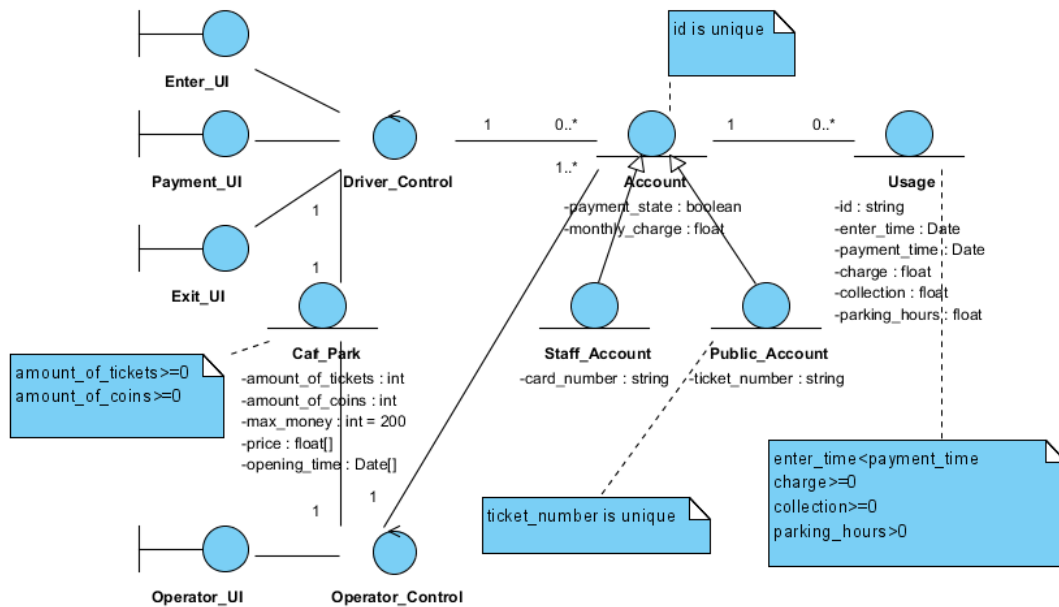
Control class: Driver_Control, Operator_Control, FinanDep_Control

Entity class: Account, Usage, Car_Park, Operation

2. Identify class relationships

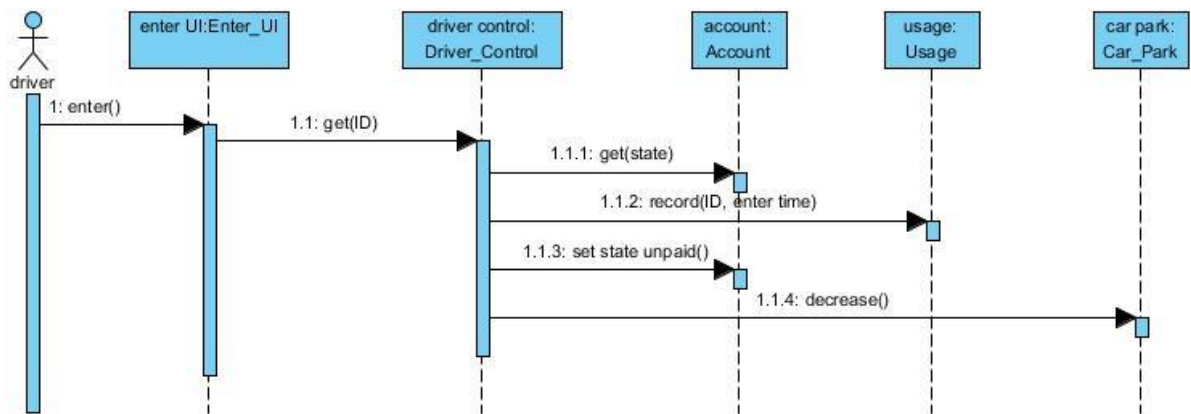


3. Add constrains and attributes to conceptual diagram

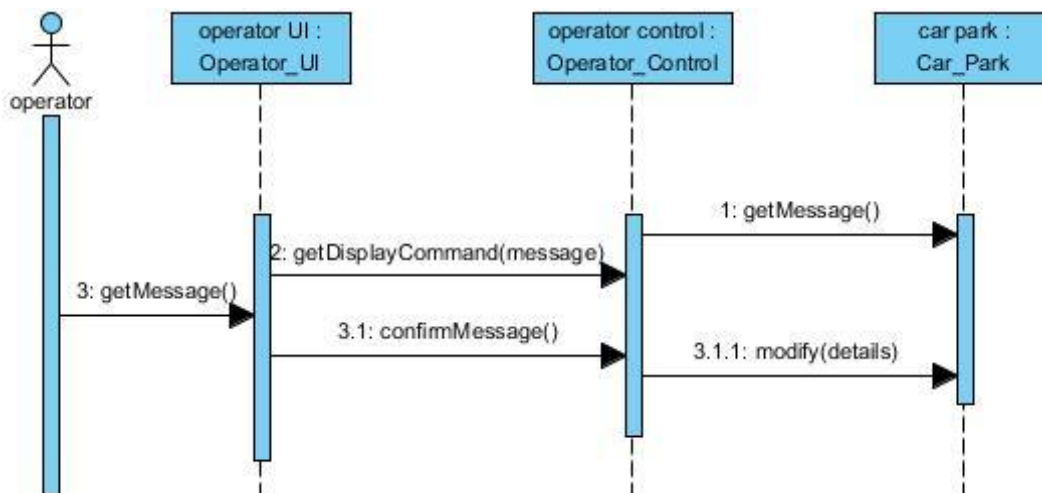


4. Sequence diagram for each use case

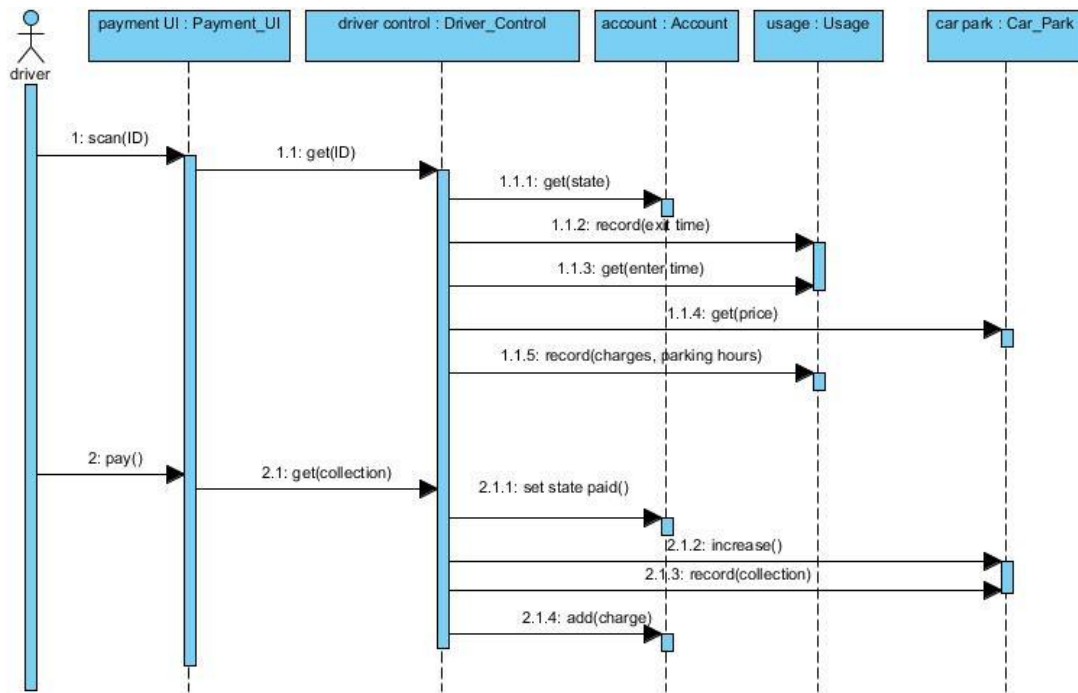
● Enter



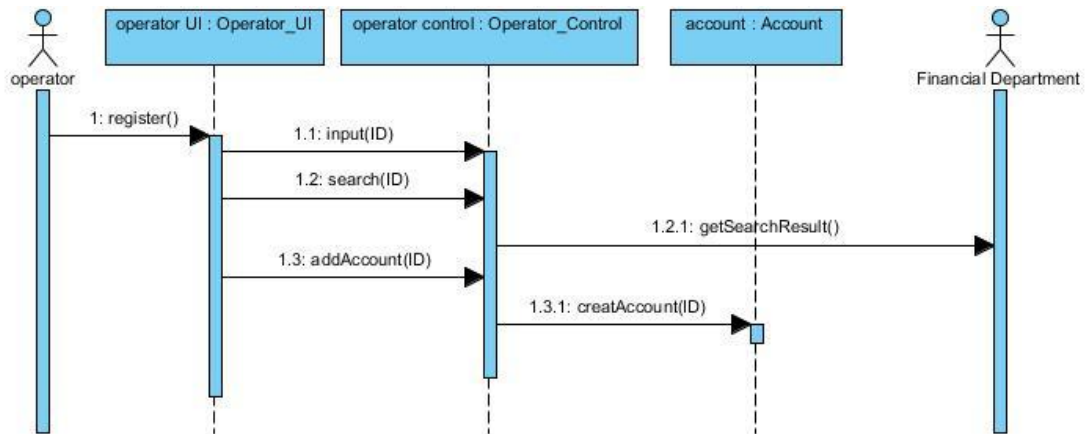
● Maintain



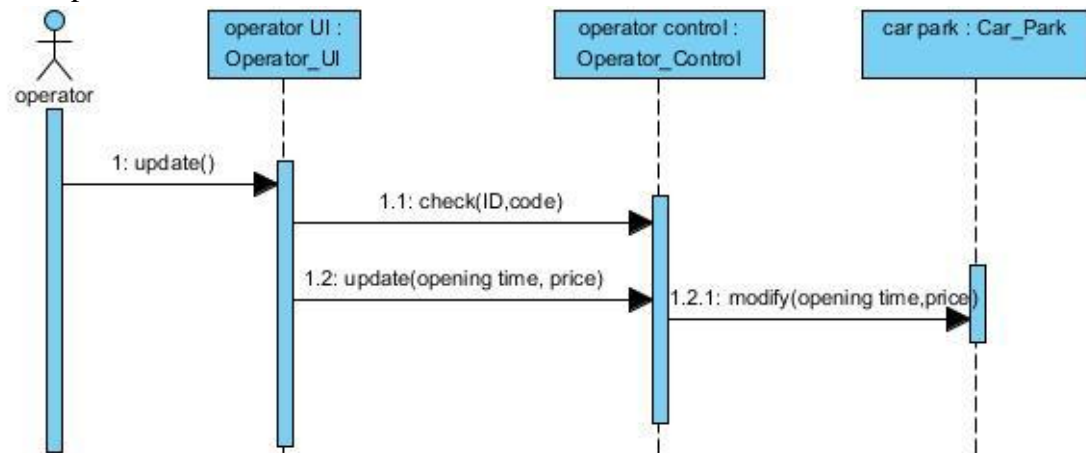
● Payment



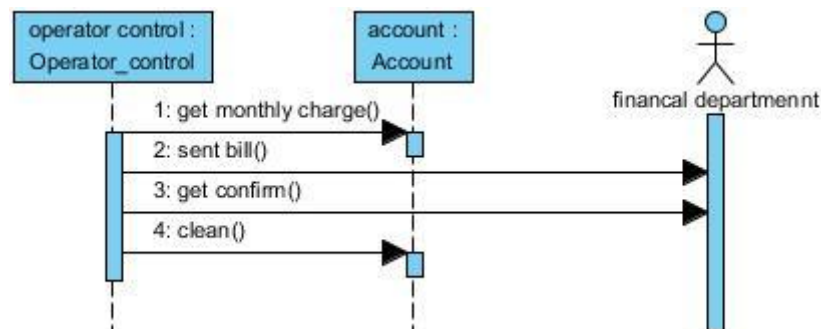
● Register



● Update



● Gather Reports



5. Operation contracts

1) Name: enterCheck(ID)

Responsibility: Record the details of the car enter

References: Enter use case

Exceptions: If it is not on holidays, the public cannot enter.

Output: The remaining place on the screen decrease one.

Open the barrier and last for a while.

Pre-condition: The amount of tickets is above zero.

The public press the button and take the ticket. (On holidays)

The staffs show the campus card.

The corresponding account exists in the system.

The account state is "paid"

The remaining place of the car park is above zero.

Post-condition: The account state is set to "unpaid"

The ID and enter time are recorded in the corresponding account.

The remaining place decrease one.

The amount of tickets decreases one.

2) Name: paymentCheck(ID)

Responsibility: Record the details of the payment.

References: Process the Payment use case

Exceptions: None

Output: Show the charge on the screen.

Pre-condition: The driver shows the campus card or the ticket.

The corresponding account exists in the system.

The account state is "unpaid"

The tariff is given.

The enter time is given.

Post-condition: Exit time, charges and parking hours are recorded in the corresponding account.

3) Name: collectionCheck(collection)

Responsibility: Record the details of the receiving charges.

References: Payment use case

Exceptions: If the drivers use the campus card, they do not need to pay. The system just records the charge and set the collection equals charge.

Output: The remaining place on the screen increase one.

The details of the usage is sent to database (txt)

Pre-condition: The collection is given.

The collection is not less than the charge.

The corresponding account exists in the system.

Post-condition: The account state is set to "paid"

The charge is recorded in the corresponding account.

The collection is recorded in car park

The remaining place increase one.

The amount of coins increases by the number of collection.

4) Name: exitCheck(ID)

Responsibility: Check whether the charge is paid

References: Exit use case

Exceptions: None

Output: Open the barrier and last for a while.

Pre-condition: The public return the ticket or the staffs show the campus card.
The account state is set to "paid"

Post-condition: None

5) Name: search(ID)

Responsibility: Check the staff campus card number

References: Register use case

Exceptions: None

Output: A request for checking the staff campus card number

Pre-condition: The ID is given
The corresponding account does not exist in the system

Post-condition: None

6) Name: addAccount(ID)

Responsibility: Create new account

References: Register use case

Exceptions: None

Output: None

Pre-condition: The corresponding account does not exist in the system
The ID, which is going to be registered, is given
The search result meets the requirement

Post-condition: The new account is created in the system
The association between account and usage is created.

7) Name: operatorCheck (ID)

Responsibility: Confirm the authority of the operator

References: Update use case

Exceptions: None

Output: None

Pre-condition: The ID and code is given

Post-condition: None

8) Name: update(opening time, price)

Responsibility: Change the tariff and opening time

References: Update use case

Exceptions: None

Output: The new tariff and opening time is showed on screen.

Pre-condition: The authority of the operator is confirmed
New tariff or opening time are given

Post-condition: New tariff or opening time replaces the old ones

9) Name: getDisplayCommand(message)

Responsibility: Report the abnormal condition of the car park

References: Maintain use case

Exceptions: None

Output: Report the abnormal condition of the car park to operator
Details of the maintain are sent to database (txt)

Pre-condition: The tickets number equals zero or the collection is not less than the max value

Post-condition: None

10) Name: confirmMessage()

Responsibility: confirm maintenance

References: Maintain use case

Exceptions: None

Output: None

Pre-condition: The report of the abnormal condition of the car park is confirmed
The authority of the operator is confirmed

Post-condition: The collection is set to 0 or the number of tickets is set to maximum

DESIGN

1. Design Principle

Since the high quality design and implementation is essential if we are to manage the complexity of modern real world software systems. We have to think of software systems at a higher level than just code.

a) Single Responsibility Principle(SRP):

Every object in a system should have a single responsibility, and all the object's services should be focused on carrying out that single responsibility. There are different degrees of linkage between classes. In this case, the account class has a link to the usage object, each account can have one or more usage, by the core linkage card number.

b) Open-Closed Principle(OCP):

This principle means that we can make a module behave in new and different ways as requirements change or to meet new needs. But we should be able to do this in a way that does not require changing the code of the module. Here, we use some abstraction to follow this principle. This is shown especially in the Account class.

c) Don't Repeat Yourself(DRY):

When we find we are repeating ourselves in code, we should see if we could replace the repeated code by writing and using more abstract code.

d) Dependency Inversion Principle(DIP):

In this case, the class public account and staff account are inherited the abstract class Account which conform to the Dependency Inversion Principle (DIP).

2. Architectural design

Nodes and their network configuration

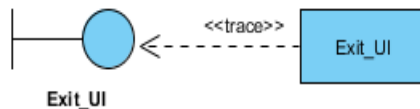
The car parking system is a simple stand-alone Java application; it isn't connected to other system or network.

Subsystem and their interfaces

There are driver subsystem and operator subsystem. They communicate through two interfaces which are the Driver_Control and the Operator_Control.

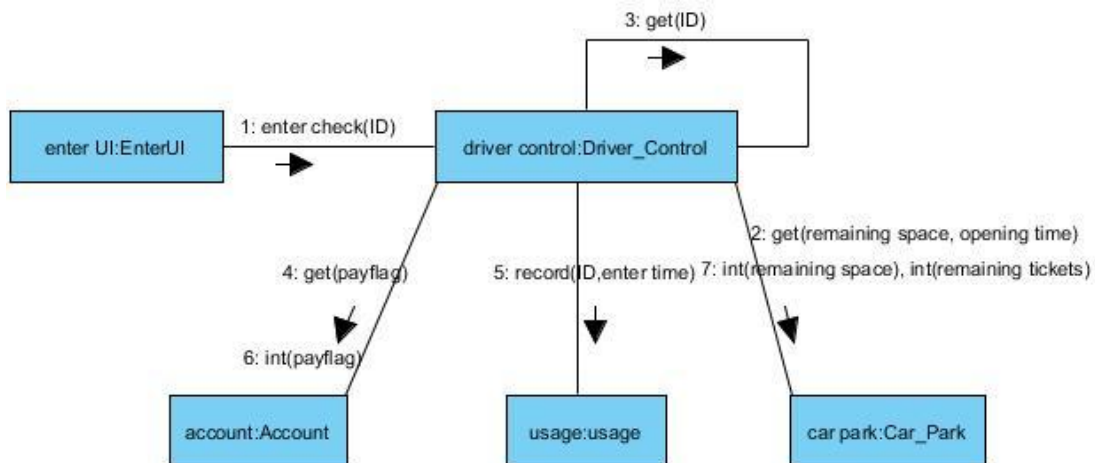
Outlining Design Classes

The design classes are mapped from classes obtained from Analysis. They are Driver_control, Operator_control, Usage, Car_park, Account, Payment_UI, Exit_UI, Operator_UI, Enter_UI. There is one example. Other mappings are in Appendix 4.

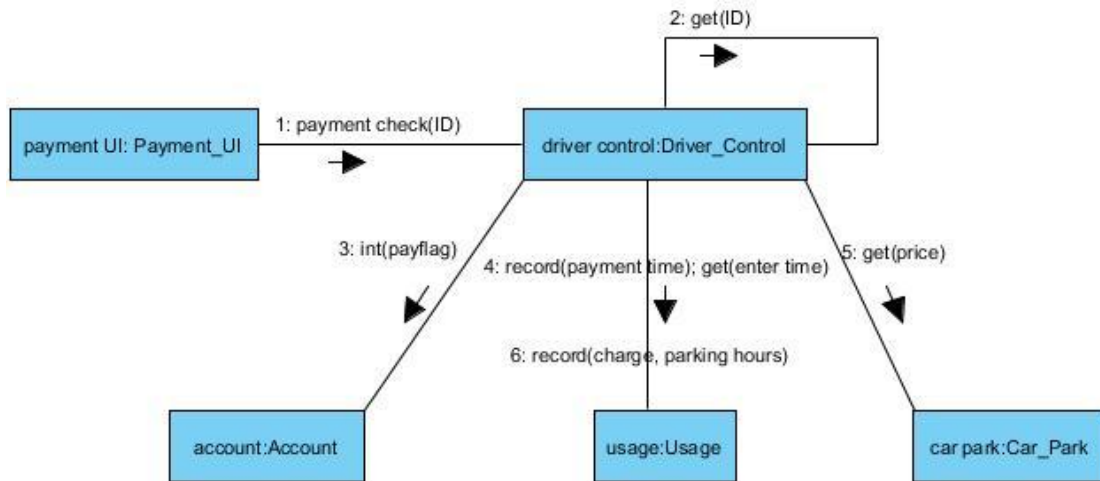


3. Communication diagrams for key operations

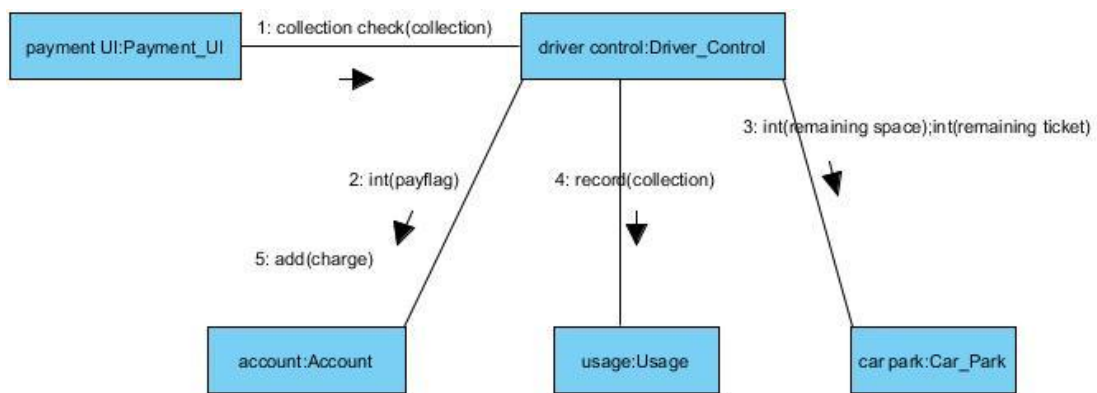
1. enterCheck



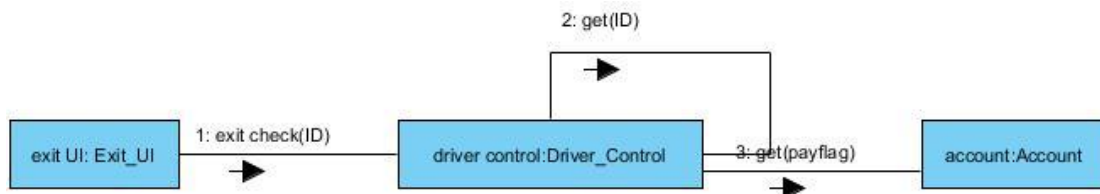
2. paymentCheck



3. collectionCheck



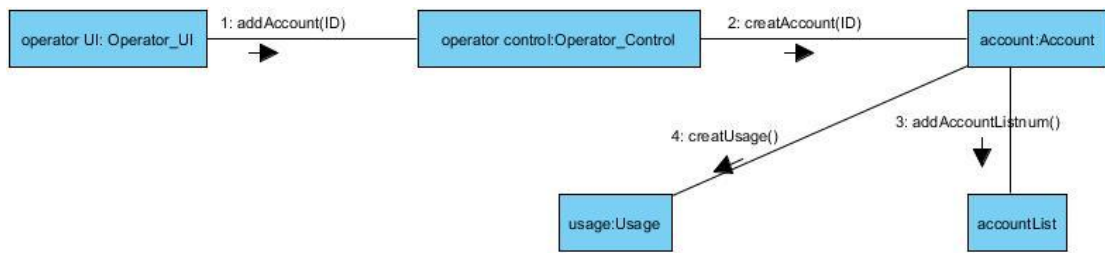
4. exitCheck



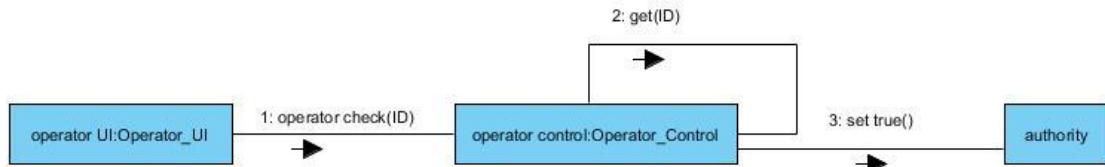
5. search



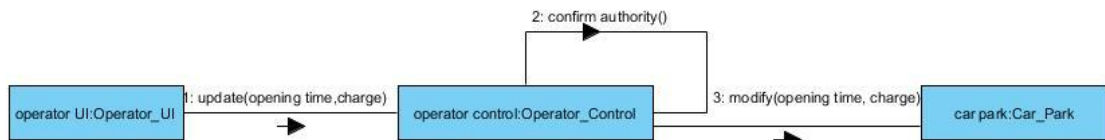
6. addAccount



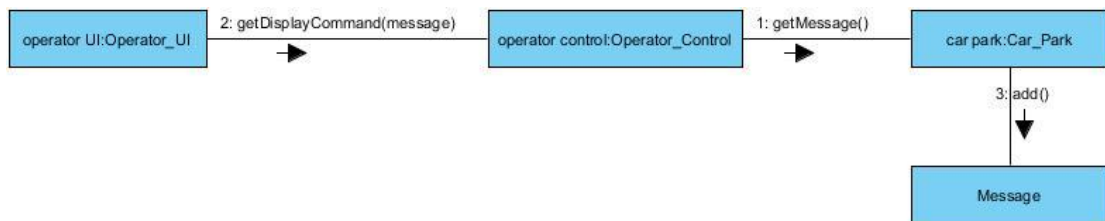
7. operatorCheck



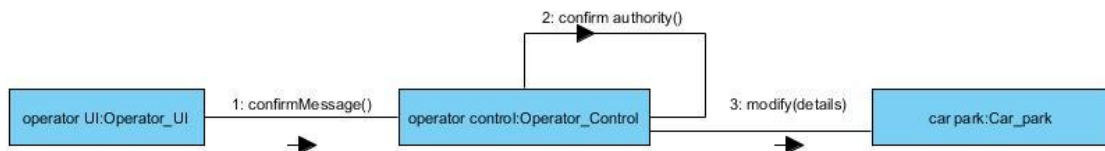
8. update



9. getDisplayCommand



10. confirmMessage



4. Design a class

1) Identifying operations and attributes

Class: Enter_UI

Identifying Operations:

Responsibilities: this class is responsible for the interaction between drivers and the entrance device. It enables drivers to take tickets or scan campus cards at the entrance.

Use case realization: enter

Class: Payment_UI

Identifying Operations:

Responsibilities: this class is responsible for the interaction between drivers and the payment station device. It gets IDs from drivers and enables public drivers to pay coins. It also displays parking hours and amount of money to be paid to the drivers.

Use case realization: payment

Class: Exit_UI

Identifying Operations:

Responsibilities: this class is responsible for the interaction between drivers and the

exit device. It can scan campus cards and take tickets back.

Use case realization: exit

Class: Operator_UI

Identifying Operations:

Responsibilities: this class is responsible for the interaction between the operator and the system. It provides the operator with a user interface with can insert username and password. It enables the operator to modify information of the car park, register staff accounts, and get warning messages of maintenance.

Use case realization: maintain, register, update

Class: Usage

Identifying Operations:

Responsibilities: This control class is responsible for the normal use for drivers of the car park. It can call methods in other entity classes to perform the operation of driver entering and driver paying, change the amount of money in cash box according to the payment, and get to know the amount of money in the cash box.

Use case realization: enter, payment, exit

Identifying Attributes:

```
public String id; // default ID is string
public int number; // the number of parking place
public Date enter_time; // default enter time is date type
public Date payment_time; // default payment time is date type
public double charge; // default one charge is float
public double collection; // default one collection is float
public float parking_hours; // default parking hour is float
public int accountlistnum; // default number of account list is int
public boolean type; // define account type
```

Class: Account

Identifying Operations:

Responsibilities: this class is responsible for managing the drivers' accounts when contents inside are to be modified. It can get and set appropriate payment state of a specific account, calculate and record the charge and real amount of money collected, get the data of charge and collection of a specific account.

Use case realization: enter, payment, exit, gather

Identifying Attributes:

```
protected String card_number; // define the ID of account
protected int payflag; // define the payment state of an account
```

Since all these attributes are inherited by the child classes of Account class, Public_Account and Staff_Account, they are all declared as protected access modifier.

Public_Account

```
public int accountNum; // default account number is string
```

Staff_Account

```
public double accountCharge // monthly charge of the staff account
```

Class: Driver_Control

Identifying Operations:

Responsibilities: This control class is responsible for the normal use for drivers of the car park. It can call methods in other entity classes to perform the operation of driver entering and driver paying, change the amount of money in cash box according to the payment, and get to know the amount of money in the cash box.

Use case realization: enter, payment, exit

Identifying Attributes: None

Class: Car_Park

Identifying Operations:

Responsibilities: this class is responsible for the management of parameters related to the car park. It can change the opening time to public, change the remaining parking spaces according to the amount of cars entering, change the tariffs to public, change the remaining number of remaining blank tickets, and get warning messages of maintenance.

Use case realization: enter, payment, maintain, update

Identifying Attributes:

```

static public int amount_of_tickets; // default amount of tickets is int
static public double amount_of_coins; // amount of coins in the collection box is float
static public double collection; // the collection in the collection box
static public int parking_space; // default parking place is int
static public int remaining_space; // the remaining space of the car park
static public int remaining_ticket; // the remaining tickets in the ticket box
static public double max_money=10; // max money is int with an initial value £10
static public double price[]; // price of parking shown to the driver is float
static public Date opening_time[]; // default opening time is date type
    
```

Class: Operator_Control

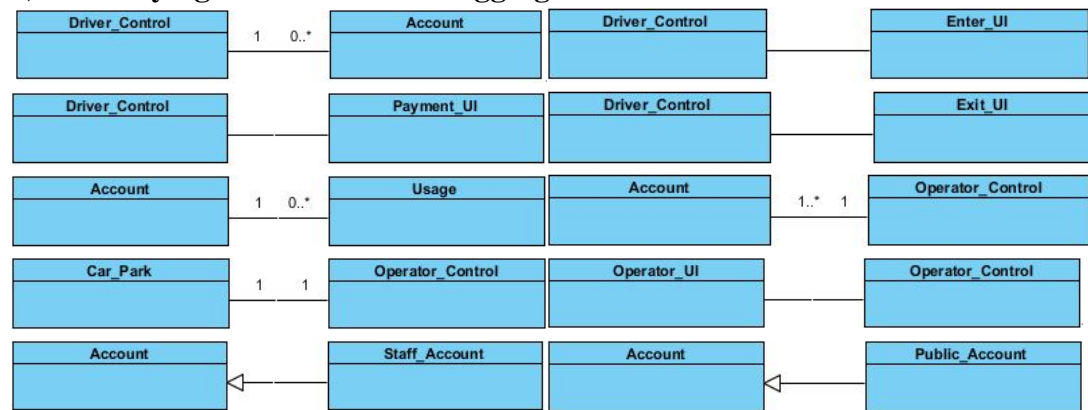
Identifying Operations:

Responsibilities: this control class is responsible for the maintenance and other operations apart from the driver use. It controls the process to register new staff accounts, update information about the car park, show warning messages on screen to remind the operator of maintenance, send monthly bill to the financial department and clear the monthly counting charge.

Use case realization: maintain, update, register, gather

Identifying Attributes: None

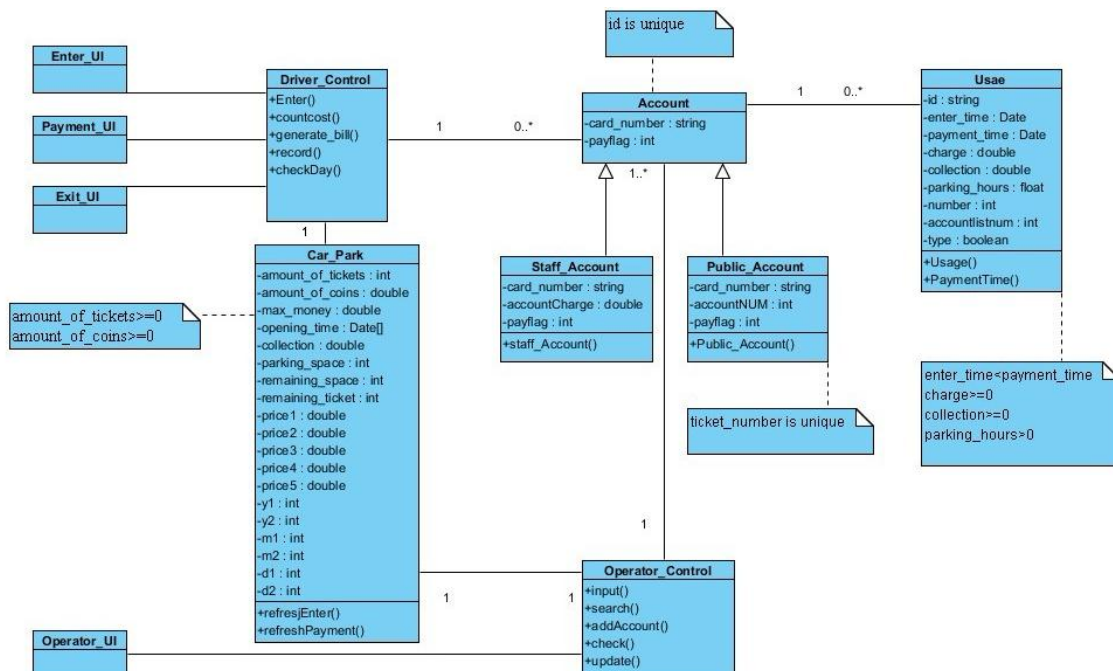
2) Identifying Associations and Aggregations



3) Describing methods

Methods are identified in IMPLEMENTATION.

5. Class diagrams



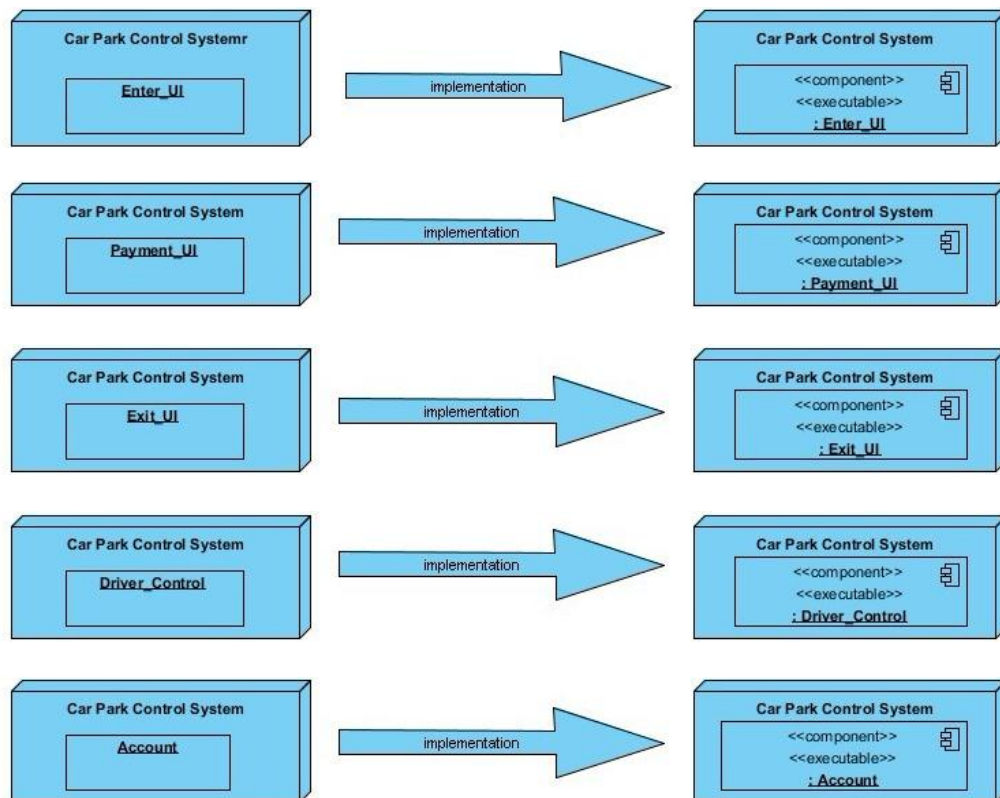
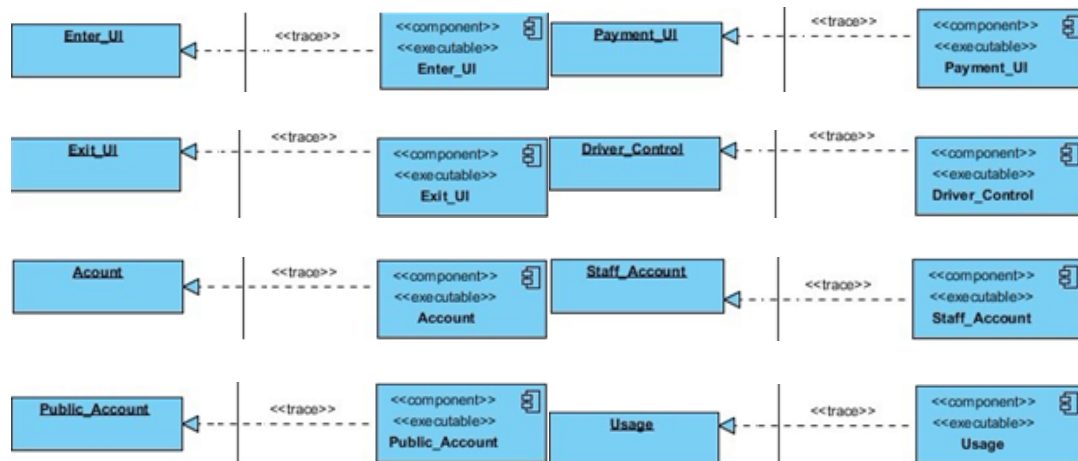
6. Re-usability of system components

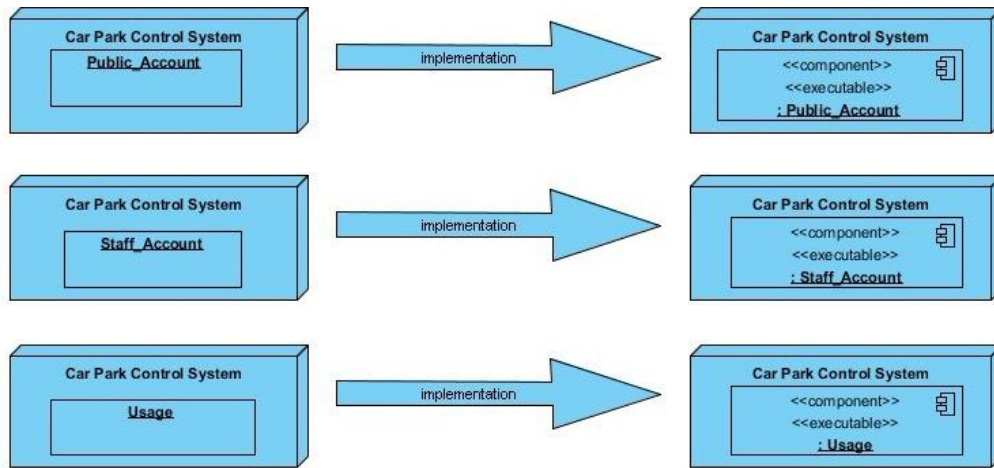
In our design, account is a parent class to be inherited by child classes. As we show in the diagram, account class has two attributes to be inherited by its child classes (Staff_Account and Public_Account). So if we extend our car park and there are more types of driver, it will be easy to add types of account, and we can simply modify some code in our design to change the type of payment without impact on others and whole system.

IMPLEMENT

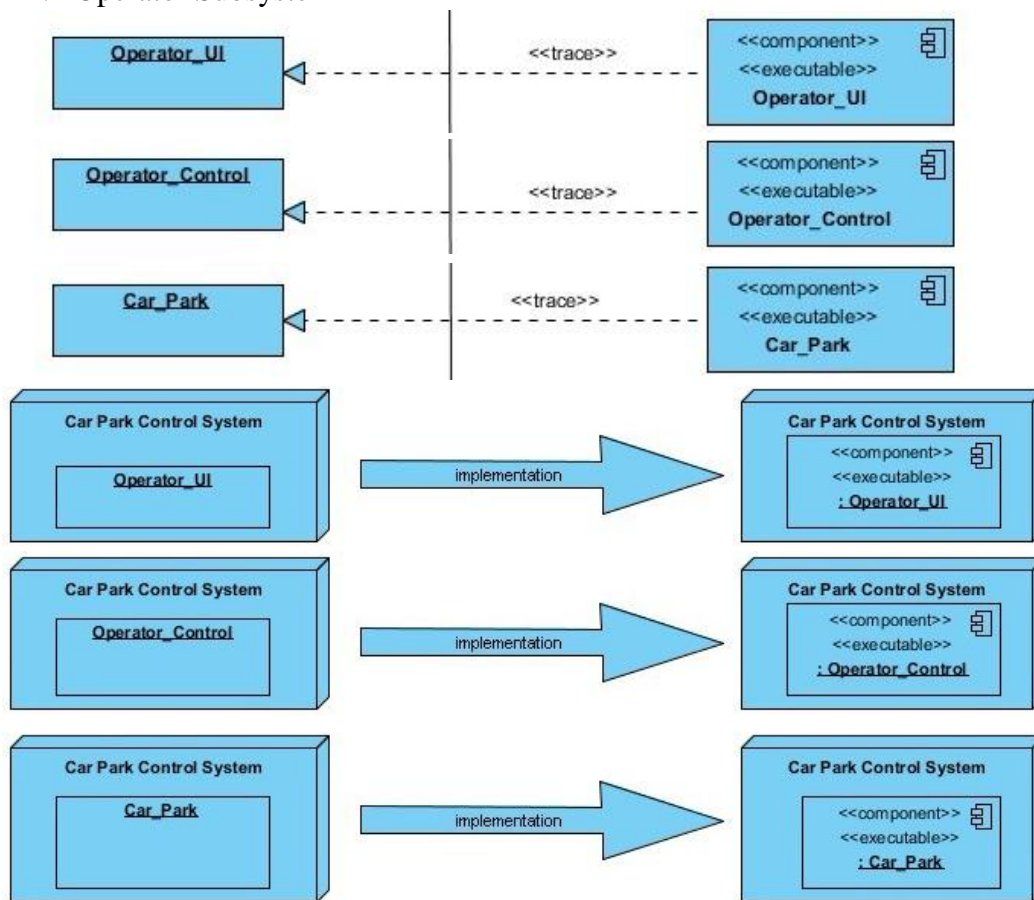
1. Architectural implementation

1.1 Driver Subsystem





1.2 Operator Subsystem



2. Integrate System

Build 1.0	
Functionality:	
1. Record ticket number	
2. Set state	
3. Change parking spaces	
4. Check the collection	
5. Calculate money	
6. Accept the ticket	

Build 2.0	
Build 1.0 is included	
Functionality:	
1. Search the ID number	
2. Register the staff	
3. Be updated the current data	
4. Inform operator to maintain	
Use cases:	
Register Staff; Update data;	

Build 3.0	
Build 2.0 is included	
Functionality:	
1. Send the bill	
Use cases:	
Gather Reports	
Subsystem stems:	
Operator subsystem; driver subsystem	
Components:	
Account.java	

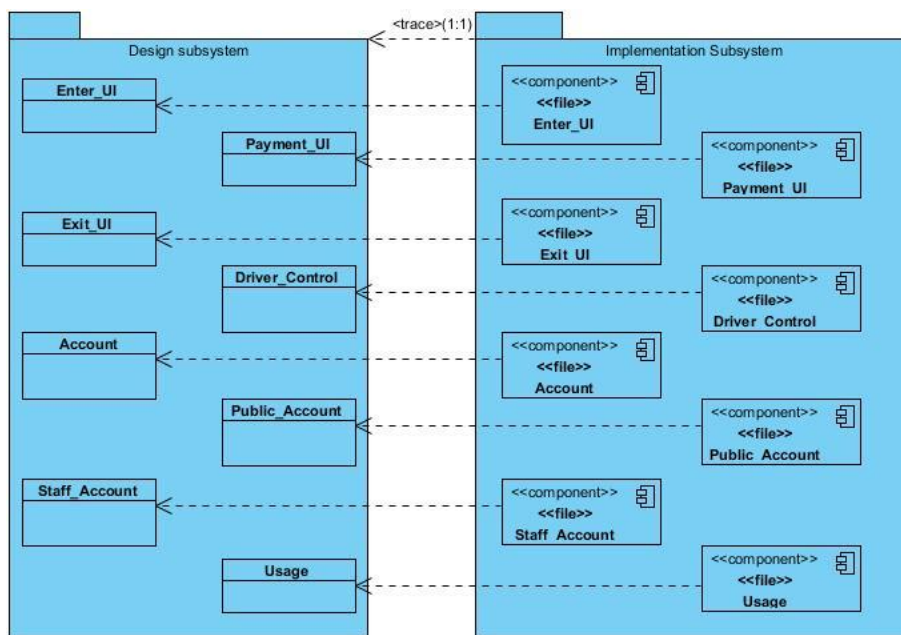
Use cases:
 Enter; Payment; Exit
 Subsystem:
 Driver subsystem;
 operator subsystem
 Components:
 Account.java
 Usage.java
 CarPark.java
 PublicAccount.java
 StaffAccount.java

Maintenance
 Subsystems:
 Operator subsystem;
 driver subsystem
 Components:
 Account.java CarPark.java
 StaffAccount.java

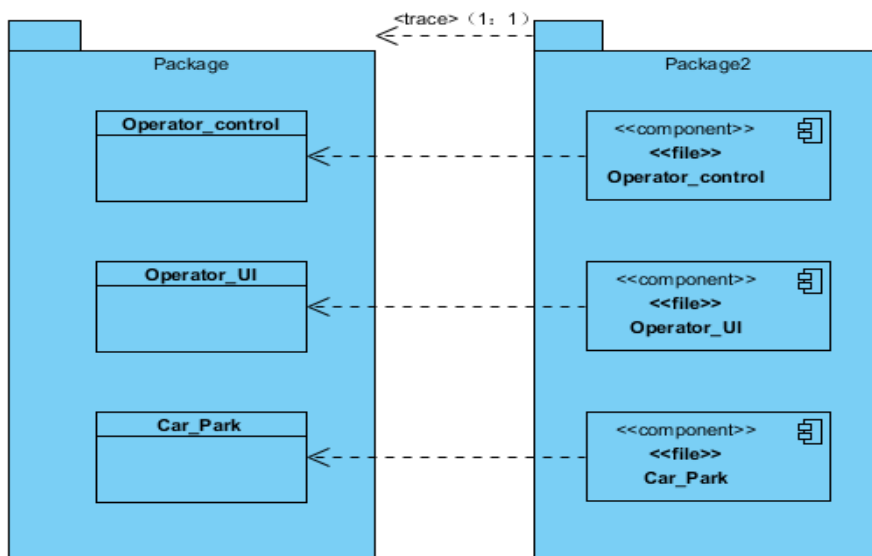
Usage.java
 StaffAccount.java

3. Implement a subsystem

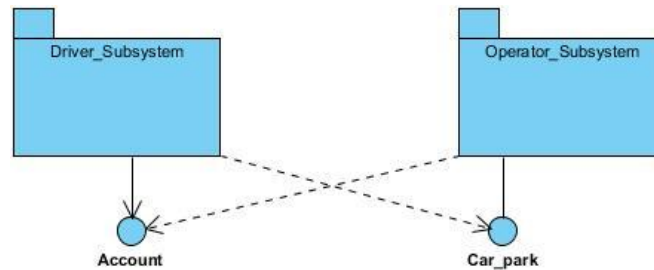
3.1 Driver Subsystem



3.2 Operator subsystem



3.3 Interfaces between subsystems



4. Implement a class

```

public class Driver_Control {
    public void enter();
    public void record();
    public double countcost(Usage
temp);
    public void generate_bill();
    public boolean checkDay(Date
date); }
abstract class Account {
    protected String card_number;
    protected int payflag; }
public class Staff_Account extends
Account {
    public double accountCharge;
    Staff_Account(String id); }
public class Public_Account extends
Account {
    public int accountNum;
    Public_Account(String id); }
public class Usage {
    public String id;
    public int number;
    public Date enter_time;
    public Date payment_time;
    public double charge;
    public double collection;
    public float parking_hours;
    public int accountlistnum;
    public boolean type;
    public void paymentTime();
    public void Usage(String id); }
  
```

```

public class Operator_Control {
    public void input();
    public void search();
    public void addAccount();
    public void check();
    public void update(); }
public class Car_Park {
    static public int amount_of_tickets;
    static public double
amount_of_coins;
    static public double collection;
    static public int parking_space;
    static public int remaining_space;
    static public int remaining_ticket;
    static public double
max_money=10;
    static public double
price1,price2,price3,price4,price5;
    static public Date opening_time[];
    static int y1=0;
    static int y2=0;
    static int m1=0;
    static int m2=0;
    static int d1=0;
    static int d2=0;
    static Date date1=new Date();;
    static Date date2=new Date();
    public String refreshEnter();
    public String refreshPayment(); }
  
```

5. Perform Unit Tests

We test the implemented components as individual units. Various testing techniques have been used in it. We cover this in the Testing section.

TESTING

1. Plan test

1.1 describing a Testing Strategy

In component testing level, we adopt white-box testing to test the internal program logic; at each build stage, regression testing is used to test the functionality of each build, and we use black-box testing in the system testing level to test whether the functional requirements are met.

- 50% of the tests will be automated and the remainder would be manual. This is to check the user interface is correct and the system can perform as expected. Each subject use case should be tested for its normal flow and behavior and other alternative flows if there are any. These should ensure that they cover incorrect

- user input.
 - Success criteria—85% of the test cases passed.
 - All high and medium priority defects resolved.
 - Component testing level
 1. Login--The operator can login the system with valid ID and password.
 2. Add new staff account--The operator (once logged in) can add new staff account if the staff ID is already in the university database.
 3. Update data--The operator (once logged in) can update data (opening time, tariffs) about the car park
 4. Confirm warning information--The operator can confirm the warning information by clicking on specific button appearing on the operator user interface so that corresponding data (amount of money in cash box, number of blank tickets) would be modified by the system.
 5. Entering recording--The system would record the ID, enter time into the corresponding account once a registered driver id is input to the system at the entrance user interface. The payment state of this account should be set to 'unpaid'.
 6. Decrement available parking spaces--The number of available parking spaces should decrease by one when a registered driver id is input to the system.
 7. Decrement number of blank tickets--The number of blank tickets should decrease by one when a registered public id is input to the system.
 8. Payment recording--The system would record the id, payment time into the corresponding account once a registered driver id is input to the system through the payment user interface. Amount of charge and parking hours should be calculated and recorded. For public driver, the real amount of money paid should be recorded once it's input to the system. The payment state of this account should be set to 'paid'.
 9. Increment available parking spaces--The number of available parking spaces should increase by one when an id has
 10. Increase the amount of money in cash box--The real amount of money paid should be added to the cash box once it's input to the system.
 - System testing level
 1. serve by the software
Tests of entering recording, decrement available parking spaces, decrement number of blank tickets, payment recording, increment available parking spaces and increase the amount of money in cash box should be processed as a test for the whole subsystem.
 2. operate with the software
Tests of login, add new staff account, update data and confirm warning information should be processed as a test for the whole subsystem.
- 1.2 estimating the requirements to the testing effort
Operating system: Windows XP, Vista, 7
Platform: java
Human resources:
1 Testing engineers, 3 component testers, 2 system testers
- 1.3 scheduling of the testing effort
The testing process would last about 1 week.
Day 1: plan the test
Day 2: design the test into detail
Day 3: implement the test
Day 4: perform integration test and system test
Day 5: evaluate the test
- 2. design test**
Component test cases:
- Login (other tests for login are in the Appendix 4)
Test case for login (incorrect input)
 - Input
A valid operator account exists; operator ID is OP0001 and password is 1234.
Account ID OP0001 has been entered and password 1243 has been entered.

The system carried out the check.

- Result

The system confirms that OP0001 exists and the associated password is incorrect.

- Conditions

No conditions exist on this test

Test procedure for login (incorrect input)

From the login screen, enter the operator ID OP0001 and password 1243 in the appropriate boxes and select enter. The system checks the operator ID and password and returns “Incorrect operator ID and password, please try again”, and the login screen reappears below the message.

- Update (other tests for update are in Appendix 4)

Test case for modify open time (correct input)

- Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

Delete all the values in the open time blanks.

The system carried out the check.

- Result

The system accepts the new open time and updates it in the system

- Conditions

Operator has logged in.

Test procedure for modify open time (correct input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, delete all the values in the open time blanks and select update. The system checks the new open time, update the new open time in the system and returns “New park info has been updated successfully”, the park info screen reappears below the message.

Test case for modify tariff (incorrect input)

- Input

The tariff exists in the system; values are 1, 2, 3, 4, and 5.

New values, 1, 2, 3, 4 and 10000 have been entered in the blanks of tariff.

The system carried out the check.

- Result

The system does not accept the new tariff.

- Conditions

Operator has logged in

Test procedure for modify tariff (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new tariff, 5, 4, 3, 2 and 1 have been entered in the blanks of tariff and select update. The system checks the new tariff, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

Tests for register, confirm, enter record, and payment record are in the Appendix 4.

System test cases:

Test case for staff use (correct input)

- Input

A valid staff account exists in the system, account ID is S000001; the remaining space is set to 2; the account state of 1 is paid

Account ID S000001 has been entered in enter UI.

Account ID S000001 has been entered in payment UI.

Account ID S000001 has been entered in exit UI.

The system carried out the check.

- Result

The system adds charge to the account according to the tariff of staff.

- Conditions

No conditions exist on this test.

Test case for staff use twice (correct input)

• Input

A valid staff account exists in the system, account ID is S000001; the remaining space is set to 2; the account state of S000001 is paid; account date is the date of today.

Account ID S000001 has been entered in enter UI.

Account ID S000001 has been entered in payment UI.

Account ID S000001 has been entered in exit UI.

The system carried out the check.

• Result

The system adds charge, which equals zero, to the account. There is no stop during the operation

• Conditions

No conditions exist on this test.

Test case for public use (correct input)

• Input

A valid public account exists in the system, account ID is 2; the remaining space is set to 2; the account state of 2 is paid; the open time is set to 1900, 01, 01, 9999, 12, 31; tariff are set to 2, 3, 4, 5, and 6

Account ID 2 has been entered and ticket bottom is pressed in enter UI.

Account ID 2 has been entered in payment UI.

\$1 been entered in payment UI.

\$1 been entered in payment UI once again.

Account ID 2 has been entered in exit UI.

The system carried out the check.

• Result

The system adds 2 to the collection and decreases 1 from the remaining tickets.

• Conditions

No conditions exist on this test.

Test procedure for staff use (correct input)

1. From the entrance screen, enter the ID S000001 in the appropriate area. The system checks the ID, remaining space and the account state, opens the entrance barrier, records the enter time in the account S000001; changes the account state of S000001 to unpaid, decrease the remaining space by 1
2. From the payment screen, enter the ID S000001 in the appropriate area. The system checks the ID, account state and account date; records the exit time in the account S000001; get the enter time and tariff, calculate the charge and record charge in the account.
3. From the exit screen, enter the ID S000001 in the appropriate area. The system checks the ID and the account state, opens the entrance barrier.

Test procedure for staff use twice (correct input)

1. From the entrance screen, enter the ID S000001 in the appropriate area. The system checks the ID, remaining space and the account state, opens the entrance barrier, records the enter time in the account S000001; changes the account state of S000001 to unpaid, decrease the remaining space by 1
2. From the payment screen, enter the ID S000001 in the appropriate area. The system checks the ID, account state and account date; records the exit time in the account S000001; get the enter time and tariff, calculate the charge (equals zero) and record charge in the account.
3. From the exit screen, enter the ID S000001 in the appropriate area. The system checks the ID and the account state, opens the entrance barrier.

Test procedure for public use (correct input)

1. From the entrance screen, enter the ID 2 in the appropriate area and select ticket button. The system checks the ID, remaining space, open time and the account state; gives the ticket, opens the entrance barrier, records the enter time in the account 2; changes the account state of 2 to unpaid, decrease the remaining space and the amount of tickets by 1.

2. From the payment screen, enter the ID 2 in the appropriate area. The system checks the ID and the account state; records the exit time in the account 2; get the enter time and tariff, calculate the charge and record charge in the account, show the charge on payment screen.
3. \$1 been entered in payment UI twice. The system checks the collection and compares it to the charge, set the account state to paid when the collection is bigger than charge; add the collection in the car park.
4. From the exit screen, enter the ID 2 in the appropriate area. The system checks the ID and the account state, opens the entrance barrier.

Example - Test matrix (other matrixes are in APPENDIX 4)

Test case	Test description	Pass/ Fail	No. of Bugs	Bug #	Comments
Set up 4	The max money is set to 100 and the amount of coins is set to 100; The max tickets is set to 100 and the amount of tickets is set to 0				Add to system for testing
4.1	Test press the money confirm button				
4.2	Test press the tickets confirm button				

3. Implement Test

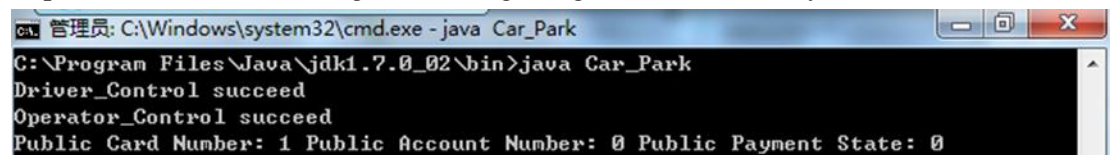
3.1. Automatic Test

For the component level we mainly write the test in the implementation stage.

For the system level, the test code is as shown in APPENDIX 4.

3.2. Manual Test

The purpose of implementing tests is to automate test procedures by creating test components. We use database application for different inputs and resulting outputs. We implement the test according to the design stage, and record every result.



```

line1 0P0001
line2 1234
ID 0P0002
Password 0000
line1 0P0002
line2 0000
ID 0P0002
Password 0000
Account: 11
the collection1 is 0.0
the collection2 is 0.0
the collection3 is 15.0
The park max is 100.0
The park coin is 22.0
In the end, accountiflag number is0
    
```

4. Perform Integration Test

Defect number: 001

Defect Title: Illegal time can be input into system.

Test number: 3.4

Description: The system can't identify whether the time is legal or not and accept all.

Assigned to Component Engineer: Zhao Chenyu

Raised by/assigned to Test Engineer: Liu Jiawei

Defect number: 002

Defect Title: Inexistent account entered in car park.

Test number: 5.4

Description: The account which doesn't exist in system can enter in the car park with the permission of system.

Assigned to Component Engineer: Zhao Chenyu

Raised by/assigned to Test Engineer: Liu Jiawei

5. Perform system test

Requirement	Functionality	If	Where
-------------	---------------	----	-------

		finished	
Create an unpaid ID	Add new usage with information	Finished	Decrease a space in Enter_UI
Account the charge	Read the information in the usage and account the charge should be paid and change the payment state	Finished	Show the information and charge in Payment_UI
Read the payment state	Read the payment state and pass the car if paid	Finished	Add a space of park in Exit_UI
Update data	Modify the information of park	Finished	Change the information in Operator_UI
Monitor state of park	Obtain the instant information from entrance and payment	Finished	Receive information in Operator_UI

We test all parts of system and put them together to test the whole system. And we can see that all parts work well and satisfy the basic requirements. There isn't serious defect in the whole system.

6. Evaluate test

We have done all tests and defects are found. The corresponding component engineers are informed to correct the defects.

Testing completeness

All codes and functions of the system have been tested twice.

Reliability

The system can meet the requirements after defects are corrected. Although some further functions can be added to the system to improve it and extra defects may occur as well, the system is ready to be released.

CONCLUSION

Following the steps of capturing original requirements, analyzing the whole process, designing the system, implementing the design through JAVA programming and finally completing the testing, our group has completed a car park control system that meets given requirements without delay.

Although the system has passed the testing, there are still little defects to be removed and the efficiency and ease of use can still be improved. Since we have considered the possible advanced functions to be added to the system, we have designed some interfaces in order to make further update easily.

REFERENCE

- [1] McLaughlin, Brett, Gary Pollice, and David West. Head First Object-oriented Analysis and Design. Sebastopol, CA: O'Reilly, 2007. Print.
- [2] Vliet, Hans Van. Software Engineering: Principles and Practice. Chichester [England: John Wiley, 2000. Print.
- [3] Stevens, Perdita, and R. J. Pooley. Using UML: Software Engineering with Objects and Components. Harlow: Addison-Wesley, 2000. Print.

Interview Dr. Marie-Luce Bourguet	2.28
Integrate the information from the interviews	2.29
Get the works done of the team and work them into “Applying Requirements Finding Techniques”	3.3
Participate in group discussion on the function of the system	3.6

<Siyuan LIU, jp092986>

Tasks	Milestones
Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books	2.25
First group meeting for the subgroups and individual tasks	2.26
Decide the questions for interviews	2.27
Do background reading and search the existing examples of parking system	2.28
Integrate the information collected by each subgroup member	2.29
Participate in group discussion on the function of the system	3.5
Capture the functional requirements and non-functional requirements	3.6

Subgroup2:

<Yao PING, jp092985>

Tasks	Milestones
Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books	2.25
First group meeting for the subgroups and individual tasks	2.26
Design the questionnaire and distribute it on the Internet	2.28
Collect the results of the survey and analyze the results	2.29
Participate in group discussion on the function of the system	3.5
Capture and quantify the non-functional requirements	3.6

<Chenyu ZHAO, jp093165>

Tasks	Milestones
Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books	2.25
First group meeting for the subgroups and individual tasks	2.26
Finish the observation and draw the GANTT charts	2.28
Write a report about Observation	2.29
Participate in group discussion on the function of the system	3.5
Capture and quantify the non-functional requirements	3.6

<Jiawei LIU, jp093167>

Tasks	Milestones
Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books	2.25
First group meeting for the subgroups and individual tasks	2.26
Design the questionnaire and draw the GANTT charts	2.28
Collect the results of the survey and analyze the results	2.29
Participate in group discussion on the function of the system	3.5
Analyze and integrate each part of non-functional requirements	3.6

Subgroup3:

<Zhijiao LIU, jp092987>

Tasks	Milestones
Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books	2.25
First group meeting for the subgroups and individual tasks	2.26
Design the questions of the interviews and do the interviews in BUPT main campus car park	2.27
Interview Dr. Frank Gao and Andy Watson	2.28
Integrate the information from the interviews	2.29
Participate in group discussion on the function of the system	3.5
Finish the analysis of the functional requirements	3.6

<Yang LIU, jp092989>

Tasks	Milestones
Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books	2.25
First group meeting for the subgroups and individual tasks	2.26
Do background reading and search the existing examples of parking system	2.28
Integrate the information collected by each subgroup member	2.29
Participate in group discussion on the function of the system	3.5
Analyze the functional requirements	3.6

Week Summary

Accomplishments:

1. After the first meeting, leader and three subgroups are decided
2. Using fact-finding techniques to complete Background Reading, Interviews and questionnaires
3. Finishing investigation and doing the analysis based on fact-finding data
4. Functional and non-functional requirements are analyzed.

Each group member gets familiar with each other and finishes the tasks on time. Some problems about the fact-finding techniques are solved by group discussion.

Weekly report**EBU5304 Software Engineering – Group Coursework****Weekly Report for:** <from 2.24 to 3.6>**Group Number:** <No.47>**Individual Member:** <Yao PING, jp092985>**Tasks and any key milestones for this week and progress**

- Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books
- Get to know the group members and divide our team into three subgroups.
- Design questionnaire of using car park system.
- Analyze the results of the questionnaire.
- Review the lecture to see how to capture requirements

- Gather and analyze the requirements get in all of the requirements finding techniques
- Understand the system context
- Capture and quantify the non-functional requirements

Impact of issues and delays on task completion and milestones

- The result of online questionnaire is inadequate. So we also finish some questionnaires in our campus.
- The considering of non-functional requirements gets blocked, and it continues after discussion inside the subgroup and review of information.

Tasks for next week

Next week we will do the use case model and draw the prototype

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- I study the requirement of the course work.
- I met other team members and exchange phone number.
- We divide into 3 subgroups.
- We discuss with the requirement.
- I complete the background of fact-finding
 - We decide the question which we want to ask in the interview
 - Analyze the answer we get.
- Capture the functional requirement and non-functional requirement

Impact of issues and delays on task completion and milestones

- In the first time, our background is not complete, and we add something to it.
- A little late is resulted by the schedule of teacher

Tasks for next week

Decide use case and prototype

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Read coursework requirements, lecture notes and background knowledge to get a better understanding of the whole requirements.
- Contact group members, generate a team leader and email Paula. Discuss with all group members and decide the subgroups and individual work.
- Go to BUPT main campus and Teaching Building One to do the interviews.
- Conclude and analyse the responds to meet the needs of fact-finding aspects.

- Based on the results of using the fact-finding techniques, discuss the work assignments of capturing requirement.
- Discuss with the whole group to determine the logic relationship of each part of this system.
- Based on 4 aspects we defined, Entrance Management, Payment station, Exit Management and Main control, finish the analysis of the functional requirements.
- Discuss the parameters of each aspect and prepare for the use case and prototype.

Impact of issues and delays on task completion and milestones

- Because one of our interviewees has not come to Beijing, we have to finish our interview by Email.
- When consider the functions and requirements, everyone has a different view about the aspects of our control system. So it is hard to reach agreement. We discuss for a long time and finally reach an agreement.

Tasks for next week

Analyse the requirements and draw the use case diagrams

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Read coursework requirement and do the background readings to get more understanding of the coursework.
- Contact group members and divide subgroup.
- Discuss the task together and get a better understanding of the whole requirements.
- Try to find out some information related to the project and get a further understanding of the background of this project.
- Based on the result of using the Fact-finding Technique, discuss the work assignment of capturing requirement. We divide into 3 subgroups, based on the requirements classification, I was assigned to describe what the system should do –Functionality.
- Read the knowledge about functional requirements in our class in order to do the work better.
- Apply the basic knowledge to this case, do the analysis from 4 aspects, including Entrance Management, Payment station, Exit Management, Main control. All the discussion is based on the original material.
- Meet the other member of the subgroup which is mainly in charge of software programming. Discuss the overall algorithm of the software and the way to cooperate in the further.
- Within the subgroup, try to find out the rough prototype ,which is needed to discuss to the whole team next week.

Impact of issues and delays on task completion and milestones

- When looking for the information about existing system, we observe and analyze many kinds of system, which causing a little delay in time.

- When try to find the basic rough prototype, due to the limited number of member and not familiar with the project, which causes some delay.

Tasks for next week

Finish what's left on the requirements and move on the analysis

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Going through the coursework requirements, get familiar with the content of the lecture slides and other materials.
- Meet the members of the group and distribute the general task of each member.
- Go to BUPT main campus and Teaching Building One to do the interviews.
- Conclude and analyse the responds to meet the needs of fact-finding aspects.
- Get the background reading, interviews and questionnaires done last week from the group members. Working them into a complete version of “Applying requirements finding techniques”
- Carrying on group discussion on the function of the system

Impact of issues and delays on task completion and milestones

The delay of interview will result in the block of requirements writing, then the programming of the system will also be delayed.

Tasks for next week

Finish the use case diagram with other group members

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and other materials if need.
- Connect and get familiar with all group members and discuss to divide the group into subgroups.
- Do background reading and be in charge of the parking policy description of the existing parking system in QM.
- Collect last week's results from all members including background reading, interviews and questionnaires. Polish and synthesize them into a complete version of “Applying requirements finding techniques”
- Participate in group discussion on the function of the system, and make system assumptions

Impact of issues and delays on task completion and milestones

- Synthesizing of the whole background reading cannot be conducted if the policy description is not finished.

- The delay of assumption will block the programming of the system since the flow path is not in detail

Tasks for next week

We will start working on use case

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

- Read and analyze the Coursework requirements and constrains
- Review the lecture and search relative information
- Complete the GANTT charts.
- Design the details of one kind of requirements finding technique-Observation
- Go to a car parking to do observation and record information
- Write a report about Observation
- Review the lecture to see how to capture requirements
- Gather and analyse the requirements get in all of the requirements finding techniques
- Understand the system context
- Capture and quantify the non-functional requirements

Impact of issues and delays on task completion and milestones

Further discussion cannot be launched if full understanding of the coursework requirements is not obtained.

Tasks for next week

Next week we begin to discuss use cases

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

- Understand the coursework requirements, and have a basic thought about how to do the task referring to lecture slides and reference books
- Get to know the group members and divide our team into three subgroups.
- Complete the GANTT charts.
- Design questionnaire of using car park system.
- Analyze the results of the questionnaire.
- Analyze non-functional requirements
- Finish non-functional requirements.
- Art design and make spreadsheets
- Modify the whole part

Impact of issues and delays on task completion and milestones

The questionnaire may not reflect all the customers' opinions well. So we do researches as much as we can, and design more comprehensive convey.

Tasks for next week

We will work on use cases.

Week 2

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Group Members:

<Yao PING, jp092985>	<Tianshu ZHANG, jp092990>
<Siyuan LIU, jp092986>	<Tong GE, jp092991>
<Zhijiao LIU, jp092987>	<Chenyu ZHAO, jp093165>
<Yang LIU, jp092989>	<Jiawei LIU, jp093167>

Group Leaders Name: <Zhijiao LIU, jp092987>

Project Progress Summary

GANTT

chart:

ID	Task	Start time	Finish time	Duration	March 7th				March 12th			
					7	8	9	10	11	12	13	
1	Finding the actors	2012/3/7	2012/3/7	1d	█							
2	Build the use case model	2012/3/8	2012/3/8	1d		█						
3	Refine use case model	2012/3/9	2012/3/9	1d			█					
4	Plan the overall codes and structure of the system	2012/3/12	2012/3/12	1d							█	

Individual tasks

Subgroup1:

<Tong GE, jp092991>

Tasks	Milestones
Discuss the actor and use case around the group	3.7
Define the actor and use case model	3.9
Draw the diagram of register and maintain	3.12
Discuss the prioritize use case around group	3.13

<Tianshu ZHANG, jp092990>

Tasks	Milestones
Discuss the actor and use case around the group	3.7
Define the actor and use case model	3.9
Draw the diagram of enter and exit	3.12
Discuss the prioritize use case around group	3.13

<Siyuan LIU, jp092986>

Tasks	Milestones
Discuss the actor and use case around the group	3.7
Define the actor and use case model	3.9
Draw the diagram of payment station	3.12
Discuss the prioritize use case around group	3.13

Subgroup2:

<Yao PING, jp092985>

Tasks	Milestones
Participate in group meeting about the use case	3.7
Refine use case model.	3.11
Capture special requirements.	3.12
Prioritize use case.	3.13

<Chenyu ZHAO, jp093165>

Tasks	Milestones
Participate in group meeting about the use case	3.7
Refine use case model.	3.11
Capture special requirements.	3.12
Prioritize use case.	3.13

<Jiawei LIU, jp093167>

Tasks	Milestones
Participate in group meeting about the use case	3.7
Refine use case model.	3.11
Capture special requirements.	3.12
Prioritize use case.	3.13

Subgroup3:

<Zhijiao LIU, jp092987>

Tasks	Milestones
Review the notes and lectures of this part.	3.7
Plan the overall codes and structure of the system.	3.9
Begin to write the structure.	3.12
Arrange the works of next week.	3.12

<Yang LIU, jp092989>

Tasks	Milestones
Review the notes and lectures of this part.	3.7
Plan the overall codes and structure of the system.	3.9
Begin to write the structure.	3.12
Arrange the works of next week.	3.12

Week Summary

1. We complete the use case diagram and description
2. The design of prototype is complete
3. We refine the use case

Each group member gets familiar with each other and finishes the tasks on time. At first we define use cases in a wrong way because of the unclear of their definition, so we have to revise them with great efforts thus lag behind the schedule.

Weekly report

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Individual Member: <Yao PING, jp092985>

Tasks and any key milestones for this week and progress

- Writing detailed description of process the payment and register staff.
- Participated in prioritizing use case jobs.
- Discuss with group mates and refine use case.

Impact of issues and delays on task completion and milestones

When I started writing use case, I'm confused about the process of finding actor and use case. But with the help of my group mates and Searching more material, this problem was solved.

Tasks for next week

- Design user interface of log in and registration.
- Consider UI of modifying car park information

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of coursework about this part.
- Discuss with group and define the actors and model.
- Diagram of payment station.
- Discuss the prioritize use case type with group.

Impact of issues and delays on task completion and milestones

We complete the first diagram. But when we put them in the Word Document, the diagram is too little to see the text clearly. So we draw a new one and change the format of text.

Tasks for next week

- Discuss and decide the prototype.
- Draw the interface.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.7>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Review the notes and lectures of this part.
- Plan the overall codes and structure of the system.
- Begin to write the structure.
- Arrange the works of next week.

Impact of issues and delays on task completion and milestones

- Because the structure is very abstract, so planning the overall codes consumes a lot of time.
- We need to consider the entire member's work when planning the structure.

Tasks for next week

Do the prototype.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.8 to 3.13>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Review the notes and lectures of this part.
- Plan the overall codes and structure of the system.
- Begin to write the structure.
- Arrange the works of next week.

Impact of issues and delays on task completion and milestones

- Because the structure is very abstract, so planning the overall codes consumes a lot of time.
- We need to consider the entire member's work when planning the structure.

Tasks for next week

Finish the prototype.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of coursework about this part.
- Discuss with group and define the actors and model.
- Diagram of enter and exit.
- Discuss the prioritize use case type with group.

Impact of issues and delays on task completion and milestones

We spend a long time deciding the actors and use cases. Once we find we can't continue next step because of the wrong definition of the use cases, we should revise constantly.

Tasks for next week

- Discuss and decide the prototype.
- Draw the interface.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of coursework about this part.
- Discuss with group and define the actors and model.
- Diagram of register and maintain.
- Discuss the prioritize use case type with group.

Impact of issues and delays on task completion and milestones

At first, we find wrong and many use cases. Then we find some use case can be merged into one. So we define our final use cases.

Tasks for next week

- Discuss and decide the prototype.
- Draw the interface.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Individual Member: <Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

- Finding use case for actors.
- Seeking more information about process of use case and made a detailed analysis about enter and exit.
- Discuss with group mates and finish priority.
- Refine the use case with group mates after discuss.

Impact of issues and delays on task completion and milestones

At beginning we choose the wrong actor and it's almost impossible to finish this job. But our group discuss this problem in detail and re-select a better actor. After searching some information about use case and example from the text book, we finish our work successfully.

Tasks for next week

- Draw user interface of log in
- Draw user interface of registration
- UI of modifying car park information

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.7 to 3.13>

Group Number: <No.47>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

- Review the note about use case and write description of use case.
- Discuss with group members and decide the priority.
- Capture the special requirements about update data.
- Refine the use case with group members.

Impact of issues and delays on task completion and milestones

- At the first time we detailed the description of use case, we found it's too complex and it didn't get the point. But after a long time discussion with my group members, we decided to simplify some details and got a better conclusion about use case.

Tasks for next week

Consider the user interface of log in and user interface of registration. Design UI of modifying car park information according to use case.

Week 3

Weekly Report for: <from 3.15 to 3.21>

Group Number: <No.47>

Group Members:

<Yao PING, jp092985>	<Tianshu ZHANG, jp092990>
<Siyuan LIU, jp092986>	<Tong GE, jp092991>
<Zhijiao LIU, jp092987>	<Chenyu ZHAO, jp093165>
<Yang LIU, jp092989>	<Jiawei LIU, jp093167>

Group Leaders Name: <Zhijiao LIU, jp092987>

Project Progress Summary

GANTT

chart:

ID	Task	Start time	Finish time	duration	March 11th			March 18th				
					15	16	17	18	19	20	21	
1	Define use case and relationship between them.	2012/3/16	2012/3/16	1d								
2	Draw diagram of usecase.	2012/3/19	2012/3/19	1d								
3	Detailed description of usecase	2012/3/15	2012/3/15	1d								
4	Capture special requirements	2012/3/19	2012/3/19	1d								
5	Priorities usecase.	2012/3/19	2012/3/19	1d								
6	Prototype user interface.	2012/3/20	2012/3/20	1d								

Individual tasks

Subgroup1:

<Tong GE, jp092991>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Discuss with and analyze functional requirement around group.	3.16
Define the actor and use case model and relationship between these according functional requirements.	3.16
Functional requirements.	3.18
Draw diagram of register and maintain.	3.19
Discuss the prioritize use case type with group.	3.21

<Tianshu ZHANG, jp092990>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Discuss with and analyze functional requirement around group.	3.16
Define the actor and use case model and relationship between these according functional requirements.	3.16
Draw diagram of exit and enter.	3.18

Discuss the prioritize use case type with group.	3.21
--	------

<Siyuan LIU, jp092986>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Discuss with and analyze functional requirement around group.	3.16
Define the actor and use case model and relationship between these according functional requirements.	3.16
Draw diagram of payment station.	3.18
Discuss the prioritize use case type with group.	3.21

Subgroup2:

<Jiawei LIU, jp093167>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Identify boundary,entity and control class.	3.16
Identify Class relationship.	3.18
Draw conceptual class diagram.	3.19
Draw sequence diagram.	3.21

<Chenyu ZHAO, jp093165>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Identify boundary,entity and control class.	3.16
Identify Class relationship.	3.18
Draw conceptual class diagram.	3.19
Draw sequence diagram.	3.21

<Yao PING, jp092985>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Identify boundary,entity and control class.	3.16
Identify Class relationship.	3.18
Draw conceptual class diagram.	3.19
Draw sequence diagram.	3.21

Subgroup3:

<Zhijiao LIU, jp092987>

Tasks	Milestones
-------	------------

Review the note about use case and read requirement of course work about this part.	3.15
Discuss with and analyze functional requirement around group.	3.16
Priorities the uses cases.	3.18
Prototype the user interfaces.	3.19
Finish the prototype.	3.21

<Yang LIU, jp092989>

Tasks	Milestones
Review the note about use case and read requirement of course work about this part.	3.15
Discuss with and analyze functional requirement around group.	3.16
Priorities the uses cases.	3.18
Prototype the user interfaces.	3.19
Finish the prototype.	3.21

Week Summary

Accomplishments:

1. We checked the whole work we have done last week and refined the wrong places.
2. After researched the following materials, we distributed the works.
3. We built the use case models and described them.
4. We finished prototyping the user interfaces.

Group discussion does a great job of the following works and we can do the teamwork better through this. All the members get the proper jobs.

Weekly report

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.22 to 3.28>

Group Number: <No.49>

Individual Member: <Yao PING, jp092985>

Tasks and any key milestones for this week and progress

- According to the notes and discussion with group mates, identify boundary, entity and control class.
- Through analysis with group mates, identify the relationship between classes.
- Draw conceptual diagram for use case.
- Take part in finishing sequence diagram work.

Impact of issues and delays on task completion and milestones

Every time we adjust our diagram, we need to get together to have a more comprehensive analysis. Thus we can ensure our works can be well done.

Tasks for next week

- Design user interface of log in and registration.
- Consider UI of modifying car park information

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 2.24 to 3.6>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of course work about this part.
- Discuss with and analyze functional requirement around group.
- Define the actor and use case model and relationship between these according functional requirements.
- Draw diagram of payment station.
- Discuss the prioritize use case type with group.

Impact of issues and delays on task completion and milestones

- We complete the first diagram. But when we put them in the Word Document, the diagram is too little to see the text clearly. So we draw a new one and change the format of text.

Tasks for next week

Complete analysis step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.15 to 3.21>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of course work about this part.
- Discuss with and analyze functional requirement around group.
- Priorities the uses cases.
- Prototype the user interfaces.
- Finish the prototype.

Impact of issues and delays on task completion and milestones

4. Before prototyping the user interfaces, we have to see the outcomes of the previous steps.
5. When doing the UI, some details are not corresponding to what we have planned, so we have to correct the previous works.

Tasks for next week

Complete the analysis step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.15 to 3.21>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of course work about this part.
- Discuss with and analyze functional requirement around group.
- Priorities the uses cases.
 - Prototype the user interfaces.
 - Finish the prototype.

Impact of issues and delays on task completion and milestones

6. Before prototyping the user interfaces, we have to see the outcomes of the previous steps.
7. When doing the UI, some details are not corresponding to what we have planned, so we have to correct the previous works.

Tasks for next week

Complete the analysis step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.15 to 3.21>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of course work about this part.
- Discuss with and analyse functional requirement around group.
- Define the actor and use case model and relationship between these according functional requirements.
- Draw diagram of exit and enter

- Discuss the prioritize use case type with group.

Impact of issues and delays on task completion and milestones

We spend a long time deciding the actors and use cases. Once we find we can't continue next step because of the wrong definition of the use cases, we should revise constantly.

Tasks for next week

Complete step of analysis

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.15 to 3.21>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Review the note about use case and read requirement of course work about this part.
- Discuss with and analyse functional requirement around group.
- Define the actor and use case model and relationship between these according functional requirements.
- Draw diagram of register and maintain
- Discuss the prioritize use case type with group.

Impact of issues and delays on task completion and milestones

At first, we find wrong and many use cases. Then we find some use case can be merged into one. So we define our final use cases.

Tasks for next week

Do analysis.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.22 to 3.28>

Group Number: <No.49>

Individual Member:<Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

- Identify boundary,entity and control class after discuss with group mates.
- Find the relationship between classes.
- Cooperation with group mates to draw a complete conceptual diagram.
- Draw the sequence diagram.

Impact of issues and delays on task completion and milestones

Understand the courseware and note well made huge impact to task completion. Exchanging opinion with group mates is also important.

Tasks for next week

- Draw user interface of log in
- Draw user interface of registration
- UI of modifying car park information

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.22 to 3.28>

Group Number: <No.49>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

- Review the notes and figure out boundary, entity and control class.
- Connect Class and find their relationship.
- Draw conceptual class diagram with group mates together.
- Sequence diagram relate to use case.

Impact of issues and delays on task completion and milestones

The task completion and milestones have no delays. It's significant to grasp the software for finishing works.

Tasks for next week

Consider the user interface of log in and user interface of registration. Design UI of modifying car park information according to use case.

Week 4

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Group Members:

<Yao PING, jp092985>	<Tianshu ZHANG, jp092990>
<Siyuan LIU, jp092986>	<Tong GE, jp092991>
<Zhijiao LIU, jp092987>	<Chenyu ZHAO, jp093165>
<Yang LIU, jp092989>	<Jiawei LIU, jp093167>

Group Leaders Name: <Zhijiao LIU, jp092987>

Project Progress Summary

GANTT chart:

ID	Task	Start time	Finish time	Duration	March 22nd			March 28th				
					22	23	24	25	26	27	28	
1	Identify Entity, Boundary and Control classes.	2012/3/23	2012/3/23	1d	■	■						
2	Identify class relationships	2012/3/23	2012/3/23	1d	■	■						
3	A conceptual class diagram	2012/3/26	2012/3/26	1d						■	■	
4	Identify attributes for each entity class	2012/3/26	2012/3/26	1d						■	■	
5	Add constraints	2012/3/27	2012/3/27	1d							■	■
6	Sequence diagrams for use cases	2012/3/27	2012/3/27	1d							■	■
7	Operation contracts	2012/3/28	2012/3/28	1d								■

Individual tasks

Subgroup1:

<Tong GE, jp092991>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss classes of entity, boundary and control.	3.23
Discuss relationship of classes.	3.23
Discuss attributes of entity class.	3.23
Write operation contracts of get(TXT).	3.27
Draw diagram of class.	3.28

<Tianshu ZHANG, jp092990>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss classes of entity, boundary and control.	3.23
Discuss relationship of classes.	3.23
Discuss attributes of entity class.	3.23
Write operation contracts of get(enter time)	3.27
Draw diagram of class.	3.28

<Siyuan LIU, jp092986>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss classes of entity, boundary and control.	3.23
Discuss relationship of classes.	3.23

Discuss attributes of entity class.	3.23
Write operation contracts of record(ID, enter time and record(collection))	3.27
Draw diagram of class.	3.28

Subgroup2:

<Yao PING, jp092985>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss attributes of entity class.	3.23
Draw user interface of log in	3.27
Draw user interface of registration	3.27
UI of modifying car park information	3.28

<Chenyu ZHAO, jp093165>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss attributes of entity class.	3.23
Draw user interface of log in	3.27
Draw user interface of registration	3.27
UI of modifying car park information	3.28

<Jiawei LIU, jp093167>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss attributes of entity class.	3.23
Draw user interface of log in	3.27
Draw user interface of registration	3.27
UI of modifying car park information	3.28

Subgroup3:

<Zhijiao LIU, jp092987>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss classes of entity, boundary and control.	3.23
Discuss relationship of classes.	3.23
Discuss attributes of entity class.	3.23
Draw the conceptual class diagram.	3.27
Identify attributes for each entity class.	3.28

<Yang LIU, jp092989>

Tasks	Milestones
Integrate and improve the requirement part.	3.22
Review note about analysis.	3.22
Discuss classes of entity, boundary and control.	3.23
Discuss relationship of classes.	3.23
Discuss attributes of entity class.	3.23
Draw the conceptual class diagram.	3.27
Identify attributes for each entity class.	3.28
Check and correct the operation contracts.	3.28

Week Summary

Accomplishments:

1. Correct the wrong places of last week and discuss all the problems we have met when doing the work of last week.
2. Finish the conceptual class diagram.
3. Identify attributes for each entity class.
4. Do the operation contracts and operation contracts.
5. Draw the sequence diagram for each use cases.

After solving the problems of last week, we distribute all the works of the analysis part. In this part, there are many complex things we have to deal with. When drawing the diagrams, we met some corresponding places of last week, so we have to check the pervious works again.

Weekly report

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Yao PING, jp092985>

Tasks and any key milestones for this week and progress

- Go through notes about prototype user interface.
- Discuss with the group mates and connect use case to user interface.
- Design and draw user interface of log in and register.

Impact of issues and delays on task completion and milestones

- Detail design makes UI more attractive. So we pay more attention to beautifying interface.

Tasks for next week

- Identify attributes of each entity class.

- Consider about attributes for each entity class.
- Add contains for use case.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- Integrate and improve the requirement part
- Review note about analysis
- Discuss classes of entity, boundary and control
- Discuss relationship of classes
- Discuss attributes of entity class
- Write operation contracts of record(ID, enter time and record(collection))
- Draw diagram of class

Impact of issues and delays on task completion and milestones

- We have different mind about class, and define the relationship costs us a lot of time.

Tasks for next week

Complete design part.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Integrate and improve the requirement part.
- Review note about analysis.
- Discuss classes of entity, boundary and control.
- Discuss relationship of classes.
- Discuss attributes of entity class.
- Draw the conceptual class diagram.
- Identify attributes for each entity class.

Impact of issues and delays on task completion and milestones

- We have to plan the whole work and discuss these for a long time to give everyone suitable works.
- When identifying the attributes for each entity class we have to check the codes as well. But there are some problems happens when we doing this.

Tasks for next week

Complete the design step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Integrate and improve the requirement part.
- Review note about analysis.
- Discuss classes of entity, boundary and control.
- Discuss relationship of classes.
- Discuss attributes of entity class.
- Draw the conceptual class diagram.
- Identify attributes for each entity class.

Impact of issues and delays on task completion and milestones

- We have to plan the whole work and discuss these for a long time to give everyone suitable works.
- When identifying the attributes for each entity class we have to check the codes as well. But there are some problems happens when we doing this.

Tasks for next week

Complete the design step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.15 to 3.21>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Integrate and improve the requirement part
- Review note about analysis
- Discuss classes of entity, boundary and control
- Discuss relationship of classes
- Discuss attributes of entity class
- Write operation contracts of get(enter time)
- Draw diagram of class.

Impact of issues and delays on task completion and milestones

We discuss many times with method and get result finally.

Tasks for next week

Complete step of design.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Integrate and improve the requirement part
- Review note about analysis
- Discuss classes of entity, boundary and control
- Discuss relationship of classes
- Discuss attributes of entity class
- Write operation contracts of get(TXT)
- Draw diagram of class.

Impact of issues and delays on task completion and milestones

We draw sequence diagram twice, and modify a lot in the second time.

Tasks for next week

Do design part.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

- Searching for the material about UI.
- Had a group meeting about UI and consider the difficulty of achieve the goal.
- Connect use case to user interface, and design UI.

Impact of issues and delays on task completion and milestones

How to design UI to meet the need of use case was a problem to us. Fortunately, all the

problems were clarified after discussing with my group mates.

Tasks for next week

- Draw a conceptual class diagram.
- Identify attributes of each entity class.
- Add constrains for diagram.
- Draw sequence diagram about enter and payment.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.22 to 3.28>

Group Number: <No.47>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

- Searching information about UI design.
- Discuss with group members and analysis the use case.
- According to the analysis, arrange button distribution.

Impact of issues and delays on task completion and milestones

A correct UI is the key to gain more popularity.

Tasks for next week

- Consider and find boundary, entity and control class.
- Identify class relationship.
- Draw sequence diagram about update and gather report.

Week 5

Weekly Report for: <from 3.29 to 4.4>

Group Number: <No.47>

Group Members:

<Yao PING, jp092985>	<Tianshu ZHANG, jp092990>
<Siyuan LIU, jp092986>	<Tong GE, jp092991>
<Zhijiao LIU, jp092987>	<Chenyu ZHAO, jp093165>
<Yang LIU, jp092989>	<Jiawei LIU, jp093167>

Group Leaders Name: <Zhijiao LIU, jp092987>

Project Progress Summary

GANTT

chart:

ID	Task	Start time	Finish time	Duration	March 29th			April 4th				
					29	30	31	1	2	3	4	
1	Architectural Design	2012/3/29	2012/3/29	1d	■							
2	Communication diagrams for the key operations	2012/3/30	2012/3/30	1d		■						
3	Design a class	2012/4/2	2012/4/2	1d					■			
4	Class Diagram	2012/4/3	2012/4/3	1d							■	

Individual tasks

Subgroup1:

<Tong GE, jp092991>

Tasks	Milestones
Impact of issue Integrate and improve the analysis.	3.29
Review note about design.	3.29
Discus according analysis.	4.3
Identify operation.	4.4

<Tianshu ZHANG, jp092990>

Tasks	Milestones
Integrate and improve the analysis.	3.29
Review note about design.	3.29
Discus according analysis.	4.3
Refine the use case model.	4.4

<Siyuan LIU, jp092986>

Tasks	Milestones
Integrate and improve the analysis.	3.29
Review note about design.	3.29
Discus according analysis.	4.3
Draw class diagram.	4.4

Subgroup2:

<Yao PING, jp092985>

Tasks	Milestones
Complete Architectural Design.	3.29
Draw the communication diagrams	3.29
Correct mistakes for the Analysis part	4.3

<Chenyu ZHAO, jp093165>

Tasks	Milestones
Complete Architectural Design.	3.29
Draw the communication diagrams	3.29
Correct mistakes for the Analysis part	4.3

<Jiawei LIU, jp093167>

Tasks	Milestones
Complete Architectural Design.	3.29
Draw the communication diagrams	3.29
Correct mistakes for the Analysis part	4.3

Subgroup3:

<Zhijiao LIU, jp092987>

Tasks	Milestones
Check all part pervious work.	3.29
Distribute of the tasks of this week.	3.29
Complete the operation part.	4.4

<Yang LIU, jp092989>

Tasks	Milestones
Check all part pervious work.	3.29
Distribute of the tasks of this week.	3.29
Complete the operation part.	4.4

Week Summary

Accomplishments:

- We finished the works according to the schedule and carry on to the next step on time.
- Before carrying on the work of this week we still do some correctness to the pervious works.
- We built the architectural design.
- Draw the communication diagrams for the key operations.
- We design the class and draw the class diagram.

The main work of this step is drawing the communication diagrams and designing the class. Each group member got the suitable job and finished them on time. When meeting problems, we will communicate with each other on time so the problems can be solved on time.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.29 to 4.3>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- Integrate and improve the analysis
- Review note about design
- Discuss according analysis
- Draw class diagram

Impact of issues and delays on task completion and milestones

- When we discuss the design, we find something wrong in analysis, so we improve analysis again.

Tasks for next week

Complete implementation and testing part.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.29 to 4.4>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Check all part pervious work.
- Distribute of the tasks of this week.
- Complete the operation part.

Impact of issues and delays on task completion and milestones

- It's very complex to distribute all the works to the teammates

Tasks for next week

Complete the implementation and testing step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.29 to 4.4>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Correct the mistakes of the last week's work.
- Discuss about this week's task.
- Complete the operation part.

Impact of issues and delays on task completion and milestones

- I spent too much time on correcting the mistakes of last week.

Tasks for next week

Complete the implementation and testing step.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.29 to 4.4>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Integrate and improve the analysis
- Review note about design
- Discuss according analysis
- Refine the use case model

Impact of issues and delays on task completion and milestones

We spend much time discussing some class and then we delete and conflate some of them.

Tasks for next week

Complete step of implementation and testing.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <from 3.29 to 4.4>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Impact of issue Integrate and improve the analysis
- Review note about design
- Discuss according analysis
- Identify operation.

Impact of issues and delays on task completion and milestones

We pay much attention to satisfy the functional requirement.

Tasks for next week

Do implementation and testing part.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.29 to 4.4>

Group Number: <No.49>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

- Complete Architectural Design.
- Draw the communication diagrams for collection, command and create account parts.
- Correct mistakes for the Analysis part.

Impact of issues and delays on task completion and milestones

This week we work together to solve the problems about communication diagrams and find many errors with group members. Without the help of team, we can't deal with these problems individually.

Tasks for next week

Participate in the Implementation part.

EBU5304 Software Engineering – Group Coursework

Weekly Report for:<From 3.29 to 4.4>

Group Number: <No.49>

Individual Member: <Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

- Do the Architectural Design with group members.
- Draw the communication diagrams for get enter time, get state and increase part.
- Correct mistakes for the Analysis part.

Impact of issues and delays on task completion and milestones

Though review of our last step report, we find some key errors and modify them immediately.

Tasks for next week

We will do the implementation work next week.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.29 to 4.4>

Group Number: <No.49>

Individual Member: <Yao PING, jp092985>

Tasks and any key milestones for this week and progress

- Finish Architectural Design work.
- Draw the communication diagrams for modify, record and set part.
- Correct mistakes for the Analysis part.

Impact of issues and delays on task completion and milestones

At first we are confused about whether draw the figure for Nodes and their network configuration or not. With the help of teacher, we solve this problems.

Tasks for next week

Complete the Implementation part next week.

Week 6

Weekly Report for: <from 4.5 to 4.11>

Group Number: <No.47>

Group Members:

<Yao PING, jp092985>	<Tianshu ZHANG, jp092990>
<Siyuan LIU, jp092986>	<Tong GE, jp092991>
<Zhijiao LIU, jp092987>	<Chenyu ZHAO, jp093165>
<Yang LIU, jp092989>	<Jiawei LIU, jp093167>

Group Leaders Name: <Zhijiao LIU, jp092987>

Project Progress Summary

GANTT

chart:

ID	Task	Start time	Finish time	Duration	April 5th			April 11th			
					5	6	7	8	9	10	11
1	Implement a Subsystem	2012/4/6	2012/4/6	1d	■						
2	Perform Unit Tests	2012/4/9	2012/4/9	1d						■	
3	Implement Test	2012/4/10	2012/4/10	1d							■
4	Perform Integration Test	2012/4/10	2012/4/10	1d							■
5	Evaluate Test	2012/4/11	2012/4/11	1d							■

Individual tasks

Subgroup1:

<Tong GE, jp092991>

Tasks	Milestones
Check mistakes and integrate Design part.	4.5
Review notes about Implementation and Test parts.	4.6
Collecting problems from earlier report for next meeting.	4.7
Writing Build 2.0 of Integrate system part.	4.9
Writing component testing level part of Testing plan.	4.10

<Tianshu ZHANG, jp092990>

Task	Milestones
Discuss problems of earlier report.	4.7

Review notes about Implementation and Test parts.	4.8
Writing Build 1.0 of Integrate system part.	4.9
Writing system testing level part of Testing plan.	4.11

<Siyuan LIU, jp092986>

Task	Milestones
Discuss problems of earlier report.	4.7
Review notes about Implementation and Test parts.	4.8
Drawing Architectural Implementation diagrams.	4.9
Checking errors of Testing plan and Architectural Implementation parts.	4.11

Subgroup2:

<Yao PING, jp092985>

Task	Milestone
Discuss problems of earlier reports.	4.7
Review notes about Implementation and Test parts.	4.8
Drawing Architectural Implementation diagram.	4.9
Writing Perform System Test part.	4.11

<Chenyu ZHAO, jp093165>

Task	Milestone
Discuss problems of earlier reports.	4.7
Review notes about Implementation and Test parts.	4.8
Drawing Architectural Implementation diagram.	4.9
Writing Designing Test	4.11

<Jiawei LIU, jp093167>

Task	Milestone
Discuss problems of earlier reports.	4.7
Review notes about Implementation and Test parts.	4.8
Drawing Architectural Implementation diagram.	4.9
Draw Test Matrix of Testing Design part.	4.11

Subgroup3:

<Zhijiao LIU, jp092987>

Task	Milestone
Discuss problems of earlier reports.	4.7
Coding.	4.8
Writing Implement a Class Part.	4.10

<Yang LIU, jp092989>

Task	Milestone
Discuss problems of earlier reports.	4.7

Coding	4.8
Writing Implement a Class Part.	4.10

Week Summary

1. We find out the problems in Design parts and correct them.
2. First half of Implementation and Test are completed.
3. Coding work was still continue.

Weekly report

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.29 to 4.4>

Group Number: <No.49>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

- Complete Architectural Design.
- Draw the communication diagrams for collection, command and create account parts.
- Correct mistakes for the Analysis part.
- Draw Test Matrix of Testing Design part.

Impact of issues and delays on task completion and milestones

This week we work together to solve the problems about communication diagrams and find many errors with group members. Without the help of team, we can't deal with these problems individually.

Tasks for next week

Participate in the Implementation part.

EBU5304 Software Engineering – Group Coursework

Weekly Report for:<From 3.29 to 4.4>

Group Number: <No.49>

Individual Member: <Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

- Do the Architectural Design with group members.
- Draw the communication diagrams for get enter time, get state and increase part.
- Correct mistakes for the Analysis part.
- Writing Designing Test

Impact of issues and delays on task completion and milestones

Though review of our last step report, we find some key errors and modify them immediately.

Tasks for next week

We will do the implementation work next week.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 3.29 to 4.4>

Group Number: <No.49>

Individual Member: <Yao PING, jp092985>

Tasks and any key milestones for this week and progress

- Finish Architectural Design work.
- Draw the communication diagrams for modify, record and set part.
- Correct mistakes for the Analysis part.
- Writing Perform System Test part.

Impact of issues and delays on task completion and milestones

At first we are confused about whether draw the figure for Nodes and their network configuration or not. With the help of teacher, we solve this problems.

Tasks for next week

Complete the Implementation part next week.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.5 to 4.11>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Discuss problems of earlier reports.
- Writing Implement a Class Part.
- Coding
- Read all the assignments of the team works of the last week.
- Distribute the works of this week.

Impact of issues and delays on task completion and milestones

Doing the research of the lectures took too much time.

Tasks for next week

- Complete the testing part and Implementation part.
- Finishing Coding.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.5 to 4.11>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Discuss problems of earlier reports.
- Writing Implement a Class Part.
- Coding
- Integrate the works of the design part.
- Research the staff of the implement part.

Impact of issues and delays on task completion and milestones

Something wrong with the code of last week so we spent much time on it.

Tasks for next week

- Complete the testing part and Implementation part.
- Finishing Coding.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.5 to 4.11>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Check mistakes and integrate Design part.
- Review notes about Implementation and Test parts
- Collecting problems from earlier report for next meeting
- Writing Build 2.0 of Integrate system part
- Writing component testing level part of Testing plan.

Impact of issues and delays on task completion and milestones

We argue with how many systems we have, it cost us a lot of time.

Tasks for next week

Complete Implementation and Test parts.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.5 to 4.11>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Review notes about implementation and testing
- Discuss to decide the subsystem
- According to the class we define in the analysis and design, we make division of labor
- According to the attribute we define in the design, we define the variable.
- Write the code
- Planning a build

Impact of issues and delays on task completion and milestones

It's very complex to distribute all the works to the teammates.

Tasks for next week

Complete Implementation and Test parts.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.5 to 4.11>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- Review notes about implementation and testing
- Discuss to decide the subsystem
- According to the class we define in the analysis and design, we make division of labor
- According to the attribute we define in the design, we define the variable.
- Write the code
- Draw diagram

Impact of issues and delays on task completion and milestones

We spend a little time ensuring the subsystem

Tasks for next week

Complete implement and testing

Week 7

Weekly Report for: <from 4.12 to 4.18>

Group Number: <No.47>

Group Members:

<Yao PING, jp092985>	<Tianshu ZHANG, jp092990>
<Siyuan LIU, jp092986>	<Tong GE, jp092991>
<Zhijiao LIU, jp092987>	<Chenyu ZHAO, jp093165>
<Yang LIU, jp092989>	<Jiawei LIU, jp093167>

Group Leaders Name: <Zhijiao LIU, jp092987>

Project Progress Summary

GANTT chart:

ID	Task	Start time	Finish Time	Duration	April 12th			April 18th				
					12	13	14	15	16	17	18	
1	Architectural Implementation	2012/4/13	2012/4/13	1d	■							
2	Integrate System	2012/4/16	2012/4/16	1d								■
3	Implement a Class	2012/4/17	2012/4/18	2d								■
4	Plan Test	2012/4/18	2012/4/18	1d								■
5	Design Test	2012/4/17	2012/4/18	2d								■
6	Perform System Test	2012/4/12	2012/4/12	1d	■							

Individual tasks

Subgroup1:

<Tong GE, jp092991>

Task	Milestone
Integrate last week reports and collect problems.	4.12
Writing perform unit tests.	4.14
Writing Evaluate test part.	4.15
Integrate the whole report.	4.18

<Tianshu ZHANG, jp092990>

Task	Milestone
Discuss problems found in last week report.	4.12
Participate in perform unit tests work.	4.14
Integrate weekly report.	4.15
Coding.	4.18

<Siyuan LIU, jp092986>

Task	Milestone
Discuss problems found in last week report.	4.12
Participate in perform unit tests work.	4.14
Writing Evaluate test part.	4.15
Coding.	4.18

Subgroup2:

<Yao PING, jp092985>

Task	Milestone
Discuss problems found in last week report.	4.12
Continue doing Perform integration test.	4.14
Coding.	4.15
Integrate weekly reports.	4.18

<Chenyu ZHAO, jp093165>

Task	Milestone
Discuss problems found in last week report.	4.12
Finishing Design Test.	4.14
Test maritx	4.15
Coding	4.18

<Jiawei LIU, jp093167>

Task	Milestone
Discuss problems found in last week report.	4.12
Finishing Design test.	4.14
Review the earlier report and find out problems	4.15
Coding.	4.18

Subgroup3:

<Zhijiao LIU, jp092987>

Task	Milestone
Discuss problems found in last week report.	4.12
Writing Implement Test	4.14
Coding.	4.15
Adjust variable name in report and code.	4.18

<Yang LIU, jp092989>

Task	Milestone
Discuss problems found in last week report.	4.12
Writing Evaluating Test	4.14
Coding	4.15
Adjust variable name in report and code.	4.18

Week summary

1. Accomplishments:
2. Finishing Implementation and test part.
3. Finishing coding and adjust variable name.
4. Get together and review the earlier report to find out problems.

Weekly report**EBU5304 Software Engineering – Group Coursework**

Weekly Report for: <From 4.5 to 4.12>

Group Number: <No.47>

Individual Member: <Jiawei LIU, jp093167>

Tasks and any key milestones for this week and progress

Integrate system.

Implement subsystems for the system.

Get together to talk about problems in coding.

Impact of issues and delays on task completion and milestones

After participate in Coding work,we got our work connected with Code and got well know about the essence of this course work.

Tasks for next week

Start testing the whole system.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.13 to 4.20>

Group Number: <No.47>

Individual Member: <Chenyu ZHAO, jp093165>

Tasks and any key milestones for this week and progress

Testing design for the system.

Test matrix.

Perform integration testing.

Impact of issues and delays on task completion and milestones

At first I was confused about Test matrix,but with the help of our group member,I solve the problems successfully.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.13 to 4.20>

Group Number: <No.47>

Individual Member: <Yao PING, jp092985>

Tasks and any key milestones for this week and progress

Testing design for the system.

Draw the matrix testing.

Perform integration testing.

Impact of issues and delays on task completion and milestones

This week I didn't have any problems because we communicating information frequently and got well known of the whole process.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.12 to 4.18>

Group Number: <No.47>

Individual Member: <Tong GE, jp092991>

Tasks and any key milestones for this week and progress

- Integrate last week reports and collect problems.
- Writing perform unit tests.
- Writing Evaluate test part.
- Integrate the whole report.

Impact of issues and delays on task completion and milestones

Through the testing, we get the final system,so there is a delay.

Tasks for next week

Integrating the whole report.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.12 to 4.18>

Group Number: <No.47>

Individual Member: <Tianshu ZHANG, jp092990>

Tasks and any key milestones for this week and progress

- Integrate last week reports and collect problems.
- Writing perform unit tests.
- Writing Evaluate test part.
- Integrate the whole report.

Impact of issues and delays on task completion and milestones

When doing the testing code, there are some expectations happened. Then delay happened.

Tasks for next week

Integrating the whole report.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.12 to 4.18>

Group Number: <No.47>

Individual Member: <Siyuan LIU, jp092986>

Tasks and any key milestones for this week and progress

- Integrate last week reports and collect problems.
- Participate in perform unit tests.
- Doing Evaluate test part work.
- Integrate the whole report.

Impact of issues and delays on task completion and milestones

Testing must be important in software engineering, so we do many times of testing.

Tasks for next week

Integrating the whole report.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.12 to 4.18>

Group Number: <No.47>

Individual Member: <Zhijiao LIU, jp092987>

Tasks and any key milestones for this week and progress

- Discuss problems found in last week report.
- Writing Implement Test
- Coding.
- Adjust variable name in report and code.

Impact of issues and delays on task completion and milestones

We spent much time on discussing the work distribution.

Tasks for next week

Integrating the whole report.

EBU5304 Software Engineering – Group Coursework

Weekly Report for: <From 4.12 to 4.18>

Group Number: <No.47>

Individual Member: <Yang LIU, jp092989>

Tasks and any key milestones for this week and progress

- Discuss problems found in last week report
- Writing Evaluating Test
- Coding
- Adjust variable name in report and code.

Impact of issues and delays on task completion and milestones

When doing the testing code, there are some expectations happened.

Tasks for next week

Integrating the whole report.

Appendix 2 – JAVA Code

Enter_UI.java

```
/**
 * @author Zhijiao Liu
 *
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.EmptyBorder;

public class Enter_UI extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    JTextField txtReadCard;
    JButton btnNewButton;
    JTextArea textArea;

    protected String IDInput;
```

payment_UI.java

```
/**
 * @author Zhijiao Liu
 *
 */

import javax.swing.*;

import java.awt.*;

import javax.swing.border.EmptyBorder;

public class Payment_UI extends JFrame {

    /**
     *
     */

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    public JPanel panel ;
    public JPanel panel_2;
```

```
public JTextField textField_1;
public JTextField textField_2;
public JTextArea textArea;

/**
 * Launch the application.
 */
/**
 * Create the frame.
 */
public Payment_UI() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(450, 400, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.NORTH);

    JLabel lblPaymentStation = new JLabel("Payment Station");
    panel.add(lblPaymentStation);
    lblPaymentStation.setFont(new Font("Times New Roman", Font.PLAIN,
28));

    JPanel panel_1 = new JPanel();
    contentPane.add(panel_1, BorderLayout.CENTER);
    panel_1.setLayout(new GridLayout(0, 2, 0, 0));

    panel_2 = new JPanel();
    panel_1.add(panel_2);
    panel_2.setLayout(new BorderLayout(0, 0));

    JPanel panel_4 = new JPanel();
    panel_2.add(panel_4, BorderLayout.NORTH);

    JPanel panel_5 = new JPanel();
    panel_2.add(panel_5, BorderLayout.SOUTH);

    JPanel panel_6 = new JPanel();
    panel_2.add(panel_6, BorderLayout.WEST);

    JPanel panel_7 = new JPanel();
```

```
panel_2.add(panel_7, BorderLayout.EAST);

textArea = new JTextArea();
textArea.setEditable(false);
textArea.setText("Welcome to our car park!\nHere is the payment station");
panel_2.add(textArea, BorderLayout.CENTER);

JPanel panel_3 = new JPanel();
panel_1.add(panel_3);
panel_3.setLayout(new GridLayout(2, 1, 0, 0));

JPanel panel_8 = new JPanel();
panel_3.add(panel_8);
panel_8.setLayout(new BorderLayout(0, 0));

textField_1 = new JTextField();
textField_1.setHorizontalAlignment(SwingConstants.CENTER);
textField_1.setForeground(Color.WHITE);
textField_1.setBackground(Color.BLACK);
panel_8.add(textField_1, BorderLayout.CENTER);
textField_1.setColumns(10);

JPanel panel_10 = new JPanel();
panel_8.add(panel_10, BorderLayout.NORTH);

JPanel panel_11 = new JPanel();
panel_8.add(panel_11, BorderLayout.SOUTH);

JPanel panel_12 = new JPanel();
panel_8.add(panel_12, BorderLayout.WEST);

JPanel panel_13 = new JPanel();
panel_8.add(panel_13, BorderLayout.EAST);

JPanel panel_9 = new JPanel();
panel_3.add(panel_9);
panel_9.setLayout(new GridLayout(1, 2, 0, 0));

JTextArea txtrPleaseScanYour = new JTextArea();
txtrPleaseScanYour.setEditable(false);

txtrPleaseScanYour.setBackground(UIManager.getColor("Button.background"));
txtrPleaseScanYour.setFont(new Font("Times New Roman", Font.PLAIN,
14));
```

```
        txtrPleaseScanYour.setText("Please scan \r\nyour card/ticket \r\nfirst, then  
finish \r\nthe payment.");  
        panel_9.add(txtrPleaseScanYour);  
  
        JPanel panel_14 = new JPanel();  
        panel_9.add(panel_14);  
        panel_14.setLayout(new GridLayout(0, 3, 5, 10));  
  
        JPanel panel_15 = new JPanel();  
        panel_14.add(panel_15);  
  
        JPanel panel_16 = new JPanel();  
        panel_14.add(panel_16);  
        panel_16.setLayout(new BorderLayout(0, 0));  
  
        textField_2 = new JTextField();  
        panel_16.add(textField_2, BorderLayout.EAST);  
        textField_2.setHorizontalAlignment(SwingConstants.CENTER);  
        textField_2.setForeground(Color.WHITE);  
        textField_2.setBackground(Color.BLACK);  
        textField_2.setColumns(4);  
  
        JPanel panel_17 = new JPanel();  
        panel_16.add(panel_17, BorderLayout.NORTH);  
  
        JPanel panel_18 = new JPanel();  
        panel_16.add(panel_18, BorderLayout.SOUTH);  
    }  
  
}
```

Exit_UI.java

```
/**
```

```
 * @author Siyuan Liu
```

```
 *
```

```
 */
```

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
import javax.swing.border.EmptyBorder;
```

```
public class Exit_UI extends JFrame {
```

```
/**
 *
 */
private static final long serialVersionUID = 1L;
private JPanel contentPane;
JTextField textField_1;
JTextArea txtrIdSParking;

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Exit_UI frame = new Exit_UI();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */
public Exit_UI() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(900, 400, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.NORTH);

    JLabel lblThankYou = new JLabel("Thank you!");
    panel.add(lblThankYou);
    lblThankYou.setFont(new Font("Times New Roman", Font.PLAIN, 28));

    JPanel panel_1 = new JPanel();
```

```
contentPane.add(panel_1, BorderLayout.SOUTH);

JLabel lblPleaseScanThe = new JLabel("Please scan the card or insert the
ticket.");
panel_1.add(lblPleaseScanThe);
lblPleaseScanThe.setFont(new Font("Times New Roman", Font.PLAIN,
17));

JPanel panel_2 = new JPanel();
contentPane.add(panel_2, BorderLayout.EAST);
panel_2.setLayout(new GridLayout(5, 1, 10, 3));

JPanel panel_3 = new JPanel();
panel_2.add(panel_3);
panel_3.setLayout(new GridLayout(4, 1, 0, 0));

JPanel panel_4 = new JPanel();
panel_2.add(panel_4);

JLabel lblInsertTicket = new JLabel("Insert Ticket");
panel_4.add(lblInsertTicket);

textField_1 = new JTextField();
textField_1.setHorizontalAlignment(SwingConstants.CENTER);
textField_1.setBackground(Color.BLACK);
textField_1.setForeground(Color.WHITE);
panel_2.add(textField_1);
textField_1.setColumns(10);

JPanel panel_5 = new JPanel();
contentPane.add(panel_5, BorderLayout.CENTER);
panel_5.setLayout(new BorderLayout(10, 10));

JPanel panel_6 = new JPanel();
panel_5.add(panel_6, BorderLayout.SOUTH);

JPanel panel_7 = new JPanel();
panel_5.add(panel_7, BorderLayout.WEST);

JPanel panel_8 = new JPanel();
panel_5.add(panel_8, BorderLayout.NORTH);

JPanel panel_9 = new JPanel();
panel_5.add(panel_9, BorderLayout.EAST);
```



```
        txtrIdSParking = new JTextArea();
        txtrIdSParking.setText("Welcome to our car park!\nHere is the exit.");
        txtrIdSParking.setEditable(false);
        panel_5.add(txtrIdSParking, BorderLayout.CENTER);
    }
}
```

```
}
```

Driver_Control.java

```
/**
```

```
 * @author Yang Liu
```

```
 *
```

```
 */
```

```
import java.io.FileNotFoundException;
import java.io.FileWriter ;
import java.io.BufferedWriter ;
import java.io.IOException;
import java.io.FileReader ;
import java.io.BufferedReader;
import java.math.BigDecimal;
import java.math.RoundingMode;
```

```
import java.awt.EventQueue;
```

```
import javax.swing.JFrame;
```

```
import javax.swing.JOptionPane;
```

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

```
import java.util.ArrayList;
import java.util.Date;
import java.util.LinkedList;
```

```
public class Driver_Control extends JFrame implements ActionListener {
    /**
```

```

    *
    */
private static final long serialVersionUID = 1L;
Enter_UI enterFrame;
Payment_UI payFrame;
Exit_UI exitFrame ;
public Car_Park park;

String IDInput;
String payInput;
String leaveInput;
String collectionInput;
String AccountInput;
Usage transhtemp=new Usage();
int accountNum;
int key=0;
Public_Account transpaytemp;

int flag=0;
int count=0;
int size=0; //mylist3 size
int ticketcount=0;

LinkedList<Usage> mylist1=new LinkedList<Usage>();//Stack1, is for usage
LinkedList<Usage> mylist2=new LinkedList<Usage>();//Stack2, is for account

ArrayList<Staff_Account> mylist3=new ArrayList<Staff_Account>();
//Stack1, is for usage
ArrayList<Public_Account> mylist4=new ArrayList<Public_Account>();
//Stack2, is for account

FileWriter fw;
BufferedWriter bw;
FileReader fr;
BufferedReader br;

public Driver_Control(){
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                /*
                 * mylist4 is for the public user;

```

```

        * mylist3 is for the QM staff
        */
        park=new Car_Park();
        enter();

        record();

        fr=new FileReader("StaffDB.txt");
        br=new BufferedReader(fr);
        enterFrame = new Enter_UI();
        enterFrame.setVisible(true);

        enterFrame.btnNewButton.addActionListener(new
ActionListener(){
            public void actionPerformed(ActionEvent evt) {
                Date today=new Date();

                boolean b=checkDay(today);
                System.out.println("Today? "+b);
                if(b)
                {

                    if(ticketcount==Car_Park.amount_of_tickets)
                        {JOptionPane.showMessageDialog(null, "Sorry,
there is no ticket.", "SORRY", JOptionPane.ERROR_MESSAGE);
                        choose.addTicket=1;
                        if(choose.addTicket==1)
                        {

                            Operator_UI.frame0.lblOperator.setText("Attention: The ticket is used up, please
fill in the ticket box.");

                                JOptionPane.showMessageDialog(null, "Sorry,
There is no ticket. Operator fills in the ticket box.", "SORRY",
JOptionPane.INFORMATION_MESSAGE);
                                Car_Park.amount_of_tickets=10;
                                ticketcount=0;
                            }
                            System.out.println("You can't get
in."+choose.addTicket);
                        }
                    }
                {
                    Public_Account pubTemp= mylist4.get(ticketcount);

```

```

        while(pubTemp.payflag!=0)
        {   pubTemp= mylist4.get(ticketcount);
            ticketcount++;
        }   //used to test the situation after add ticket,
ticket count back to 0

        pubTemp.card_number=""+ticketcount;
        pubTemp.payflag=1;
        JOptionPane.showMessageDialog(null, "Welcome,
your ticket number is "+pubTemp.card_number, "SUCCESS",
JOptionPane.INFORMATION_MESSAGE);

        System.out.println("Welcome"+pubTemp.accountNum);

        Usage temp=new Usage();
        temp.accountlistnum=pubTemp.accountNum;
        temp.type=false;
        temp.id=""+ticketcount;
        temp.enter_time=new Date();
        mylist1.add(temp);
        ticketcount++;
        count++;

        Car_Park.remaining_ticket=Car_Park.amount_of_tickets-ticketcount;

        Car_Park.remaining_space=Car_Park.parking_space-count;

        enterFrame.textArea.setText(park.refreshEnter());
    } //end if

    }
    else
        JOptionPane.showMessageDialog(null, "Public
can't step in today.", "FAILURE", JOptionPane.ERROR_MESSAGE);
    }

});

/*listen to the enterFrame's text field to process the entrance of
the staff*/

```

```
enterFrame.txtReadCard.addActionListener(new
ActionListener(){
    public void actionPerformed(ActionEvent evt) {

        IDInput =
enterFrame.txtReadCard.getText().toString();
        AccountInput="ID: "+IDInput;

        try{
            FileReader fr=new FileReader("registerlist.txt");
            BufferedReader br=new BufferedReader(fr);
            String line=br.readLine();

            int check=0;
            while (line!=null)
            {
                check++;
                if(check>size)
                {
                    Staff_Account tempAccount=new Staff_Account(line);
                    mylist3.add(tempAccount);
                }

                line=br.readLine();
            }
            size=mylist3.size();

            System.out.println("Now the size is "+size);
            Staff_Account a=mylist3.get(1);
            System.out.println("The name is "+a.card_number);

            br.close();
            fr.close();
        }
        catch(IOException e)
        {
            System.out.println("Cannot find the file.");
        }

        int j;
        for ( j=0;j<size;j++)
```

```
        {
            Staff_Account b=mylist3.get(j);
            if(b.card_number.equals(AccountInput))
                break;
        }
        if(j==size)
        {
            System.out.println("You can't enter.");
            JOptionPane.showMessageDialog(null, "You can't
enter, you haven't registered.", "FAILURE", JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            Staff_Account findaccount=mylist3.get(j);
            if(findaccount.payflag==1)
                JOptionPane.showMessageDialog(null, "You
can't enter, your last car usage had a problem.", "FAILURE",
JOptionPane.ERROR_MESSAGE);
            else{
                findaccount.payflag=1;
                System.out.println("It
is"+findaccount.card_number);

                Usage temp=new Usage(IDInput);
                temp.type=true;
                temp.accountlistnum=j;

                temp.enter_time=new Date();
                mylist1.add(temp);
                count++;

                Car_Park.remaining_space=Car_Park.parking_space-count;

                JOptionPane.showMessageDialog(null,
"Welcome"+findaccount.card_number, "SUCCESS",
JOptionPane.INFORMATION_MESSAGE);

                enterFrame.textArea.setText(park.refreshEnter());
            }
        }
    });
```

```

payFrame = new Payment_UI();
payFrame.setVisible(true);

payFrame.textField_1.addActionListener(new ActionListener(){
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent evt) {

        payInput=payFrame.textField_1.getText();
        Usage temp,htemp=new Usage();
        if(flag==0)    //whether the car is in car park
        {
            while(!empty(mylist1))

                {   if(mylist1.size()!=0)
                    {       temp=(Usage)mylist1.getLast();

                            mylist1.removeLast();
                            if( payInput.equals(temp.id))
                                { count--;
                                  htemp=temp;
                                  flag=1;

                                }
                            else mylist2.add(temp);
                    }
                }
            while(!empty(mylist2))
                {temp=(Usage)mylist2.getLast();
                  mylist2.removeLast();
                  mylist1.add(temp);
                }

            if(flag==0)    //no such a car in car park
                {JOptionPane.showMessageDialog(null,    "Sorry,
there is no such a car.", "SORRY", JOptionPane.ERROR_MESSAGE);
                payFrame.textField_1.setText("There is no such a
car");}

            else    //find the info of the car
            {

                flag=0;
                double s=0;

```

```
        if (htemp.type==true)
        {
            int year1=0;int month1=0;int day1=0;
            Date tempory=new Date();
            int year=tempory.getYear();
            int month=tempory.getMonth();
            int day=tempory.getDate();

if( mylist3.get(htemp.accountlistnum).accountDate!=null)
        {
            System.out.println("Calculate the days of
parking.");
            year1=
mylist3.get(htemp.accountlistnum).accountDate.getYear();
            month1=
mylist3.get(htemp.accountlistnum).accountDate.getMonth();
            day1=
mylist3.get(htemp.accountlistnum).accountDate.getDate();
            htemp.payment_time=tempory;

if(year==year1&&month==month1&&day==day1)
            s=0;
            else
            {
                if(month-month1==0)
                {
                    int dayin=day-day1;
                    s=1*dayin;
                }else{

JOptionPane.showMessageDialog(null, "Sorry, your last month bill has problems,
\nplease ask operator for help.", "SORRY", JOptionPane.ERROR_MESSAGE);
                }
            }
        }

    }
else
    {htemp.payment_time=tempory;

    s=1;
    }
}
```



```

    }
    if (htemp.type==false)
    {
        htemp.payment_time=new Date();
        s=countcost(htemp);
    }
    htemp.charge=s;
    if(htemp.type==true)
    {
        Staff_Account
findaccount=mylist3.get(htemp.accountlistnum);

findaccount.accountCharge=findaccount.accountCharge+s;
        System.out.println("Your account charge is
"+findaccount.accountCharge);
        findaccount.payflag=0;
    }

    try{
        fw = new
FileWriter("usage_information.txt", true);
        bw = new BufferedWriter(fw);
        bw.newLine();

bw.write("\t"+htemp.id+"\t"+htemp.enter_time+"\t"+htemp.payment_time+"\t"+htem
p.charge);

        bw.close();

    }
    catch(IOException e)
    {
        System.out.println("Cannot find the file.");
    }

    if(htemp.type==true)
        payFrame.textArea.setText("Successfully
out.\n This time your expense is "+htemp.charge+".\n" +
        "Your account charge is
"+mylist3.get(htemp.accountlistnum).accountCharge);
        Date lastTime=new Date();

mylist3.get(htemp.accountlistnum).accountDate=lastTime;

```

```

payFrame.textArea.append(park.refreshPayment());
                    if(htemp.type==false)
                    {
                        payFrame.textArea.setText("Please pay
your fee.\nYou need to pay "+(float)htemp.charge);

payFrame.textArea.append(park.refreshPayment());
                    Public_Account
paytemp=mylist4.get(htemp.accountlistnum);

////////////////////////////////////

                    System.out.println("Now you have to pay
the fee before you leave the car park.");

                    accountNum=htemp.accountlistnum;

                    htemp.collection=0;
                    transhtemp=htemp;
                    System.out.println("Usage:
"+transhtemp.id+transhtemp.collection+"\n");
                    payFrame.textField_2.setEditable(true);

payFrame.textField_2.addActionListener(new ActionListener(){

                    public void
actionPerformed(ActionEvent evt) {
                        Public_Account
transpaytemp=mylist4.get(accountNum);

                        if(transpaytemp.payflag!=0)
                        {
                            System.out.println("Account:
"+transpaytemp.card_number+transpaytemp.payflag+"\n");;
                            System.out.println("the
collection1 is "+transhtemp.collection);

                            collectionInput=payFrame.textField_2.getText();

```

```

    payFrame.textField_2.setText("");
                                                                    float
money=Float.parseFloat(collectionInput);
                                                                    System.out.println("the
collection2 is "+transhtemp.collection);

    transhtemp.collection=transhtemp.collection+money;
                                                                    System.out.println("the
collection3 is "+transhtemp.collection);
                                                                    if
(transhtemp.collection>=transhtemp.charge)
                                                                    {
                                                                    transpaytemp.payflag=0;
                                                                    BigDecimal a = new
                                                                    BigDecimal(transhtemp.collection);
                                                                    BigDecimal paidMoney
= a.setScale(2, RoundingMode.DOWN);

    JOptionPane.showMessageDialog(null, "You have paid \u00A3 "+paidMoney,
"Welcome", JOptionPane.INFORMATION_MESSAGE);

    Car_Park.amount_of_coins=Car_Park.amount_of_coins+transhtemp.collection;
                                                                    System.out.println("The
park max is "+Car_Park.max_money);
                                                                    System.out.println("The
park coin is "+Car_Park.amount_of_coins);
                                                                    if
(Car_Park.amount_of_coins>=Car_Park.max_money)

    {JOptionPane.showMessageDialog(null, "The money box is full, you have to
collect the money.", "Operator", JOptionPane.INFORMATION_MESSAGE);

    JOptionPane.showMessageDialog(null, "The money (\u00A3
"+Car_Park.amount_of_coins+" is taken away.", "Operator",
JOptionPane.INFORMATION_MESSAGE);

    Car_Park.amount_of_coins=0;}

    payFrame.textField_2.setEditable(false);

```

```

                                transhtemp.collection=0;
                                }
                                }

                                System.out.println("In the end,
account"+transpaytemp.card_number+"flag number is"+transpaytemp.payflag+"\n");

                                }
                                });
                                }

                                generate_bill();
                                }
                                }
                                });

```

```

exitFrame = new Exit_UI();
exitFrame.setVisible(true);
exitFrame.textField_1.addActionListener(new
ActionListener(){

                                public void actionPerformed(ActionEvent evt) {

                                        leaveInput =
exitFrame.textField_1.getText().toString();
                                        int length=mylist4.size();
                                        int i;
                                        for(i=0;i<length;i++)
                                        {
                                                if(leaveInput.equals(mylist4.get(i).card_number))
                                                break;
                                        }
                                        if(i!=length)
                                        {
                                                Public_Account exit=mylist4.get(i);

```

```
        if(exit.payflag==0)
            exitFrame.txtrIdSParking.setText("Thank you
for using our car park.");
        if(exit.payflag==1)
            exitFrame.txtrIdSParking.setText("You have to
pay before exit");
    }
    else
    {
        int length1=mylist3.size();
        int j;
        leaveInput="ID: "+leaveInput;
        for(j=0;j<length1;j++)
        {
            if(leaveInput.equals(mylist3.get(j).card_number))
                break;
        }
        if(j!=length1)
        {
            Staff_Account exit=mylist3.get(j);
            if(exit.payflag==0)
            {
                exitFrame.txtrIdSParking.setText("QM: "+mylist3.get(j).card_number+".\nNow your
account charge is "+mylist3.get(j).accountCharge+"\nPlease pay it off at the end of
the month.\nWelcome to you again.");
            }
            if(exit.payflag==1)
                exitFrame.txtrIdSParking.setText("You have to RECORD your expense in your
account");
        }
        else
            exitFrame.txtrIdSParking.setText("There
is something wrong in your account, \nyou are not in the car park.");
    }
};
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

}

public void enter() throws IOException{
    FileReader fr = null;
    try {
        fr = new FileReader("registerlist.txt");
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    BufferedReader br=new BufferedReader(fr);
    String line = null;
    try {
        line = br.readLine();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    flag=0;
    int checktype=0;
    while (line!=null)
    {   checktype++;
        if(checktype<=Car_Park.amount_of_tickets)
        {
            Public_Account tempAccount=new Public_Account(line);
            mylist4.add(tempAccount);
        }

        else{
            Staff_Account tempAccount=new Staff_Account(line);
            mylist3.add(tempAccount);
        }

        try {
            line=br.readLine();
```

```
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    size=mylist3.size();

    System.out.println("the size3 is "+mylist3.size());
    System.out.println("the size4 is "+mylist4.size());
    Staff_Account a=mylist3.get(1);
    System.out.println("the name is"+a.card_number);

    br.close();
    fr.close();
}

public void record() throws IOException{
    fw=new FileWriter("usage_information.txt");
    bw=new BufferedWriter(fw);
    bw.write("ID number\t\tEnter time\t\tpayment time\t\t\tcharge");
    bw.close();
}

public double countcost(Usage temp ) //calculate the charge
{
    int
s1=temp.payment_time.getSeconds()+temp.payment_time.getMinutes()*60+temp.pay
ment_time.getHours()*3600+temp.payment_time.getDay()*216000;
    int
s2=temp.enter_time.getSeconds()+temp.enter_time.getMinutes()*60+temp.enter_time
.getHours()*3600+temp.enter_time.getDay()*216000;
    double s=0;

    {
        int s3=s1-s2;

        if(s3>0&& s3<2*3600)
            s=Car_Park.price1;
        else if(s3>=2*3600&& s3<4*3600)
            s=Car_Park.price2;
        else if(s3>=4*3600&& s3<8*3600)
            s=Car_Park.price3;
        else if(s3>=8*3600&& s3<12*3600)
```

```
        s=Car_Park.price4;
    else if(s3>=12*3600&&3<24*3600)
        s=Car_Park.price5;
    else
        s=Car_Park.price5;
    }

    return s;
}

/*It is a method to generate immediately bill*/
public void generate_bill(){
    try{
        fw=new FileWriter("bill.txt");
        bw=new BufferedWriter(fw);
        bw.write("ID card number\t\t\taccount Charge");
        bw.close();
        FileWriter fw=new FileWriter("bill.txt",true);
        BufferedWriter bw=new BufferedWriter(fw);
        for(int i=0;i<size;i++)
        {
            Staff_Account billaccount=mylist3.get(i);
            bw.newLine();

            bw.write(billaccount.card_number+"\t\t\t"+billaccount.accountCharge);

        }

        bw.close();

    }
    catch(IOException ev)
    {
        System.out.println("Cannot find the file!");
    }
}

@SuppressWarnings("deprecation")
public boolean checkDay(Date date){
    int judge=0;

    judge=Car_Park.y1*Car_Park.y2*Car_Park.m1*Car_Park.m2*Car_Park.d1*Car
_Park.d2;
    System.out.println("!!!!!!! Result of check day: "+judge);
}
```



```
int i=date.getDay();
if(i==6)
{
    return true;
}
if(i==0)
{
    return true;
}

if(judge!=0)
{
    System.out.println("!!!!!!!");
    Car_Park.date1.setYear(Car_Park.y1);
    Car_Park.date2.setYear(Car_Park.y2);
    Car_Park.date1.setMonth(Car_Park.m1);
    Car_Park.date2.setMonth(Car_Park.m2);
    Car_Park.date1.setDate(Car_Park.d1);
    Car_Park.date2.setDate(Car_Park.d2);

    Car_Park.date1.setHours(0);
    Car_Park.date2.setHours(23);
    Car_Park.date1.setMinutes(0);
    Car_Park.date2.setMinutes(59);
    Car_Park.date1.setSeconds(0);
    Car_Park.date2.setSeconds(59);

    System.out.println(date);
    System.out.println(Car_Park.date1);
    System.out.println(Car_Park.date2);
    if(date.after(Car_Park.date1)&&date.before(Car_Park.date2))
        return true;
}

return false;
}

public void push(Usage usage,LinkedList<Usage> mylist)//压栈
```

```
{
    mylist.add(usage);
}
public Usage pop(LinkedList<Usage> mylist) //弹栈
{
    Usage temp=new Usage();
    if(mylist.size()!=0) {temp=(Usage)mylist.getLast();}
mylist.removeLast();return (Usage)temp;}
    else return null;
}
public boolean empty(LinkedList<Usage> mylist)//是否为空
{
    if(mylist.size()==0) return true;
    else return false;
}

public static boolean isInteger(String value) {
    try {
        Integer.parseInt(value);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

@Override
public void actionPerformed(ActionEvent arg0) {
    // TODO Auto-generated method stub

}
}
```

Account.java

```
/**
 * @author Siyuan Liu
 *
 */
class Account {

    protected String card_number; //identify the ID of account
    protected int payflag; //decide the payment state

}
```

Staff_Account.java

```
import java.util.Date;

/**
 * @author Tianshu Zhang
 *
 */

public class Staff_Account extends Account {

    public double accountCharge; //define the charge of each account
    public Date accountDate; //record the date

    Staff_Account(String id){
        this.card_number=id;
        this.payflag=0;
        System.out.println("Staff Card Number: "+this.card_number+" Staff
Payment State: "+this.payflag);
    }

}

```

Public_Account.java

```
/**
 * @author Zhijiao Liu
 *
 */

public class Public_Account extends Account {

    public int accountNum; //provide the predetermined id of public account

    Public_Account(String id){
        this.card_number=""+id;
        this.accountNum=Integer.parseInt(id)-1;
        this.payflag=0;
        System.out.println("Public Card Number: "+this.card_number+" Public
Account Number: "+this.accountNum+" Public Payment State: "+this.payflag);
    }
}

```

```
}
```

```
}
```

Operator_UI.java

```
/**
```

```
 * @author Zhijiao Liu
```

```
 *
```

```
*/
```

```
import java.util.ArrayList;
```

```
import java.util.Date;
```

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import java.io.BufferedReader;
```

```
import java.io.BufferedWriter;
```

```
import java.io.FileNotFoundException;
```

```
import java.io.FileReader;
```

```
import java.io.FileWriter;
```

```
import java.io.IOException;
```

```
import java.util.Arrays;
```

```
import javax.swing.border.EmptyBorder;
```

```
import javax.swing.event.AncestorListener;
```

```
import javax.swing.event.AncestorEvent;
```

```
public class Operator_UI extends JFrame {
```

```
    /**
```

```
     *
```

```
    */
```

```
    private static final long serialVersionUID = 1L;
```

```
    Car_Park park;
```

```
    private JPanel contentPane;
```

```
    private JTextField textField;
```

```
    private JPasswordField passwordField;
```

```
    private JPanel panel_1_0;
```

```
    private JPanel panel_1_5;
```

```
    private JPanel panel_1_1;
```

```
    private JPanel panel_1_3;
```

```
    private JPanel panel_1_4;
```

```
    private JPanel panel_1_2;
```

```
private JButton btnLogin;
int ID1 = 1;
int PW1 = 1;
public String ID;
static choose frame0;

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Operator_UI frame = new Operator_UI();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */
public Operator_UI() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(450, 100, 450, 300);
    contentPane = new JPanel();
    setContentPane(contentPane);
    contentPane.setLayout(new GridLayout(6, 1, 0, 0));

    panel_1_0 = new JPanel();
    contentPane.add(panel_1_0);

    panel_1_1 = new JPanel();
    contentPane.add(panel_1_1);
    panel_1_1.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

    JLabel lblCarParkSystem = new JLabel("Car Park Control System");
    panel_1_1.add(lblCarParkSystem);
    lblCarParkSystem.setFont(new Font("Times New Roman", Font.PLAIN,
28));
```

```
panel_1_2 = new JPanel();
contentPane.add(panel_1_2);
panel_1_2.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

JLabel lblId = new JLabel("          ID:");
panel_1_2.add(lblId);
lblId.setHorizontalAlignment(SwingConstants.RIGHT);
lblId.setFont(new Font("Times New Roman", Font.PLAIN, 20));

textField = new JTextField();
panel_1_2.add(textField);
textField.setHorizontalAlignment(SwingConstants.LEFT);
textField.setColumns(10);

panel_1_3 = new JPanel();
contentPane.add(panel_1_3);

JLabel lblPassword = new JLabel("Password:");
panel_1_3.add(lblPassword);
lblPassword.setFont(new Font("Times New Roman", Font.PLAIN, 20));

        passwordField = new JPasswordField();
        panel_1_3.add(passwordField);
        passwordField.setColumns(10);
        passwordField.setHorizontalAlignment(SwingConstants.LEFT);

panel_1_4 = new JPanel();
contentPane.add(panel_1_4);

btnLogin = new JButton("Login");
panel_1_4.add(btnLogin);

panel_1_5 = new JPanel();
contentPane.add(panel_1_5);
login();

}

public void login(){
```

```
btnLogin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ID = textField.getText();
        choose.ID=ID;
        RegisterNewCar.ID=ID;
        RegisterNewCar_1.ID=ID;
        ModifyParkInfo.ID=ID;
        char[] PW = passwordField.getPassword();
        String Password = new String(PW);
        try {
            FileReader fr=new FileReader("OperatorList.txt");
            BufferedReader br=new BufferedReader(fr);
            String num=br.readLine();
            int numOfOP=Integer.parseInt(num);
            while(numOfOP!=0){
                String line1=br.readLine();
                String line2=br.readLine();
                System.out.println("line1 "+line1);
                System.out.println("line2 "+line2);
                System.out.println("ID "+ID);
                System.out.println("Password "+Password);

                if(ID.equals(line1) && Password.equals(line2)){
                    ID1 = 1;
                    PW1 = 1;
                    break;
                }
            }
            else{
                ID1 = 0;
                PW1 = 0;
            }
            numOfOP--;
        }

        if((ID1 == 1) && (PW1 == 1)){
            setVisible(false);
            frame0=new choose();
            frame0.setVisible(true);
            frame0.lblOperator.setText("Operator: "+ID);

        }
        else{
            JOptionPane.showMessageDialog(null, "Sorry, you entered
a wrong ID or Password.", "SORRY", JOptionPane.ERROR_MESSAGE);
```

```
        }
    } catch (FileNotFoundException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    } catch (IOException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }
}

});

}

}
```

```
class choose extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane1;
    RegisterNewCar_1 frame1;
    ModifyParkInfo initial;
    JLabel lblOperator;
    Date today;
    static String ID;
    static int addTicket=0;

    /**
     * Launch the application.
     */

    /**
```



```
* Create the frame.
*/
public choose() {

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(450, 100, 450, 300);
    contentPane1 = new JPanel();
    contentPane1.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane1);
    contentPane1.setLayout(new GridLayout(5, 0, 0, 0));

    JPanel panel_4 = new JPanel();
    FlowLayout fl_panel_4 = (FlowLayout) panel_4.getLayout();
    fl_panel_4.setAlignment(FlowLayout.LEFT);
    contentPane1.add(panel_4);

    lblOperator = new JLabel("Operator:");
    lblOperator.setHorizontalAlignment(SwingConstants.CENTER);
    panel_4.add(lblOperator);

    JPanel panel_3 = new JPanel();
    contentPane1.add(panel_3);

    JLabel label = new JLabel("Car Park Control System");
    label.setFont(new Font("Times New Roman", Font.PLAIN, 28));
    panel_3.add(label);

    JPanel panel_2 = new JPanel();
    contentPane1.add(panel_2);

    JButton btnRegisterNewCar = new JButton("Register New Car");
    panel_2.add(btnRegisterNewCar);

    btnRegisterNewCar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            frame1=new RegisterNewCar_1();
            frame1.setVisible(true);
        }
    });
}
```

```
JPanel panel_1 = new JPanel();
contentPane1.add(panel_1);

JButton btnModify = new JButton("Modify Park Info");
panel_1.add(btnModify);
btnModify.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        setVisible(false);
        initial=new ModifyParkInfo();
        initial.setVisible(true);
    }
});

JPanel panel = new JPanel();
FlowLayout flowLayout = (FlowLayout) panel.getLayout();
flowLayout.setAlignment(FlowLayout.RIGHT);
contentPane1.add(panel);

JButton btnLogout = new JButton("Logout");
panel.add(btnLogout);
btnLogout.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        setVisible(false);
        new Operator_UI().setVisible(true);
    }
});
}
}
```

```
class RegisterNewCar extends JFrame {
```

```
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    public JLabel lblIdXXXXXXXX;
    public JLabel lblNameXXXXX;
    public JLabel lblEmailXXXXXXXXXXcom;
    public static String ID;
    String registerInput;
    JButton btnRegister;
```

```
int flag;//indicate whether it is registered.
    //If it already register,set flag=1.if not,flag=0

/**
 * Launch the application.
 */

    public void run() {
        try {
            RegisterNewCar frame = new RegisterNewCar();
            frame.setVisible(false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

/**
 * Create the frame.
 */
public RegisterNewCar() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(450, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.EAST);
    panel.setLayout(new GridLayout(6, 1, 0, 0));

    JPanel panel_2 = new JPanel();
    panel.add(panel_2);

    JButton btnHome = new JButton("    Home    ");
    panel_2.add(btnHome);
    btnHome.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            choose frame0=new choose();
            frame0.setVisible(true);
            frame0.lblOperator.setText("Operator: "+ID);
        }
    });
}
```

```
    }
  });

  JPanel panel_3 = new JPanel();
  panel.add(panel_3);

  JButton btnModifyParkInfo = new JButton("Modify Park Info");
  panel_3.add(btnModifyParkInfo);
  btnModifyParkInfo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      setVisible(false);
      new ModifyParkInfo().setVisible(true);
    }
  });

  JPanel panel_4 = new JPanel();
  panel.add(panel_4);

  JPanel panel_5 = new JPanel();
  panel.add(panel_5);

  JPanel panel_6 = new JPanel();
  panel.add(panel_6);

  JPanel panel_7 = new JPanel();
  FlowLayout flowLayout = (FlowLayout) panel_7.getLayout();
  flowLayout.setAlignment(FlowLayout.RIGHT);
  panel.add(panel_7);

  JButton btnLogout = new JButton("Logout");
  panel_7.add(btnLogout);
  btnLogout.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      setVisible(false);
      new Operator_UI().setVisible(true);
    }
  });

  JPanel panel_1 = new JPanel();
  contentPane.add(panel_1, BorderLayout.CENTER);
  panel_1.setLayout(new GridLayout(0, 1, 0, 0));

  JPanel panel_8 = new JPanel();
  panel_1.add(panel_8);
```

```
        JPanel panel_9 = new JPanel();
        panel_1.add(panel_9);

        JLabel lblRegisterNewCar = new JLabel("Register New Car");
        panel_9.add(lblRegisterNewCar);
        lblRegisterNewCar.setFont(new Font("Times New Roman",
Font.PLAIN, 28));

        JPanel panel_10 = new JPanel();
        panel_1.add(panel_10);
        panel_10.setLayout(new GridLayout(0, 1, 0, 0));

        JPanel panel_12 = new JPanel();
        panel_10.add(panel_12);

        lblIdXXXXXXXX = new JLabel("There is no such a ID number.");

        panel_12.add(lblIdXXXXXXXX);

        JPanel panel_13 = new JPanel();
        panel_10.add(panel_13);

        lblNameXXXXX = new JLabel("Name: XXXXX");
        panel_13.add(lblNameXXXXX);

        JPanel panel_14 = new JPanel();
        panel_10.add(panel_14);

        lblEmailXXXXXXXXXcom = new JLabel("Email:
XXXXXXXX@XXX.com");
        panel_14.add(lblEmailXXXXXXXXXcom);

        JPanel panel_11 = new JPanel();
        panel_1.add(panel_11);

        btnRegister = new JButton("Register");
        panel_11.add(btnRegister);
        register();

    }
    public void register(){
        btnRegister.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
        setVisible(false);
        check();
        new RegisterNewCar_1().setVisible(true);
    }
});
}

public void check(){
    registerInput=lblIdXXXXXXX.getText();
    if( registerInput.equals("There is no such a ID number. "))
    {
        JOptionPane.showMessageDialog(null, "You can't register since you are
not a QM staff.", "FAILURE", JOptionPane.ERROR_MESSAGE);
    }
    else{
        try{

            FileReader fr=new FileReader("registerlist.txt");
            BufferedReader br=new BufferedReader(fr);
            String line=br.readLine();
            flag=0;
            while (line!=null)
            {

                if (line.equals(registerInput))
                {
                    JOptionPane.showMessageDialog(null, "You can't register
again since you have registered", "FAILURE", JOptionPane.ERROR_MESSAGE);
                    flag=1;
                    break;
                }
                line=br.readLine();
            }

            br.close();
            fr.close();

            if (flag==0){
                FileWriter fw=new FileWriter("registerlist.txt",true);
                BufferedWriter bw=new BufferedWriter(fw);
                bw.write(registerInput);
                bw.newLine();
            }
        }
    }
}
```

```
        bw.flush();
        fw.close();

        JOptionPane.showMessageDialog(null, "Congratuation! You are
registered in the car park successfully!", "SUCCESS",
JOptionPane.INFORMATION_MESSAGE);
    }
    }
    catch(IOException ev)
    {
        System.out.println("Cannot find the file.");
    }
}

}
```

```
class RegisterNewCar_1 extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    public JTextField textField;
    public static String ID;
    JButton btnSearch;
    RegisterNewCar frame2;
    String searchInput;
    public RegisterNewCar_1 frame;
    /**
     * Launch the application.
     */

    public void run() {
        try {
            frame = new RegisterNewCar_1();
            frame.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
    }

/**
 * Create the frame.
 */
public RegisterNewCar_1() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(450, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.EAST);
    panel.setLayout(new GridLayout(6, 1, 0, 0));

    JPanel panel_2 = new JPanel();
    panel.add(panel_2);

    JButton btnHome = new JButton("    Home    ");
    panel_2.add(btnHome);
    btnHome.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            choose frame0=new choose();
            frame0.setVisible(true);
            frame0.lblOperator.setText("Operator: "+ID);
        }
    });

    JPanel panel_3 = new JPanel();
    panel.add(panel_3);

    JButton btnModifyParkInfo = new JButton("Modify Park Info");
    panel_3.add(btnModifyParkInfo);
    btnModifyParkInfo.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new ModifyParkInfo().setVisible(true);
        }
    });
}
```



```
        JPanel panel_4 = new JPanel();
        panel.add(panel_4);

        JPanel panel_5 = new JPanel();
        panel.add(panel_5);

        JPanel panel_6 = new JPanel();
        panel.add(panel_6);

        JPanel panel_7 = new JPanel();
        FlowLayout flowLayout = (FlowLayout) panel_7.getLayout();
        flowLayout.setAlignment(FlowLayout.RIGHT);
        panel.add(panel_7);

        JButton btnLogout = new JButton("Logout");
        panel_7.add(btnLogout);
        btnLogout.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                new Operator_UI().setVisible(true);
            }
        });

        JPanel panel_1 = new JPanel();
        contentPane.add(panel_1, BorderLayout.CENTER);
        panel_1.setLayout(new GridLayout(0, 1, 0, 0));

        JPanel panel_8 = new JPanel();
        panel_1.add(panel_8);

        JPanel panel_9 = new JPanel();
        panel_1.add(panel_9);

        JLabel lblRegisterNewCar = new JLabel("Register New Car");
        panel_9.add(lblRegisterNewCar);
        lblRegisterNewCar.setFont(new Font("Times New Roman",
Font.PLAIN, 28));

        JPanel panel_10 = new JPanel();
        panel_1.add(panel_10);

        JLabel lblId = new JLabel("ID:");
        lblId.setFont(new Font("SimSun-ExtB", Font.PLAIN, 18));
```

```
        panel_10.add(lblId);

        textField = new JTextField();
        panel_10.add(textField);
        textField.setColumns(10);

        JPanel panel_11 = new JPanel();
        panel_1.add(panel_11);

        btnSearch = new JButton("Search");
        panel_11.add(btnSearch);
        search();
    }

    public void search(){
        btnSearch.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                frame2=new RegisterNewCar();
                frame2.setVisible(true);
                searchInput=textField.getText();
                try{
                    FileReader fr=new FileReader("StaffDB.txt");
                    BufferedReader br=new BufferedReader(fr);
                    String num=br.readLine();
                    int numOfOP=Integer.parseInt(num);
                    while(numOfOP!=0)
                    {
                        String line=br.readLine();
                        String line1=br.readLine();
                        String line2=br.readLine();

                        if (line.equals(searchInput)){
                            frame2.lblIdXXXXXXXX.setText("ID: "+searchInput);
                            frame2.lblNameXXXXX.setText("Name: "+line1);
                            frame2.lblEmailXXXXXXXXXcom.setText("Email: "+line2);
                            break;
                        }
                    }
                    br.close();
                    fr.close();}
                catch(IOException ev)
```

```
        {
            System.out.println("Cannot find the file.");
        }
    }
});
}
```

```
class ModifyParkInfo extends JFrame {
```

```
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public JPanel contentPane;
    public JTextPane textField; //Park Sets
    public JTextPane textField_1; //0-2
    public JTextPane textField_2; //2-4
    public JTextPane textField_3; //4-8
    public JTextPane textField_4; //Y1
    public JTextPane textField_5; //M1
    public JTextPane textField_6; //D1
    public JTextPane textField_7; //Y2
    public JTextPane textField_8; //M2
    public JTextPane textField_9; //D2
    public JTextPane textField_10; //8-12
    public JTextPane textField_11; //12-24
    public static String ID;
    JButton btnUpdate;

    ArrayList<String> store;
    FileWriter fw;
    BufferedWriter bw;

    /**
     * Launch the application.
     */

    public void run() {
```

```
        try {
            ModifyParkInfo frame = new ModifyParkInfo();
            frame.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

/**
 * Create the frame.
 */
public ModifyParkInfo() {

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(450, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    contentPane.setLayout(new BorderLayout(0, 0));
    setContentPane(contentPane);

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.EAST);
    panel.setLayout(new GridLayout(6, 1, 0, 0));

    JPanel panel_3 = new JPanel();
    panel.add(panel_3);

    JButton button = new JButton("    Home    ");
    panel_3.add(button);
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            choose frame0=new choose();
            frame0.setVisible(true);
            frame0.lblOperator.setText("Operator: "+ID);
        }
    });

    JPanel panel_4 = new JPanel();
    panel.add(panel_4);

    JButton btnRegisterNewCar = new JButton("Register New Car");
    panel_4.add(btnRegisterNewCar);
    btnRegisterNewCar.addActionListener(new ActionListener() {
```

```
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new RegisterNewCar_1().setVisible(true);
        }
    });

    JPanel panel_5 = new JPanel();
    panel.add(panel_5);

    JPanel panel_6 = new JPanel();
    panel.add(panel_6);

    JPanel panel_7 = new JPanel();
    panel.add(panel_7);

    JPanel panel_8 = new JPanel();
    FlowLayout flowLayout = (FlowLayout) panel_8.getLayout();
    flowLayout.setAlignment(FlowLayout.RIGHT);
    panel.add(panel_8);

    JButton button_2 = new JButton("Logout");
    panel_8.add(button_2);
    button_2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new Operator_UI().setVisible(true);
        }
    });

    JPanel panel_1 = new JPanel();
    contentPane.add(panel_1, BorderLayout.CENTER);
    panel_1.setLayout(new GridLayout(0, 1, 0, 0));

    JPanel panel_2 = new JPanel();
    panel_1.add(panel_2);

    JPanel panel_9 = new JPanel();
    panel_1.add(panel_9);

    JLabel lblModifyParkInfo = new JLabel("Modify Park Info");
    lblModifyParkInfo.setFont(new Font("Times New Roman", Font.PLAIN,
28));
    panel_9.add(lblModifyParkInfo);
```

```
JPanel panel_10 = new JPanel();
panel_1.add(panel_10);
panel_10.setLayout(new GridLayout(0, 2, 0, 0));

JPanel panel_14 = new JPanel();
panel_10.add(panel_14);

JLabel lblParkSets = new JLabel("Park Sets:");
panel_14.add(lblParkSets);

textField = new JTextPane();
panel_14.add(textField);

JPanel panel_15 = new JPanel();
panel_10.add(panel_15);
panel_15.setLayout(new GridLayout(2, 0, 0, 0));

JPanel panel_16 = new JPanel();
FlowLayout flowLayout_1 = (FlowLayout) panel_16.getLayout();
flowLayout_1.setAlignment(FlowLayout.LEFT);
panel_15.add(panel_16);

JLabel lblTarrif = new JLabel("Tariffs:");
lblTarrif.setVerticalAlignment(SwingConstants.TOP);
lblTarrif.setHorizontalAlignment(SwingConstants.LEFT);
panel_16.add(lblTarrif);

JPanel panel_17 = new JPanel();
FlowLayout flowLayout_2 = (FlowLayout) panel_17.getLayout();
flowLayout_2.setVgap(0);
flowLayout_2.setHgap(0);
flowLayout_2.setAlignment(FlowLayout.LEFT);
panel_15.add(panel_17);

JLabel lblHours = new JLabel(" 0 to 2 Hours - \u00A3 ");
panel_17.add(lblHours);

textField_1 = new JTextPane();
panel_17.add(textField_1);
//textField_1.setColumns(3);

JPanel panel_11 = new JPanel();
panel_1.add(panel_11);
```

```
panel_11.setLayout(new GridLayout(1, 0, 0, 0));

JPanel panel_18 = new JPanel();
panel_11.add(panel_18);
panel_18.setLayout(new GridLayout(2, 0, 0, 0));

JLabel lblOpen = new JLabel("Open Time:");
panel_18.add(lblOpen);

JLabel lblYyyyMmDd = new JLabel("YYYY MM DD ");
lblYyyyMmDd.setHorizontalAlignment(SwingConstants.TRAILING);
panel_18.add(lblYyyyMmDd);

JPanel panel_19 = new JPanel();
panel_11.add(panel_19);
panel_19.setLayout(new GridLayout(2, 0, 0, 0));

JPanel panel_20 = new JPanel();
FlowLayout flowLayout_3 = (FlowLayout) panel_20.getLayout();
flowLayout_3.setVgap(0);
flowLayout_3.setHgap(0);
flowLayout_3.setAlignment(FlowLayout.LEFT);
panel_19.add(panel_20);

JLabel lblTo = new JLabel(" 2 to 4 Hours - \u00A3 ");
panel_20.add(lblTo);

textField_2 = new JTextPane();
//textField_2.setColumns(3);
panel_20.add(textField_2);

JPanel panel_21 = new JPanel();
FlowLayout flowLayout_4 = (FlowLayout) panel_21.getLayout();
flowLayout_4.setVgap(0);
flowLayout_4.setHgap(0);
flowLayout_4.setAlignment(FlowLayout.LEFT);
panel_19.add(panel_21);

JLabel lblTo_1 = new JLabel(" 4 to 8 Hours - \u00A3 ");
panel_21.add(lblTo_1);

textField_3 = new JTextPane();
//textField_3.setColumns(3);
panel_21.add(textField_3);
```

```
JPanel panel_12 = new JPanel();
panel_1.add(panel_12);
panel_12.setLayout(new GridLayout(0, 2, 0, 0));

JPanel panel_22 = new JPanel();
panel_12.add(panel_22);
panel_22.setLayout(new GridLayout(2, 4, 0, 0));

JPanel panel_24 = new JPanel();
panel_22.add(panel_24);

JLabel lblFrom = new JLabel("From");
panel_24.add(lblFrom);

JPanel panel_25 = new JPanel();
FlowLayout flowLayout_5 = (FlowLayout) panel_25.getLayout();
flowLayout_5.setVgap(0);
flowLayout_5.setHgap(0);
panel_22.add(panel_25);

textField_4 = new JTextPane();
//textField_4.setHorizontalAlignment(SwingConstants.LEFT);
panel_25.add(textField_4);
//textField_4.setColumns(3);

JPanel panel_26 = new JPanel();
FlowLayout flowLayout_6 = (FlowLayout) panel_26.getLayout();
flowLayout_6.setVgap(0);
flowLayout_6.setHgap(0);
panel_22.add(panel_26);

textField_5 = new JTextPane();
panel_26.add(textField_5);
//textField_5.setColumns(2);

JPanel panel_27 = new JPanel();
FlowLayout flowLayout_7 = (FlowLayout) panel_27.getLayout();
flowLayout_7.setVgap(0);
flowLayout_7.setHgap(0);
panel_22.add(panel_27);

textField_6 = new JTextPane();
//textField_6.setColumns(2);
```



```
panel_27.add(textField_6);

JPanel panel_28 = new JPanel();
panel_22.add(panel_28);

JLabel lblTo_2 = new JLabel("To");
panel_28.add(lblTo_2);

JPanel panel_29 = new JPanel();
FlowLayout flowLayout_8 = (FlowLayout) panel_29.getLayout();
flowLayout_8.setVgap(0);
flowLayout_8.setHgap(0);
panel_22.add(panel_29);

textField_7 = new JTextPane();
//textField_7.setHorizontalAlignment(SwingConstants.LEFT);
//textField_7.setColumns(3);
panel_29.add(textField_7);

JPanel panel_30 = new JPanel();
FlowLayout flowLayout_9 = (FlowLayout) panel_30.getLayout();
flowLayout_9.setVgap(0);
flowLayout_9.setHgap(0);
panel_22.add(panel_30);

textField_8 = new JTextPane();
panel_30.add(textField_8);
//textField_8.setColumns(2);

JPanel panel_31 = new JPanel();
FlowLayout flowLayout_10 = (FlowLayout) panel_31.getLayout();
flowLayout_10.setHgap(0);
flowLayout_10.setVgap(0);
panel_22.add(panel_31);

textField_9 = new JTextPane();
panel_31.add(textField_9);
//textField_9.setColumns(2);

JPanel panel_32 = new JPanel();
panel_12.add(panel_32);
panel_32.setLayout(new GridLayout(2, 0, 0, 0));

JPanel panel_33 = new JPanel();
```

```
FlowLayout flowLayout_11 = (FlowLayout) panel_33.getLayout();
flowLayout_11.setAlignment(FlowLayout.LEFT);
flowLayout_11.setVgap(0);
flowLayout_11.setHgap(0);
panel_32.add(panel_33);

JLabel lblTohours = new JLabel(" 8 to12Hours - \u00A3 ");
panel_33.add(lblTohours);

textField_10 = new JTextPane();
//textField_10.setColumns(3);
panel_33.add(textField_10);

JPanel panel_34 = new JPanel();
FlowLayout flowLayout_12 = (FlowLayout) panel_34.getLayout();
flowLayout_12.setVgap(0);
flowLayout_12.setHgap(0);
flowLayout_12.setAlignment(FlowLayout.LEFT);
panel_32.add(panel_34);

JLabel lbltohours = new JLabel(" 12to24Hours - \u00A3 ");
panel_34.add(lbltohours);

textField_11 = new JTextPane();
//textField_11.setColumns(3);
panel_34.add(textField_11);

JPanel panel_13 = new JPanel();
panel_1.add(panel_13);

btnUpdate = new JButton("Update");
panel_13.add(btnUpdate);
confirm();
setInitial();
}

public void confirm(){
    btnUpdate.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new choose().setVisible(true);
            update();
        }
    });
}

System.out.println(Car_Park.d1+""+Car_Park.m1+""+Car_Park.y1);
```

```
System.out.println(Car_Park.d2+""+Car_Park.m2+""+Car_Park.y2);

    }
});
}

public void update(){

    if(isInteger(textField.getText()))
        Car_Park.parking_space=Integer.parseInt(textField.getText());
    if(isDouble(textField_1.getText())||isInteger(textField_1.getText()));
        Car_Park.price1=(Double.parseDouble(textField_1.getText()));
    if(isDouble(textField_2.getText())||isInteger(textField_2.getText()))
        Car_Park.price2=Double.parseDouble(textField_2.getText());
    if(isDouble(textField_3.getText())||isInteger(textField_3.getText()))
        Car_Park.price3=Double.parseDouble(textField_3.getText());
    if(isDouble(textField_10.getText())||isInteger(textField_10.getText()))
        Car_Park.price4=Double.parseDouble(textField_10.getText());
    if(isDouble(textField_11.getText())||isInteger(textField_11.getText()))
        Car_Park.price5=Double.parseDouble(textField_11.getText());
    if(isInteger(textField_4.getText()))
        Car_Park.y1=Integer.parseInt(textField_4.getText())-1900;
    if(isInteger(textField_5.getText()))
        Car_Park.m1=Integer.parseInt(textField_5.getText())-1900;
    if(isInteger(textField_6.getText()))
        Car_Park.d1=Integer.parseInt(textField_6.getText())-1;
    if(isInteger(textField_7.getText()))
        Car_Park.y2=Integer.parseInt(textField_7.getText())-1;
    if(isInteger(textField_8.getText()))
        Car_Park.m2=Integer.parseInt(textField_8.getText());
    if(isInteger(textField_9.getText()))
        Car_Park.d2=Integer.parseInt(textField_9.getText());

}

public void setInitial(){

    textField.setText(""+Car_Park.parking_space);
    textField_1.setText(""+Car_Park.price1);
    textField_2.setText(""+Car_Park.price2);
    textField_3.setText(""+Car_Park.price3);
```

```
        textField_10.setText(""+Car_Park.price4);
        textField_11.setText(""+Car_Park.price5);
        textField_4.setText("xxxx");
        textField_5.setText("xx");
        textField_6.setText("xx");
        textField_7.setText("xxxx");
        textField_8.setText("xx");
        textField_9.setText("xx");

    }

    public static boolean isDouble(String value) {
        try {
            Double.parseDouble(value);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

    public static boolean isInteger(String value) {
        try {
            Integer.parseInt(value);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

}

```

Operator_Control.java

```
/**
 * @author Yao Ping
 *
 */

import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;

```

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import javax.swing.JOptionPane;

public class Operator_Control {

    Operator_UI frame;
    choose frame1;
    RegisterNewCar_1 frame2;
    RegisterNewCar frame3;
    ModifyParkInfo frame4;

    public Operator_Control(){
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new Operator_UI();
                    frame.setVisible(true);

                    frame1 = new choose();
                    frame1.setVisible(false);

                    frame2 = new RegisterNewCar_1();
                    frame2.setVisible(false);
                    if(frame2.isVisible()==true){
                        search();
                    }

                    frame3 = new RegisterNewCar();
                    frame3.setVisible(false);
                    if(frame3.isVisible()==true){
                        addAccount();
                        check();
                    }

                    frame4 = new ModifyParkInfo();
                    frame4.setVisible(false);
                    if(frame3.isVisible()==true){
                        update();
                    }
                }
            }
        });
    }
}
```

```
        }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

}

public void input(){
    frame.login();
}

public void addAccount(){
    frame3.register();
}

public void search(){
    frame2.search();
}

public void check(){
    frame3.check();
}

public void update(){
    frame4.confirm();
}

}
Car_Park.java

/**
 * @author Tong Ge
 *
 */

import java.util.ArrayList;
```

```
import java.util.Date;
import java.awt.EventQueue;
import java.io.BufferedWriter;
import java.io.FileWriter;

public class Car_Park {
    static public int amount_of_tickets;           //total amount of
tickets
    static public double amount_of_coins;         //total coins in
the collection box
    static public double collection;             //the collection
in payment station
    static public int parking_space;             //total parking
places
    static public int remaining_space;           //remaining
parking places
    static public int remaining_ticket;          //remaining
tickets in the ticket box
    static public double max_money=100;          //Maximum
money can be stored in the collection box
    static public double price1,price2,price3,price4,price5; //price for different
time interval
    static public Date opening_time[];          //opening time
for public
    static int y1=0;                             //start year
    static int y2=0;                             //end year
    static int m1=0;                             //start
month
    static int m2=0;                             //end
month
    static int d1=0;                             //start day
    static int d2=0;                             //end day
    static Date date1=new Date();
    static Date date2=new Date();

    FileWriter fw;
    BufferedWriter bw;
    ArrayList<String> update;
    ArrayList<Integer> updatenum;
```

```
public Car_Park(){
    amount_of_tickets=10;           //initial number of
tickets
    amount_of_coins=0;             //initial amount of coins
    parking_space=30;             //initial parking places
    remaining_space=parking_space;
    remaining_ticket=amount_of_tickets;
    price1=0.5;                   //initial parking
tariffs
    price2=1;
    price3=2;
    price4=3;
    price5=5;
}
```

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                new Driver_Control();
                System.out.println("Driver_Control succeed");
                new Operator_Control();
                System.out.println("Operator_Control succeed");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

```
public static boolean isInteger(String value) {
    try {
        Integer.parseInt(value);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}
```



```

    }

    public String refreshEnter(){
        String output;
        output="The remaining space: "+remaining_space+"\nTarrif:\n" +
            "QM STAFF:\t"+"\" +price1
        +"\n\t2-4hours\t\" +price2 +
            "\n\t4-8hours\t\" +price3  +"\n\t8-12hours\t\" +price4
        +"\n\t12-24hours\t\" +price5 +"\r\n\r\n";
        return output;
    }

    public String refreshPayment(){
        String output;
        output="\n\nPUBLICTarrif:\n" +
            "0-2hours\t\" +price1 +"\n2-4hours\t\" +price2 +
            "\n4-8hours\t\" +price3  +"\n8-12hours\t\" +price4
        +"\n12-24hours\t\" +price5 +"\r\n\r\n";
        return output;
    }
}

```

Usage.java

```

/**
 * @author Zhijiao Liu
 *
 */

import java.util.*;

public class Usage {

    public String id;           //ID number
    public int number;         //park place number
    public Date enter_time;    //record enter time
    public Date payment_time;  //record payment time
    public double charge;      //the charge of each use
    public double collection;  //the collection that the public paid
    public float parking_hours; //the total hours in this parking time
    public int accountlistnum; //record the usage of an account
    public boolean type;       //decide the usage of public or staff
}

```

```
Usage(String id)
{
    this.id=id;
    this.enter_time=new Date();
    System.out.println("ID: "+this.id+" Enter time: "+this.enter_time);
}
Usage(){

}

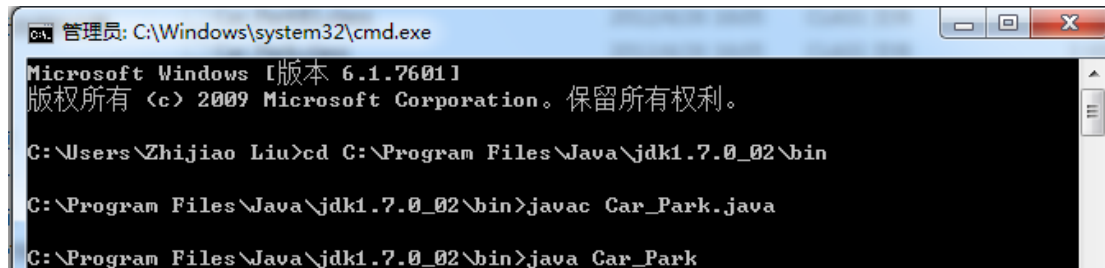
public void paymentTime()
{
    this.payment_time=new Date();
    System.out.println("ID: "+this.id+" Payment time:
"+this.payment_time);
}

}
```

Appendix 3 – User Manual

1. How to run the java program

After implement the java programming environment, press Win + R and type cmd to open cmd.exe. Find the path of java, like “C:\Program Files\Java\jdk1.7.0_02\bin”. Use “cd” to change the path in cmd to reach the javac.exe location. Copy all the .java files in src folder to bin folder, then type the commands as shown in the picture.



```

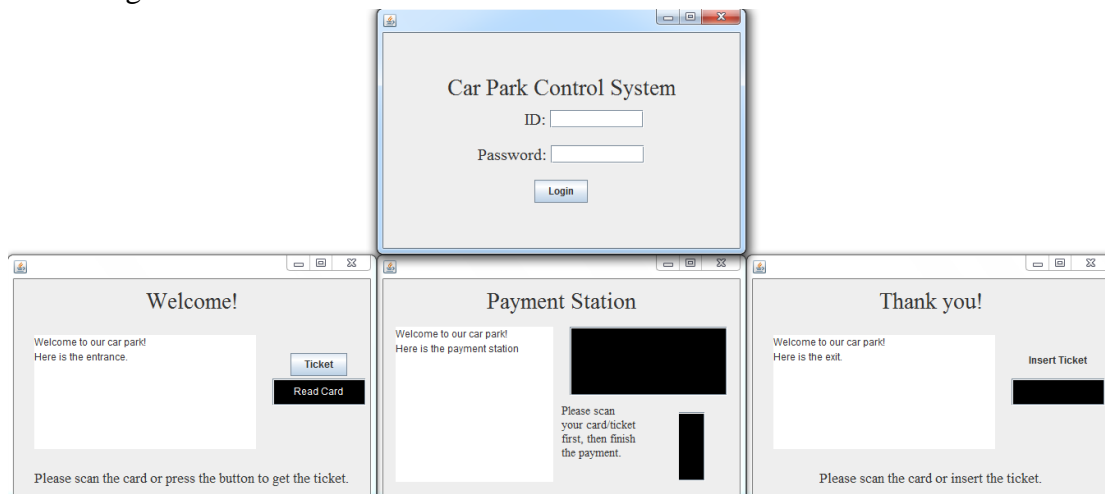
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Zhijiao Liu>cd C:\Program Files\Java\jdk1.7.0_02\bin

C:\Program Files\Java\jdk1.7.0_02\bin>javac Car_Park.java

C:\Program Files\Java\jdk1.7.0_02\bin>java Car_Park
  
```

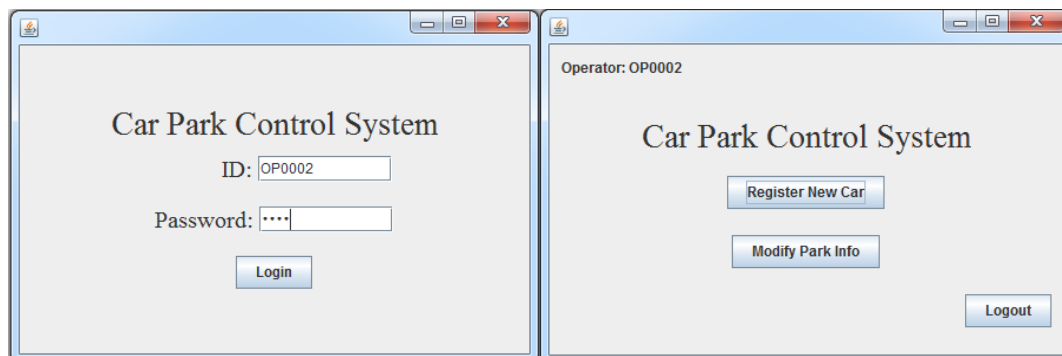
You will get the UI after a few seconds.



2. Operator use guide

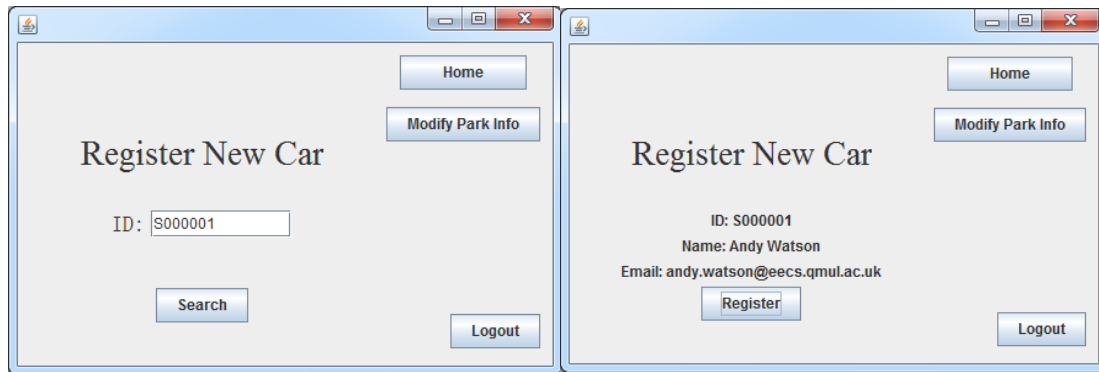
a) Login

Type the Operator’s ID and password. If the ID and password are correct, login succeeds and the operator can choose the following step.



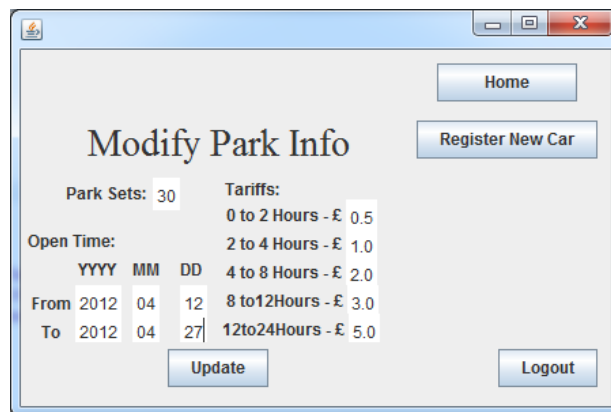
b) Register new car

Enter the staff’s ID and search his/her information in QM’s database. If the ID is in the database, the staff’s info will appear in the window and operator can do the registration.



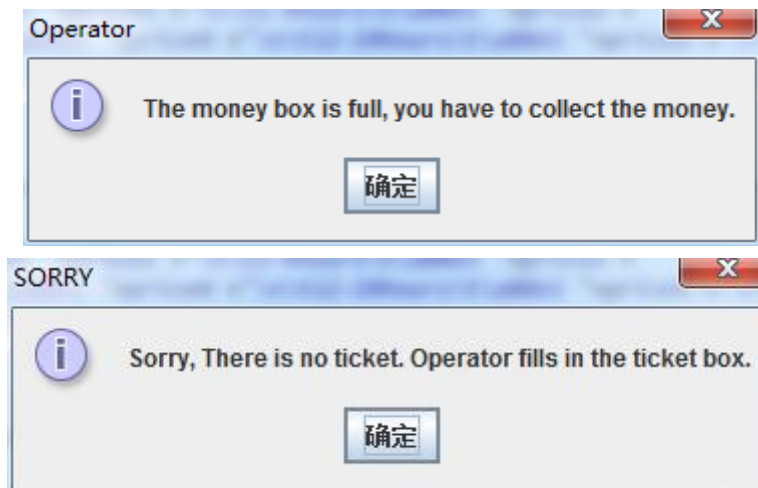
c) Modify park information

In this section, operator can update the data of park sets, tariffs and opening time for public.



d) Get the usage warning message

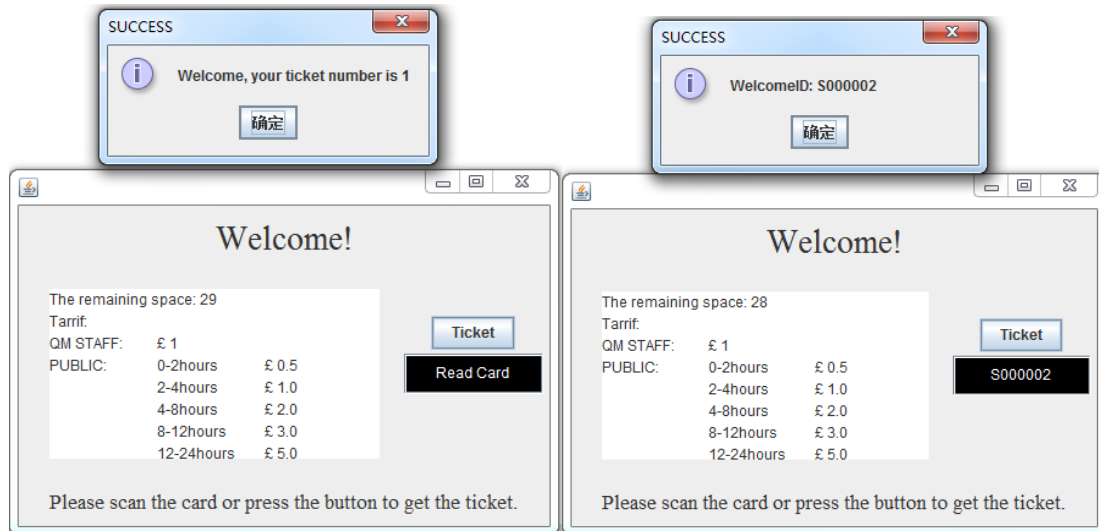
When the money box is full or the ticket box is empty, this program will show a warning to inform the operator.



3. Driver use guide

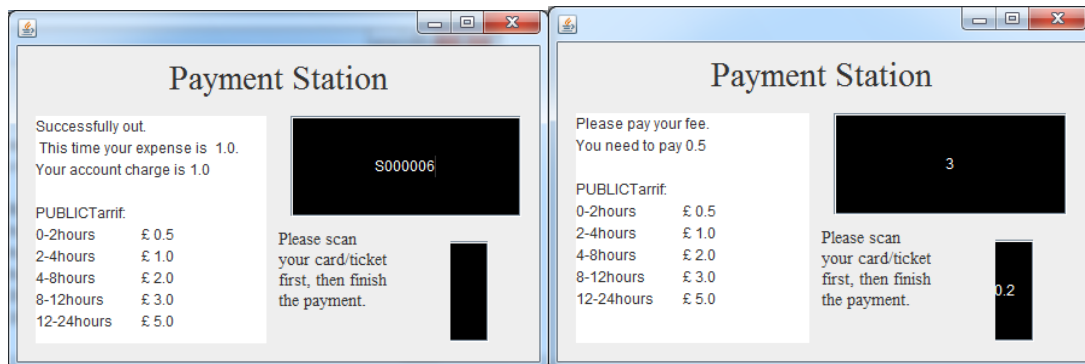
a) Entrance

Staff can scan his/her campus card to get into the park and public can press the “ticket” button to get the ticket, then the timer of parking will start.



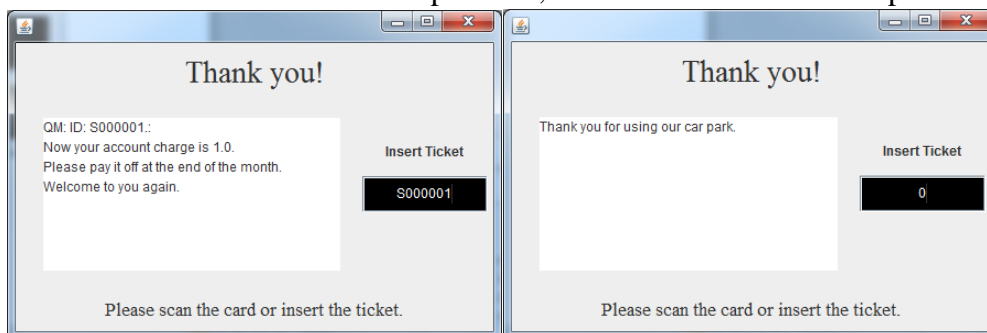
b) Payment station

In the payment station, after scan the staff's campus card or the public's ticket, the screen will show the parking expense of this time. And the public can insert the coins to finish the payment.



c) Exit

After insert the ticket or scan the campus card, the driver can exit the car park.



Appendix 4 – Supplement Information

Interview:

Title	Response from staffs
Type	One to One talking, Email
Respondents	Dr. Marie-Luce Bourguet, Dr. Frank Gao, Andy Watson
Time	2011.3
Location	Teaching Building 1
Core Questions	<ul style="list-style-type: none"> ● What are factors that influence your choice of parking cars? ● The system requires staffs register accounts of them before using. Do you prefer to use ID number or personal details?
Analysis	Car parks located near work places are preferred. Registering only with ID number is more secure and convenient.

Title	Response From End Users and Operators
Type	One to One talking
Respondents	two operators and two drivers
Time	2011.3
Location	BUPT main campus
Core Questions	<ul style="list-style-type: none"> ● What's the problem in the existing parking system? ● If the payment station can only accept coins and cannot give change, would you still want to use the system?
Analysis	The most critical concern of the users is usability and reliability. It's acceptable to only accept coins and no change given.

Questionnaire:

Core questions of our questionnaire and results are as follows:

1. What information do you want to use when register your car?(to drivers)

A. ID number only.	48%
B. ID number with proper personal information.	26%
C. ID number with detailed personal information.	14%
D. Do not care.	12%

2. Which feature is most desirable when choosing parking system software?

A. Whether it is extensible.	31%
B. Whether it is easy to use.	43%
C. Low error rate.	15%
D. Others	11%

3. What kind of operations do you want?(to operator)

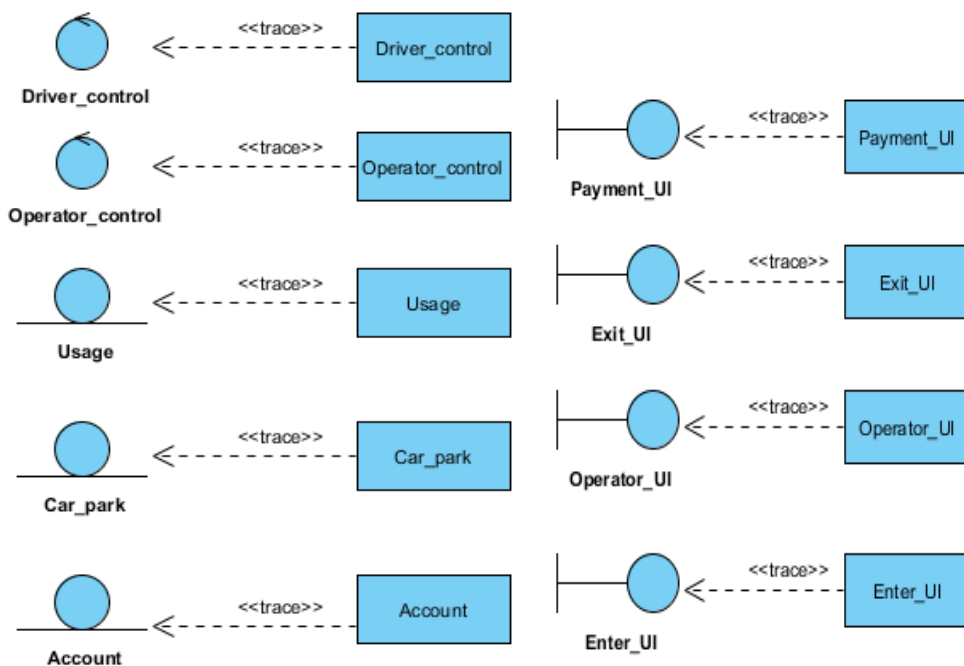
A. Visualization operation	35%
----------------------------	-----

B. Input command language	16%
C. Both A and B	39%
D. Others	10%

4. Which elements influence your evaluation to the parking lot? (to drivers)

A. The response time of the system.	30%
B. Simple procedure	53%
C. Update record punctually	15%
D. Others	2%

Outlining Design Classes



Testing

Test cases and procedures

1. Login

1.2 Test case for login (correct input)

Input

A valid operator account exists; operator ID is 01 and password is 1234.

Operator ID 01 has been entered and password 1234 has been entered.

The system carried out the check.

Result

The system confirms that 01 exists and the associated password is correct.

Conditions

No conditions exist on this test.

1.3 Test case for login (incorrect input)

Input

A valid operator account exists; operator ID is 01 and password is 1234.

Account ID 02 has been entered and password 1234 has been entered.

The system carried out the check.

Result

The system confirms that the 02 does not exist.

Conditions

No conditions exist on this test.

1.4 Test case for login (incorrect input)

Input

A valid operator account exists; operator ID is 01 and password is 1234.

Account ID 02 has been entered and password 1243 has been entered.

The system carried out the check.

Result

The system confirms that the 02 does not exist.

Conditions

No conditions exist on this test.

1.1 Test procedure for login(correct input)

From the login screen, enter the operator ID, 01 and password, 1234 in the appropriate boxes and select enter. The system checks the operator ID and password, and the main screen appears.

1.2 Test procedure for login(incorrect input)

From the login screen, enter the operator ID, 02 and password, 1234 in the appropriate boxes and select enter. The system checks the operator ID and password and returns “Incorrect account number or password, please try again”, and the login screen reappears.

1.3 Test procedure for login(incorrect input)

From the login screen, enter the operator ID, 02 and password, 1243 in the appropriate boxes and select enter. The system checks the operator ID and password and returns “Incorrect account number or password, please try again”, and the login screen reappears.

2. Register

2.1 Test case for register (incorrect input)

Input

A valid staff account exists in the system, account ID is 00001; and a valid staff ID exists in the financial department (txt), staff ID is 00002.

ID 00001 has been entered.

The system carried out the check.

Result

The system confirms that 00001 exists in the system.

Conditions

Operator has logged in.

2.2 Test case for register (correct input)

Input

A valid staff account exists in the system, account ID is 00001; and a valid staff ID exists in the financial department (txt), staff ID is 00002.

ID 00002 has been entered.

The system carried out the check.

Result

The system confirms that 00002 does not exist in the system, but in the financial department (txt); creates new account with the ID 00002.

Conditions

Operator has logged in.

2.3 Test case for register (incorrect input)

Input

A valid staff account exists in the system, account ID is 00001; and a valid staff ID exists in the financial department (txt), staff ID is 00002.

ID 00003 has been entered.

The system carried out the check.

Result

The system confirms that 00003 exists neither in the system, nor in the financial department (txt)

Conditions

Operator has logged in.

2.1 Test procedure for register (incorrect input)

1. From the main screen, select the register new car button. The registration screen is displayed.
2. From the registration screen, enter the staff ID 00001 in the appropriate boxes and select search. The system checks the account list, returns “00001 had been registered before” and the registration screen reappears below the message.

2.2 Test procedure for register (correct input)

1. From the main screen, select the register new car button. The registration screen is displayed.
2. From the registration screen, enter the staff ID 00002 in the appropriate boxes and select search. The system checks the account list and the financial department (txt), returns the details of the staff ID.
3. From the registration screen, select the register button. The system returns “00002 has been registered successfully” and creates new account with the ID 00002, the registration screen reappears below the message.

2.3 Test procedure for register (incorrect input)

1. From the main screen, select the register new car button. The registration screen is displayed.
2. From the registration screen, enter the staff ID 00003 in the appropriate boxes and select search. The system checks the account list and the financial department (txt),

returns “00003 is not an available ID” and the registration screen reappears below the message.

Update

3.1 Test case for modify open time (incorrect input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

Delete the value in the first blank of open time blanks.

The system carried out the check.

Result

The system does not accept the new open time.

Conditions

Operator has logged in.

3.2 Test case for modify open time (correct input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, 2012, 01, 01, 2012, 01, 01 have been entered in the blanks of open time.

The system carried out the check.

Result

The system accepts the new open time and updates it in the system.

Conditions

Operator has logged in.

3.3 Test case for modify open time (incorrect input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, 2012, 01, 02, 2012, 01, 01 have been entered in the blanks of open time.

The system carried out the check.

Result

The system does not accept the new open time.

Conditions

Operator has logged in.

3.4 Test case for modify open time (incorrect input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, 2012, 01, 01, 2012, 01, 32 have been entered in the blanks of open time

The system carried out the check.

Result

The system does not accept the new open time.

Conditions

Operator has logged in.

3.5 Test case for modify open time (incorrect input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, 01, 01, 2012, 02, 01, 2012 have been entered in the blanks of open time

The system carried out the check.

Result

The system does not accept the new open time.

Conditions

Operator has logged in.

3.6 Test case for modify open time (incorrect input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, a, 01, 01, 2012, 01, 01 have been entered in the blanks of open time.

The system carried out the check.

Result

The system does not accept the new open time.

Conditions

Operator has logged in.

3.7 Test case for modify open time (correct input)

Input

The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, 2012, 01, 03, 2012, 01, 04 have been entered in the blanks of open time.

The system carried out the check.

Result

The system accepts the new open time and updates it in the system.

Conditions

Operator has logged in.

3.8 Test case for modify tariff (incorrect input)

Input

The tariff exists in the system; values are 1, 2, 3, 4, and 5.

Delete all the values in the tariff blanks..

The system carried out the check.

Result

The system does not accept the new tariff.

Conditions

Operator has logged in

3.9 Test case for modify tariff (correct input)

Input

The tariff exists in the system; values are 1, 2, 3, 4, and 5.

New values, 0, 1, 1, 2 and 99.99 have been entered in the blanks of tariff.

The system carried out the check.

Result

The system accepts the new tariff and updates it in the system.

Conditions

Operator has logged in.

3.10 Test case for modify tariff (incorrect input)

Input

The tariff exists in the system; values are 1, 2, 3, 4, and 5.

New values, 5, 4, 3, 2 and 1 have been entered in the blanks of tariff.

The system carried out the check.

Result

The system does not accept the new tariff.

Conditions

Operator has logged in

3.11 Test case for modify tariff (incorrect input)

Input

The tariff exists in the system; values are 1, 2, 3, 4, and 5.

New values, 1, 2, 3, 4 and a have been entered in the blanks of tariff.

The system carried out the check.

Result

The system does not accept the new tariff.

Conditions

Operator has logged in

3.2 Test procedure for modify open time (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, delete the value in the first blank of open time blanks and select update. The system checks the new open time, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.3 Test procedure for modify open time (correct input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new open time, 2012, 01, 01, 2012, 01, 01 have been entered in the blanks of open time and select update. The system checks the new open time, update the new open time in the system and returns “New park info has been updated successfully”, the park info screen reappears below the message.

3.4 Test procedure for modify open time (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new open time, 2012, 01, 02, 2012, 01, 01 have been

entered in the blanks of open time and select update. The system checks the new open time, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.5 Test procedure for modify open time (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new open time, 2012, 01, 01, 2012, 01, 32 have been entered in the blanks of open time and select update. The system checks the new open time, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.6 Test procedure for modify open time (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new open time, 01, 01, 2012, 02, 01, 2012 have been entered in the blanks of open time and select update. The system checks the new open time, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.7 Test procedure for modify open time (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new open time, a, 01, 01, 2012, 01, 01 have been entered in the blanks of open time and select update. The system checks the new open time, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.8 Test procedure for modify open time (correct input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new open time, 2012, 01, 03, 2012, 01, 04 have been entered in the blanks of open time and select update. The system checks the new open time, update the new open time in the system and returns “New park info has been updated successfully”, the park info screen reappears below the message.

3.9 Test procedure for modify tariff (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, delete all the values in the tariff blanks and select update. The system checks the new tariff, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.10 Test procedure for modify tariff (correct input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new tariff, 0, 1, 1, 2 and 99.99 have been entered in the blanks of tariff and select update. The system updates the new tariff in the system and returns “New park info has been updated successfully”, the park info screen reappears below the message.

3.12 Test procedure for modify tariff (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new tariff, 1, 2, 3, 4 and 10000 have been entered in the blanks of tariff and select update. The system checks the new tariff, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

3.13 Test procedure for modify tariff (incorrect input)

1. From the main screen, select the modify park info button. The park info screen is displayed.
2. From the park info screen, new tariff, 1, 2, 3, 4 and a have been entered in the blanks of tariff and select update. The system checks the new tariff, returns “Each value of park info can be formed with four numbers at most, may having a point in it and please ensure the new park info is logical” and the park info screen reappears below the message.

Confirm

4.1 Test case for money confirm (correct input)

Input

The max money is set to 100 and the amount of coins is set to 100.

Money confirm button is pressed.

The system carried out the check.

Result

The system accepts the order.

Conditions

Operator has logged in.

4.2 Test case for tickets confirm (correct input)

Input

The max tickets is set to 100 and the amount of tickets is set to 0.

Tickets confirm button is pressed.

The system carried out the check.

Result

The system accepts the order.

Conditions

Operator has logged in.

4.1 Test procedure for money confirm (correct input)

From the confirmation screen, select money confirm button. The system set the amount of coins to 0.

4.2 Test procedure for tickets confirm (correct input)

From the confirmation screen, select tickets confirm button. The system set the amount of tickets to 100, the value of max tickets.

Enter record

5.1 Test case for staff enter record (correct input)

Input

A valid staff account exists in the system, account ID is 00001; the remaining space is set to 2; the account state of 00001 is paid

Account ID 00001 has been entered.

The system carried out the check.

Result

The system confirms that account 00001 exists, the remaining space is above 0 and the account state of 00001 is available; opens the entrance barrier.

Conditions

No conditions exist on this test.

5.2 Test case for staff enter record (incorrect input)

Input

A valid staff account exists in the system, account ID is 00001; the remaining space is set to 0; the account state of 00001 is paid

Account ID 00001 has been entered.

The system carried out the check.

Result

The system confirms that the remaining space is 0.

Conditions

No conditions exist on this test.

5.3 Test case for staff enter record (incorrect input)

Input

A valid staff account exists in the system, account ID is 00001; the remaining space is set to 2; the account state of 00001 is unpaid

Account ID 00001 has been entered.

The system carried out the check.

Result

The system confirms that account 00001 exists, the remaining space is above 0 and the account state of 00001 is not available;

Conditions

No conditions exist on this test.

5.4 Test case for staff enter record (incorrect input)

Input

A valid staff account exists in the system, account ID is 00001; the remaining space is set to 2; the account state of 00001 is paid

Account ID 00003 has been entered.

The system carried out the check.

Result

The system confirms that account 00003 does not exist.

Conditions

No conditions exist on this test.

5.5 Test case for public enter record (correct input)

Input

A valid public account exists in the system, account ID is 00002; the remaining space is set to 2; the account state of 00002 is paid; the open time is set to 1900, 01, 01, 9999, 12, 31.

Account ID 00002 has been entered and ticket bottom is pressed.

The system carried out the check.

Result

The system confirms that public account 00002 exists, the remaining space is above 0, the account state of 00002 is available and the open time is available; opens the entrance barrier.

Conditions

No conditions exist on this test.

5.6 Test case for public enter record (incorrect input)

Input

A valid public account exists in the system, account ID is 00002; the remaining space is set to 2; the account state of 00002 is paid; the open time is set to 1900, 01, 01, 1900, 01, 01.

Account ID 00002 has been entered and ticket bottom is pressed.

The system carried out the check.

Result

The system confirms that the open time is not available

Conditions

No conditions exist on this test.

5.1 Test procedure for staff enter record (correct input)

From the entrance screen, enter the ID 00001 in the appropriate area. The system checks the ID, remaining space and the account state, opens the entrance barrier, records the enter time in the account 00001; changes the account state of 00001 to unpaid, decrease the remaining space by 1

5.2 Test procedure for staff enter record (incorrect input)

From the entrance screen, enter the ID 00001 in the appropriate area. The system checks remaining space, does nothing.

5.3 Test procedure for staff enter record (incorrect input)

From the entrance screen, enter the ID 00001 in the appropriate area. The system checks the ID, remaining space and the account state, does noting.

5.4 Test procedure for staff enter record (incorrect input)

From the entrance screen, enter the ID 00003 in the appropriate area. The system checks the ID, remaining space, does noting.

5.5 Test procedure for public enter record (correct input)

From the entrance screen, enter the ID 00002 in the appropriate area and select ticket button. The system checks the ID, remaining space, open time and the account state; gives the ticket, opens the entrance barrier, records the enter time in the account 00002; changes the account state of 00001 to unpaid, decrease the remaining space and the amount of tickets by 1.

5.6 Test procedure for public enter record (incorrect input)

From the entrance screen, enter the ID 00002 in the appropriate area. The system checks the open time, does noting.

Payment record

6.1 Test case for payment record (correct input)

Input

A valid account exists in the system, account ID is 00001; the account state of 00001 is unpaid

Account ID 00001 has been entered.

The system carried out the check.

Result

The system confirms that account 00001 exists and the account state of 00001 is available.

Conditions

No conditions exist on this test.

6.2 Test case for payment record (incorrect input)

Input

A valid account exists in the system, account ID is 00001; the account state of 00001 is unpaid

Account ID 00002 has been entered.

The system carried out the check.

Result

The system confirms that account 00002 does not exist.

Conditions

No conditions exist on this test.

6.3 Test case for payment record (incorrect input)

Input

A valid account exists in the system, account ID is 00001; the account state of 00001 is paid

Account ID 00001 has been entered.

The system carried out the check.

Result

The system confirms that account 00001 exists and the account state of 00001 is not available.

Conditions

No conditions exist on this test.

6.1 Test procedure for payment record (correct input)

From the payment screen, enter the ID 00001 in the appropriate area. The system checks the ID and the account state; records the exit time in the account 00001.

6.2 Test procedure for payment record (incorrect input)

From the payment screen, enter the ID 00001 in the appropriate area. The system checks the ID; does nothing.

6.3 Test procedure for payment record (incorrect input)

From the payment screen, enter the ID 00001 in the appropriate area. The system checks the ID and the account state; does nothing.

Test case	Test description	Pass/ Fail	No. of Bugs	Bug #	Comments
Set up 1	Operator account is added to the system. (Id is 01;password is 1234)				Add to system for testing
1.1	Test login with right ID and right password (Id is 01;password is 1234)				
1.2	Test login with right ID and wrong password (Id is 01;password is 1243)				
1.3	Test login with wrong ID and right password (Id is 02;password is 1234)				
1.4	Test login with wrong ID and wrong password (Id is 02;password is 1243)				

Test case	Test description	Pass/ Fail	No. of Bugs	Bug #	Comments
Set up 2	Staff account is added to system (ID is 00001) Staff ID is add to financial txt (ID is 00002)				Add to system for testing
2.1	Test register with ID which has existed				

	in the system. (ID is 00001)				
2.2	Test register with ID which does not exist in the system, but in financial txt. (ID is 00002)				
2.3	Test register with ID which exists neither in the system, nor in financial txt. (ID is 00003)				

Test case	Test description	Pass/ Fail	No. of Bugs	Bug #	Comments
Set up 3	Open time is added to system (2012, 01, 01, 2012, 01, 02) ; Tariff is added to system (1, 2, 3, 4, 5) .				Add to system for testing
3.1	Test enter nothing in all open time blanks. (, , , ,)				
3.2	Test enter nothing in a part of open time blanks.(, 01, 01, 2012, 01, 02)				
3.3	Test set open time to one day. (2012, 01, 01, 2012, 01, 01)				
3.4	Test set open time to a reverse period (2012, 01, 02, 2012, 01, 01)				
3.5	Test set open time with a date which does not exist (2012, 01, 01, 2012, 01, 32)				
3.6	Test set open time with a reverse date, like day, month and year. (01, 01, 2012, 02, 01, 2012)				
3.7	Test set open time with an alphabet (a, 01, 01, 2012, 01, 01)				
3.8	Test set open time with a date, like year, month and day. (2012, 01, 03, 2012, 01, 04)				
3.9	Test enter nothing in tariff blanks (, , , ,).				
3.10	Test set tariff with a free period and a 4-number input (0, 1, 1, 2 and 99.99)				
3.11	Test set tariff with a reverse price(5, 4, 3, 2 ,1)				
3.12	Test set tariff with a 5-number input (1, 2, 3, 4, 10000)				
3.13	Test set tariff with an alphabet (1, 2, 3, 4, a)				

Test case	Test description	Pass/ Fail	No. of Bugs	Bug #	Comments
Set up 5	Staff account is added to the system (ID is 00001, state is paid); Public account is added to the system (ID is 00002, state is paid); The remaining space is set to 2; The open time is set to 1900, 01, 01, 9999, 12, 31				Add to system for testing
5.1	Test staff enter when park is not full (ID is 00001)				
5.2	The remaining space is set to 0; Test staff enter when park is full. (ID is 00001)				
5.3	The account state of 00001 is set to unpaid; Test staff enters twice before exit. (ID is 00001)				
5.4	Test staff enter with wrong ID. (ID is 00003)				
5.5	Test staff enter when park is not full in open time (ID is 00002)				
5.6	The open time is set to 1900, 01, 01, 1900, 01, 01; Test staff enter when park is not full out of open time (ID is 00002)				

Test case	Test description	Pass/ Fail	No. of Bugs	Bug #	Comments
Set up 6	Account is added to system (ID is 00001, state is unpaid)				Add to system for testing
6.1	Test payment check (ID is 00001)				
6.2	Test payment check with wrong ID (ID is 00002)				
6.3	the account state of 00001 is paid; Test payment check with an ID which has not entered (ID is 00001, state is paid)				

Implement test

3.1. Automatic Test:

For the component level we mainly write the test in the implementation stage.
For the system level, the test code is as follows:

```
//Check the date
Date today=new Date();
boolean b=checkDay(today);
System.out.println("Today? "+b);

//judge whether the entrance is successful
pubTemp.card_number="" +ticketcount;
pubTemp.payflag=1;
JOptionPane.showMessageDialog(null, "Welcome, your ticket number is
"+pubTemp.card_number, "SUCCESS", JOptionPane.INFORMATION_MESSAGE);
System.out.println("Welcome"+pubTemp.accountNum);

//judge amount of public account
int check=0;
while (line!=null)
{
    check++;
    if(check>size)
    {
        Staff_AccounttempAccount=new Staff_Account(line);
        mylist3.add(tempAccount);
    }
    line=br.readLine();
}
size=mylist3.size();
System.out.println("Now the size is "+size);
Staff_Account a=mylist3.get(1);
System.out.println("The name is "+a.card_number);

//Calculate the days of parking
Date tempory=new Date();
int year=tempory.getYear();
int month=tempory.getMonth();
int day=tempory.getDate();
if(mylist3.get(htemp.accountlistnum).accountDate!=null)
{
    System.out.println("Calculate the days of parking.");
    year1= mylist3.get(htemp.accountlistnum).accountDate.getYear();
    month1= mylist3.get(htemp.accountlistnum).accountDate.getMonth();
    day1= mylist3.get(htemp.accountlistnum).accountDate.getDate();
    htemp.payment_time=tempory;
```

```

if(year==year1 &&month==month1 &&day==day1)
    s=0;
else
{   if(month-month1==0){
        intdayin=day-day1;
        s=1*dayin;
    }else{
OptionPane.showMessageDialog(null, "Sorry, your last month bill has problems,
\please ask operator for help.", "SORRY", JOptionPane.ERROR_MESSAGE);
    }
}

```

//Calculate the monthly charge of staff

```

htemp.charge=s;
if(htemp.type==true)
{
Staff_Accountfindaccount=mylist3.get(htemp.accountlistnum);
findaccount.accountCharge=findaccount.accountCharge+s;
System.out.println("Your account charge is "+findaccount.accountCharge);
findaccount.payflag=0;
}

```

//Report the usage of the car park

```

OptionPane.showMessageDialog(null, "You have paid \u00A3 "+paidMoney,
"Welcome", JOptionPane.INFORMATION_MESSAGE);
Car_Park.amount_of_coins=Car_Park.amount_of_coins+transhtemp.collection;
System.out.println("The park max is "+Car_Park.max_money);
System.out.println("The park coin is "+Car_Park.amount_of_coins);

```

//Compare the input with the stored ID and password of operator

```

FileReaderfr=newFileReader("OperatorList.txt");
BufferedReaderbr=newBufferedReader(fr);
String num=br.readLine();
intnumOfOP=Integer.parseInt(num);
while(numOfOP!=0){
String line1=br.readLine();
String line2=br.readLine();
System.out.println("line1 "+line1);
System.out.println("line2 "+line2);
System.out.println("ID "+ID);
System.out.println("Password "+Password);
if(ID.equals(line1) &&Password.equals(line2)){
    ID1 = 1;
}

```

```
    PW1 = 1;
    break;
}
else{
    ID1 = 0;
    PW1 = 0;
}
```

Perform Integration Test

Defect number: 003

Defect Title: Invalid account was entered

Test number: 1.3

Description: The inexistent account has been entered and this was not excepted.

Assigned to Component Engineer: Ge Tong

Raised by/assigned to Test Engineer: Liu Zhijiao

Defect number: 004

Defect Title: Invalid account with wrong password was given entered to the system.

Test number: 1.4

Description: Both account and password is invalid but the system can't distinguish it.

Assigned to Component Engineer: Zhang Tianshu

Raised by/assigned to Test Engineer: Liu Siyuan

Defect number: 005

Defect Title: Create new account for the ID existed in financial txt but not in system.

Test number: 2.2

Description: The system treat the ID existed in financial txt but not in system. as not exists in neither of two database.

Assigned to Component Engineer:Liu Yang

Raised by/assigned to Test Engineer: Zhao Chenyu

Defect number: 006

Defect Title: The wrong sequence time can be accepted by system.

Test number: 3.6

Description: The open time exists in the system, values are 2012, 01, 01, 2012, 01, 02.

New values, 01, 01, 2012, 02, 01, 2012 have been entered in the blanks of open time and the system accept the new open time.

Assigned to Component Engineer: Liu Jiawei

Raised by/assigned to Test Engineer: Ping Yao

Defect number: 007

Defect Title: Illegal characters were input as time successfully.

Test number: 3.7

Description: The operator input illegal characters into time blank and the system did not identify it.

Assigned to Component Engineer: Liu Zhijiao

Raised by/assigned to Test Engineer: Ge Tong

Defect number: 008

Defect Title: Irrational tariff was accepted by the system.

Test number: 3.12

Description: The data of irrational is abnormal but system still accepts it.

Assigned to Component Engineer: Liu Siyuan

Raised by/assigned to Test Engineer: Zhang Tianshu

Defect number: 009

Defect Title: Remaining space is ignored by the system.

Test number: 5.2

Description: There is no remain space but the system still opens the entrance barrier.

Assigned to Component Engineer: Zhao Chenyu

Raised by/assigned to Test Engineer: Liu Yang

Defect number: 010

Defect Title: Unpaid account could be allowed to enter in.

Test number: 5.3

Description: The account without paid cannot be identified by system and enter in successfully.

Assigned to Component Engineer: Ping Yao

Raised by/assigned to Test Engineer: Liu Jiawei

Defect number: 011

Defect Title: The system open the entrance barrier for public in invalid date.

Test number: 5.6

Description: The date of system isn't in open time, but the system still open car park for public.

Assigned to Component Engineer: Zhang Tianshu

Raised by/assigned to Test Engineer: Zhao Chenyu

Defect number: 012

Defect Title: The paid account is confirmed as available by system.

Test number: 6.3

Description: The state of an account is paid and this account ID has been entered. System confirms that the account state is available. It's unaccepted.

Assigned to Component Engineer: Ge Tong

Raised by/assigned to Test Engineer: Liu Siyuan