SELF-SERVICE TICKETING KIOSK FOR CINEMAS

Group: 99

Name:	QM	BUPT
Yu Liu	140924161	2014213404
Yujie Chen	140923418	2014213334
Yudong Wang	140924493	2014213397
Yitong Liu	140924150	2014213383
Yiwei Pei	140924275	2014213456
Yongguang Xie	140924736	2014213496

Summary of Responsibilities and Achievements

NAME	Yu Liu	POSITION	Group Leader
QM NUMBER	140924161	BUPT NUMBER	2014213404

Individual Members Contribution

As the leader of the group, I participated in all parts of the project.

At the beginning of the project, I analyzed the project with my team members and I identified the purpose and procedure of the project. Then I allocated the work to all the members.

During the project, I organized the group members to discuss and report the proceedings. And I actively collected requirements by background reading, interviewing with an administrator of a cinema and movie fans, carrying out a questionnaire survey. Then I optimized and sorted out our backlog. What's more, I also designed interface of our system by drawing prototype. In addition, I did programming and achieved the interface of screen's layout and movie's time table.

In the report, I was mainly responsible for the *Requirement* section. I used fact-finding techniques to find requirements and analyzed them. Then I also drew physical interface of our system. After that, I also consolidated the report.

Self-appraise and Assess for outcome of our coursework

I think I did the job as leader perfectly. I benefited a lot from the project.

Firstly, I had a deeper understanding of agile development method, which was very helpful for me to do other software development projects in the future.

Secondly, I also developed my programming ability, particularly Java programming.

Thirdly, by becoming the leader, I learned how to organize my group members and how to communicate with my group members.

NAME	Yujie Chen	POSITION	Designer, Analyst
QM NUMBER	140923418	BUPT NUMBER	2014213334

Individual Members Contribution

In the project, I was mainly responsible for designing and analyzing classes of our system. From the documentation of requirement, I knew the elements of our system. In the designing process, I identified entity classes, boundary classes and control classes of our system. Then I used attributes to describe theses classes and used methods to describe functionality of these classes. What's more, in the analyzing process, I modified some classes I have designed and drew the class diagram. In addition, I did some programming

and finished the interface of movie information including brief information and detailed information. As for the report, I mainly wrote the *Design and Analyze* section. I drew the class diagram, analyzed the adaptability of our system, clarified classes of the system and so on.
Self-appraise and Assess for outcome of our coursework I think I did good job in the project. I perfectly finished all tasks including designing classes, doing programming assigned by group leader. What's more, I learned a lot from the project. I understood the agile development method, which was very important. And I also learned how to design and analyze classes of system. I hope I can have chance to do some other software development projects in the future.

NAME	Yiwei Pei	POSITION	Software Tester
QM NUMBER	140924275	BUPT NUMBER	2014213456
Individual Members Contribution In the project, my work is mainly to test our system. I use White-box testing, Black-box testing, Regression testing, Junit to test our system so that our system is error free. If I detected some errors in our system, I would tell relative group members to fix it. What's more, I gave some advice to my group members and discussed with my partners in meetings. Besides, I also did some programming. I finished the functionality of our system that customers can buy different types of movie tickets. As for the report, I mainly finished the *Test* part. I use different testing techniques including White-box testing, Black-box testing in order to test our code and functionality of our system. What's more, I set some parameters which can be used to assess our system.			
Self-appraise and Assess for outcome of our coursework I tried my best in the team work. In the group discussion, I always gave advice to other group members in order to help them improve their work. And I successfully finished my work. I learned many different techniques to test our system, which was helpful for me to become software tester in the future.			

NAME	Yongguang Xie	POSITION	Programmer
QM NUMBER	140924736	BUPT NUMBER	2014213496
Individual Members Contribution In the project, my main work is to do programming. I used xml file to store some information related to entities such as movie, screen in our system. And I also wrote a parser, which could be used to transfer the data stored in xml files to java files. And I also achieved other functionalities of our system including printing tickets, creating statistic report, selecting seats. Because our group used pair programming, I often discussed with another group member who did programming with me.			

As for the report, I wrote the *Implementation* section. I described the several subsystems of our system and designed build plan.
Self-appraise and Assess for outcome of our coursework I was good at coding, so I achieved several functionalities of our system. What's more, before coding, I wrote a build plan which helped me to decide the order of completion of functionality of our system. I think the project was a very good opportunity for me to practice my programming skills. And our software is very good because we add some new features to our software.

NAME	Yitong Liu	POSITION	Planner
QM NUMBER	140924150	BUPT NUMBER	2014213383
Individual Members Contribution In the project, my main work is to analyze our project and break our project into many different tasks. I knew how to obey the rules of agile method. And I also determined milestones, deliverables, constraints and parameters of our system after discussing with my group members. Besides, I also did programming. I mainly achieved the functionality that customers can pay for their tickets by using bank system. As for the report, I was mainly responsible for *Project Management* section. I drew staff allocation graph and made task chart. And I also analyzed each sprint cycle.			
Self-appraise and Assess for outcome of our coursework I tried my best in the group work. I always discussed with my group members and gave them some advice. And I finished my work very well. I divided the project into many different tasks efficiently, and each group member had some tasks to do. In addition, I leaned some programming skills and organizational skills from the project. And I am satisfied with our software.			

NAME	Yudong Wang	POSITION	Programmer
QM NUMBER	140924493	BUPT NUMBER	2014213397
Individual Members Contribution In the project, I was mainly responsible for the implementation of the system. Because of pair programming, Yongguang Xie and I used one computer to do programming. I mainly achieved the functionality that administrator can update film information. What's more, I also checked if our code had any errors when I did programming with Yongguang Xie. As for the report, I mainly wrote the *Implementation* section. I mainly explained how our system operates and the experience of pair programming.			
Self-appraise and Assess for outcome of our coursework I think I did good job in the project. I finished several functionalities of our system so that our system had more features. And I also learned many knowledges related to agile method from the project. We tried best to develop our software, I think our software was very good.			

Content

1.Introduction

In the project, we used agile method to develop a self-service ticketing kiosk for cinemas. We strictly follow rules of agile method, such as pair programming, adaptation to requirements, generation of many versions of product and so on. We divided our project into five stages: Project Management, Requirement Engineering, Design and Analyze, Implementation, Test. Because of agile method which focuses on the speed of software development and generation of different versions of products for different iterations, our group did not completely follow five stages sequentially, instead we conducted different stages of our projects concurrently. And even if each group member had special tasks, such as collecting requirements, designing and analyzing classes of our system, testing software and so on, we helped with each other and did other tasks. For example, we used pair programming in the project and except for programmers, everyone else in our group should do programming together with another group member to finish some functionalities of our system.
We also divide our report into five sections which are similar with stages in the

project. *Project Management* section mainly focus on how to make plans for our project and how to manage our group and project. We use Scrum approach in the project. *Requirement Engineering* section is mainly related to how we use fact finding techniques to collect requirements. *Design and Analyze* section focus on how to design and analyze classes of our system and how to generate class diagram of our system. *Implementation* section is related to how to implement our system. *Test* section is related to how to detect errors in our system.

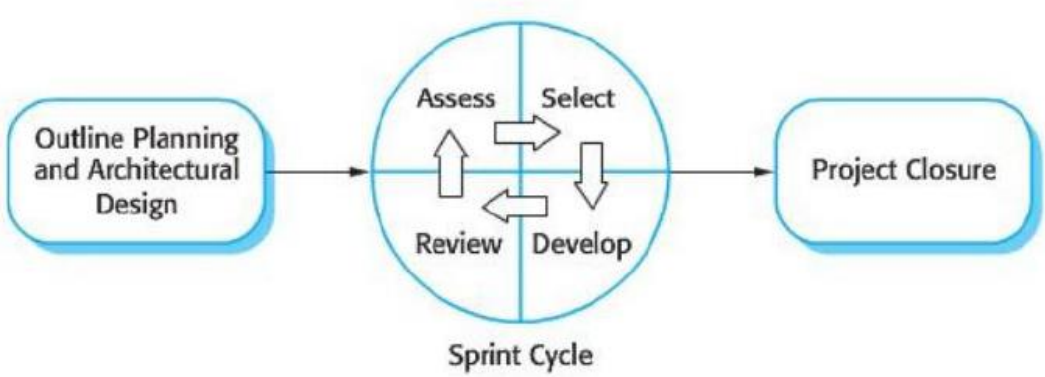
2.Project Management

2.1 Introduction

The goal of the project is to develop the software used in self-service ticketing kiosk for cinemas, which provides convenience for customers to buy the tickets and for administrators to deal with the films' information and statistic report. We use Scrum approach to set out our project management.

2.2 Project Management Techniques

Agile is a set of best practices in software development based on Scrum, Extreme Programming and Lean. Our group used Scrum approach to set out project management.



The initial phase is outline planning. During this stage, our group mainly identified the objectives of our project and made different schedules such as quality plan, staff development plan, etc.
Then we had many sprint cycles. Each sprint cycle was two weeks. For each sprint cycle, we generated a new version of our software which had some new features compared with last version of software. At the beginning of each sprint cycle, the group leader organized Sprint Planning Meeting. In the meeting, we

discussed about what need to do and how to do it. Then, group leader assigned task to each member. We tried our best to complete the tasks during this sprint cycle. We also had many brief daily meetings during sprint cycle. We mainly discussed about the progress of the task. If a member had a trouble of the task, he could bring it up and others would help him. At the end of each sprint cycle, we had wrap-up meeting, at this meeting, each member needed to show his result and summarized the problems in this Sprint cycle and avoided such problems in next cycle.

Final phase of our project is project closure. We wrapped up the products and completed required documentation such as report of the project, user manual.

2.3 Constraint

Project duration – 6 weeks
Each Sprint cycle – 2 weeks
Human Resource – 6 people in our group
Software Resource – only Java JDK 7, Text (txt), CSV or XML file format, no network programming, no database implementation
Development Method – Agile method
Scrum master: Liu Yu

2.4 Sprint Backlog

For each sprint cycle, we chose different requirements based on priority of requirements to implement. Because project duration was six weeks and each sprint cycle was two weeks, we had three sprint cycles.
At the beginning of each sprint cycle, we held a sprint planning meeting, we chose some important requirements. Then we wrote the code according to the requirements. During the sprint cycle, we also had daily meetings.
In the end of each sprint cycle, we held a wrap-up meeting and we reviewed previous work, and we were sure that the program could run perfectly.

Sprint cycle 1

Requirement	Description
brief movie information	customers can obtain brief information of each movie such as actor, director.

movie plot	customers can obtain plot of each movie
movie show time	customers can obtain show time of each movie
layout of each screen	layout of each screen can be displayed
log in the system by administrator	administrator can log in the system

In sprint cycle 1, we wrote the code according to the requirements that display information of films and select films and seats. We also achieved the functionality that administrator can log in the system.

Sprint cycle 2

Requirement	Description
label of each seat	each seat in screen should have a label
identify which seat has been chose	customers can know which seat has been chosen by others
cancel seat	customers can cancel the seat they have chosen
information about movie tickets	different types of movie tickets can be displayed
cancel transaction	customers can cancel the transaction
authentication of bank account	the bank account should be authenticated by bank system
total price of tickets	total price of tickets can be displayed
confirm transaction	confirmation message should be displayed

In sprint cycle 2, we wrote the code according to the requirements that customers can select seats and buy different types of tickets.

Sprint cycle 3

Requirement	Description
print ticket	movie tickets should be printed
change film information and film show time	administrator can change film information and film show time
classification of movie tickets	classify movie tickets in the statistical report
sales of each movie	compute total sales of each movie
total sales on the day	compute total sales of all movies
statistic report is sent to administrator	statistic report is sent to administrator automatically on the day

In sprint cycle 3, we wrote the code according to the requirements that the total sales of all movies and sales of each movie should be stored in the statistic report which should be sent to administrators automatically.

2.5 Establish Milestones and Deliverables

We defined sixteen activities as milestones. And among milestones, eight milestones can be identified as deliverables because these eight results of milestones can be delivered to customers. Milestones and Deliverables are

shown in task chart.

2.6 Task Chart

Appendix---Task Chart

2.7 Staff Allocation and Gantt Chart

Appendix---Staff Allocation and Gantt Chart

2.8 Decision Making

At the beginning of project, we setup WECHAT and QQ group in order to communicate with each other and send files. And for each sprint cycle, we had sprint planning meeting, daily meetings, wrap-up meeting. In these meetings, everyone in the team knew what was going on and we made decisions about what we should do next and how to deal with some problems in the project.

2.9 Adapting to Changes

Because we use agile method to develop our software, our project focus on adapting changes in requirements. For each sprint cycle, we added some new features to our product and generated a new version of our product.

2.10 Group photo

Appendix---Group Photo

3.Requirement Engineering

3.1 Finding Techniques

3.1.1 Background Reading

In order to have a deeper understanding of the system we tried our best to do

some reading on QMUL, and we talked with each other, took some notes. From the background reading, we knew that the main task of this project was to achieve a self-service ticketing kiosk for cinemas. We found that there are three kinds of users using the system, which are customer, administrator, ticket inspector.

Role	*Operation*
Customer	select a film, then select an available show time, choose available seat(s), choose ticket type and number, and finally confirm.
Administrator	load film information and film show time on the day to the kiosk and check the statistic report of the sales of the day.
Ticket inspector	check the ticket by the gate.

What's more, we found many detailed information by background reading such as the layout of each screen, the format of ticket, the content of statistic report and so on. It's essential to know detailed information related to our system before we design it.

3.1.2 Interview

To conclude the specific information, we separately interviewed customers and administrator in a cinema. We designed many questions in terms of what we want to know. We obtained many useful information from interviews and the information could help us identify features of our system.

a. The interview for customers

We interviewed twenty customers in a cinema one by one. We asked them some questions such as "Do you think current self-service ticketing kiosk for cinemas is good enough?" "Do you think If some features can be added to current self-service ticketing kiosk for cinemas?". For same question, different customers had different views. But we found that what customers care most was that the kiosk should be easy to use and film information should be updated as soon as possible. After interviewing customers, we concluded that our system should have user-friendly interface and is convenient for customers to use.

Sample Question
Do you think current self-service ticketing kiosk for cinemas is good enough?
What problems the existing self-service ticketing kiosk for cinemas have?
Do you think If some features can be added to kiosk?
Do you want to use your bank card to buy tickets?
Do you want the self-service ticketing kiosk to sell different types of tickets?

b. The interview for administrators

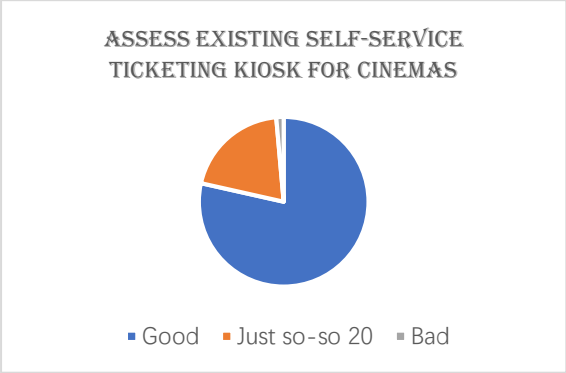
We also interviewed the administrator in a cinema. We asked him some

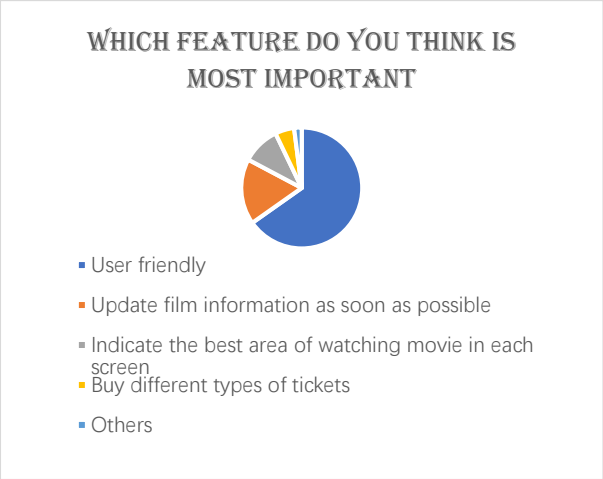
questions such as “What detailed information should be stored in statistic report?” ”Is it necessary to log in the system before updating film information?”. The administrator answered these questions patiently. We found that what administrator care most was that the kiosk should attract customers and statistic report should include all detailed information of sales so that administrator can know which movie is most popular and how to increase sales in the future.

Sample Question
What features the self-service ticketing kiosk for cinemas should have?
Which feature do you think is most important?
Is it necessary to log in the system before updating film information?
What detailed information should be stored in statistic report?
What features the existing self-service ticketing kiosk for cinemas can be added?

3.1.3 Questionnaire

We designed a questionnaire which has ten questions. These questions were mainly related to current self-service ticketing kiosk for cinemas and some additional functions the kiosk should offer. We gathered 153 questionnaires online and analyzed them.





After we analyzed the results of questionnaire survey, we found that many people think that user friendly is the important feature. When we design our system, the interface of our system should be convenient for users to use.

3.1.4 Observation

Observing the current system can provide us with a better understanding of the system to develop. When customers used the existing kiosk, we observed how the system performed. And we found some problems in existing self-service ticketing for kiosk.

Problem	Solution

Only one type of ticket	Add different types of tickets to kiosk
Customers can only use cash or online payment platform to pay for movie tickets	The system can connect to the bank system so that customers have a chance to pay for movie tickets using bank card
The label of each seat is not displayed in the screen	Display label of each seat

3.1.5 Document or Record Sampling

This is a specialist form of observation which focuses on collecting copies of documentation, obtaining details and identifying patterns in requirements for the system. And it can generate non-functional requirements.

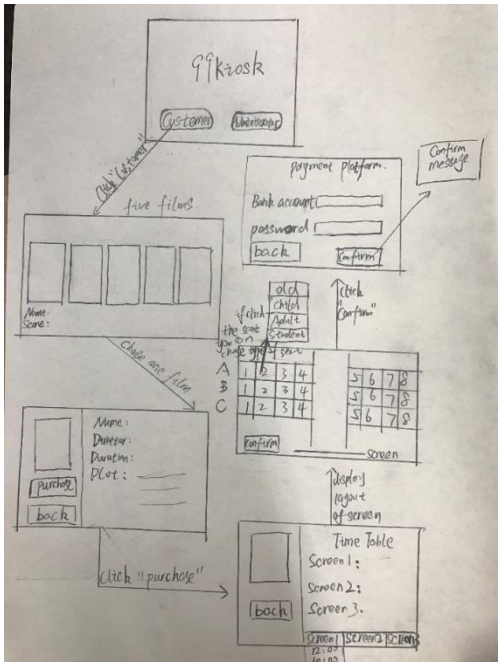
Non-functional requirements	
Language	The system should be in English
Easy to use	A person who didn’ t use our system before can understand how to use our system in five minutes
Adapt to changes	The system should adapt to any requirement changes without affecting other functions
Generic	Many film companies can use our system by adjusting just small parts of our system
Accuracy	The information about film and time table should be accurate with no error

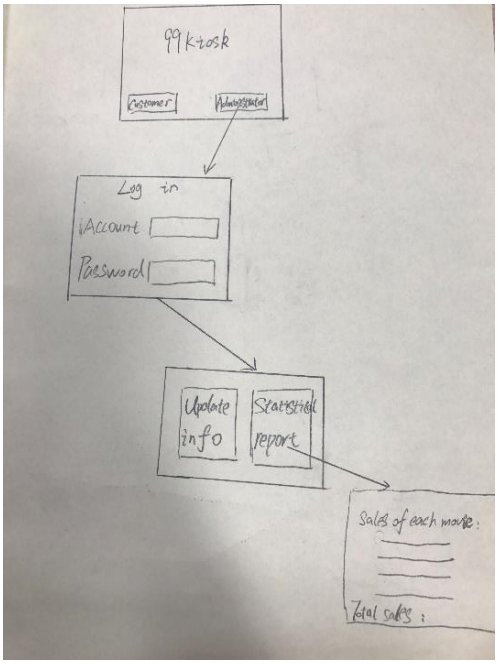
3.2 Software Prototype

Logical interface: We can identify basic elements of interface of our system.

Appendix---Logical interface

Physical interface:





From the above picture, the left picture is customer mode of our system. The right picture is administrator mode.

3.3 Change of product backlog

After Requirement Engineering, our group held a group meeting and we analyzed the requirements collected by using fact finding techniques. And then we had a discussion to determine which requirement could be placed in backlog. We also assigned priority to different requirements based on the importance of requirement. Finally, we finished our backlog.

However, as the development of project, some requirements in backlog changed. And some requirements were removed from backlog and some new requirements were added to backlog.

Requirement Name	Original	New
Brief information of film	Display a brief introduction of each film	Only display name, score of each film

With the development of project, we found that customers had much interest in the score of each movie which was assessed by expert in the movie field. And if customers want to have a deeper understanding of the movie, they can just click relative movie and then they can obtain detailed information of the movie. It's not necessary to display brief introduction of each movie before customers can obtain more detailed information of movies.

Requirement Name	Original	New
Movie show time	Display complete time table of all movies when customers have selected one film	Only display the time table of the movie selected by customers

We found that displaying whole time table of all movies after customers have selected a film is not user friendly. We decided to only display the time table of the movie selected by customers.

Requirement Name	Original	Remove the requirement from backlog
Best area of each screen	Display the best area of watching movies in each screen	

Because each screen is not big, it's not necessary to display the best area of each screen.

Requirement Name	Original	New
Sales on the day	Sales of all movies on the day should be stored in the statistic report	Except for sales of all movies, sales of each movie also should be calculated and stored in statistic report

We found that administrator wants more detailed information in statistic report including sales of each movie.

Requirement Name	New
Update score of each film	Because the score of each film changes rapidly, administrator should often update score of films

This is the new requirement we add, which is convenient for administrator to update score of films.

3.4 Iteration and Estimation of the stories

We used Scrum approach to develop our project, and we had three sprint cycles(iteration), each sprint cycle was two weeks. For each sprint cycle, we selected some requirements based on priority of requirements and evaluation of requirements by programmers.
And we used story points to assess different stories.

Sprint cycle 1

Requirement	Description	Story point	Expected Duration(days)	Actual Duration(days)
brief movie information	customers can obtain brief information of each movie such as actor, director.	3	1	1
movie plot	customers can obtain plot of each movie	3	1	1
movie show time	customers can obtain show time of each movie	5	2	2
layout of each screen	layout of each screen can be displayed	8	6	8

log in the system by administrator	administrator can log in the system	8	4	2

Sprint cycle 2

Requirement	Description	Story point	Expected Duration(days)	Actual Duration(days)
label of each seat	each seat in screen should have a label	3	2	2
identify which seat has been chose	customers can know which seat has been chosen by others	8	5	7
cancel seat	customers can cancel the seat they have chosen	5	4	3
information about movie tickets	different types of movie tickets can be displayed	5	3	2

Sprint cycle 3

Requirement	Description	Story point	Expected Duration(days)	Actual Duration(days)
print ticket	movie tickets should be printed	3	1	3
change film information and film show time	administrator can change film information and film show time	8	7	6
classification of movie tickets	classify movie tickets in the statistical report	3	3	2
sales of each movie	compute total sales of each movie	3	1	1
total sales on the day	compute total sales of all movies	3	1	1
statistic report is sent to administrator	statistic report is sent to administrator automatically on the day	3	1	1

With larger story points, the user story is more difficult to implement. If a user story's story point is greater than 13, the user story is an epic story. And the epic story should be divided into many smaller user stories.
At the beginning of each sprint cycle, we choose user stories based on priority and story points. All three sprint cycles have same total story points which means that workload of each sprint cycle is same.

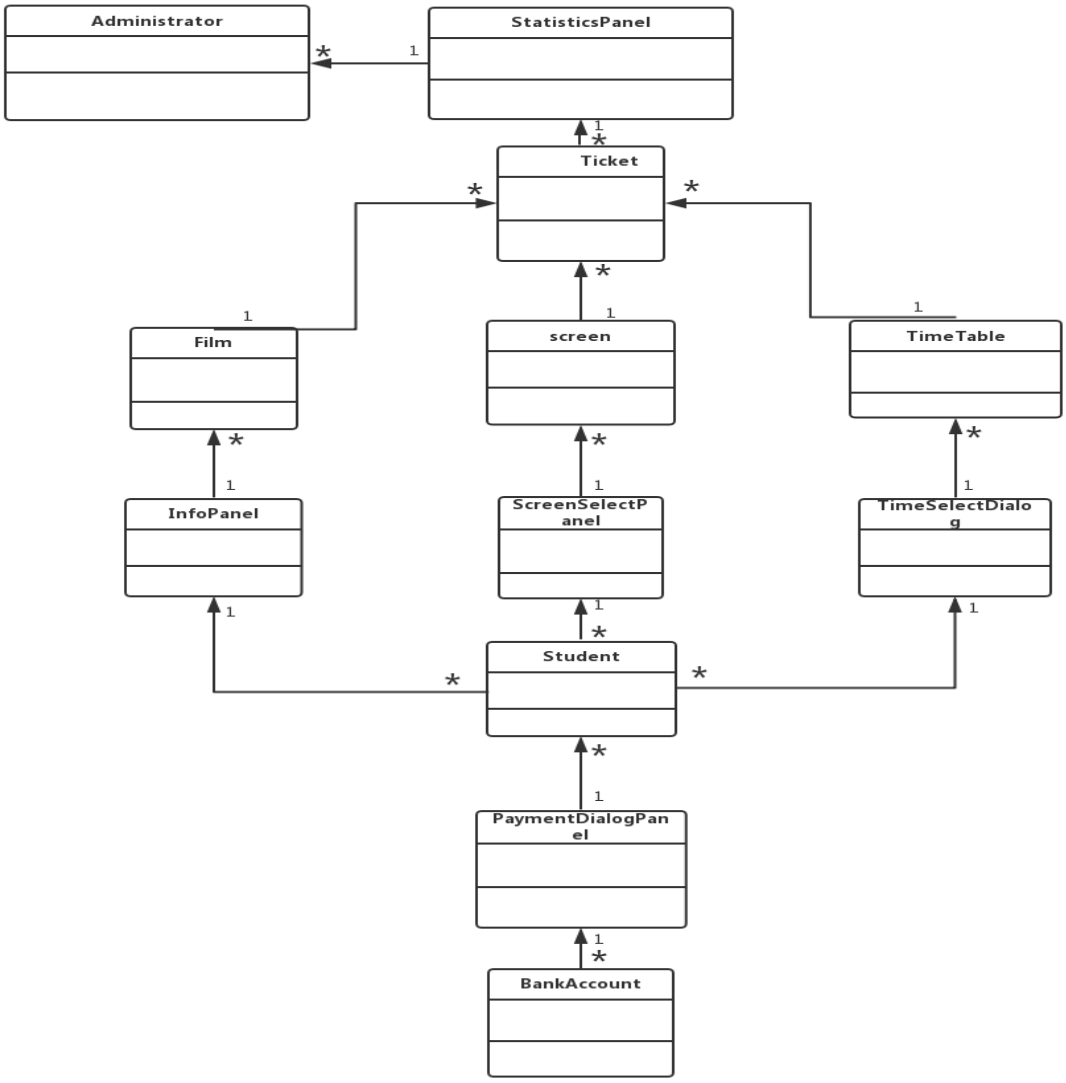
4. Design and Analyze

4.1 Identify Entity class, Boundary class, Control class

Because there are many classes in our system and we classify them into three types of classes (Entity class, Boundary class, Control class), we place them in the appendix.

Appendix---Entity class, Boundary class, Control class

4.2 Identify main class relationships and class diagram



Appendix---Complete Class Diagram

4.3 Component Design

This is used to transform structural elements of software architecture into a procedural description of software components. Firstly, we identify all design classes and find corresponding problems and infrastructure domain. Secondly, we elaborate all design classes that are reusable components. Then we describe persistent data and identify the class which we need to manage them.

Finally, the structural elements represent their corresponding components.

4.4 Architecture Design

We use layer architecture design and divided all class into 4 layers: File layer, IO layer, Control layer and UI layer. File layer contains all information about customers and managers, such as student id and manager id. We can use IO layer to read data from the file and modify data to the file easily. Control layer contains all entity classes and control classes, which are related to all the process of calculation and data structure. UI layer contains all interfaces that specify in use, the users can use the panels and buttons with the help of UI layer, which makes the system more user-friendly and convenient. There are two kinds of UI layer in our system, manager-UI and customer-UI. Manager-UI is used by administrator for managing and statistic the result of selling. Customer-UI is used by some customers to choose the film, screen, seat and pay for the film.

Appendix---Architecture Design

4.5 Re-usability of system components

Firstly, we use SRP principle, which means that some classes are designed to have only one responsibility in the system. In the project, we make classification of the class clear and write every class with single responsibility in order to make them easy to be called and used. So classes of our system are cohesive. Every class in our system has a single specific functionality. These classes can be easily reused according to their responsibilities. For example, WelcomePanle.java is used to generate a welcoming interface, which can be reused in other systems such as online shopping platform.

Secondly, we write our programs with many independent classes. There are few dependences between classes. In each class, we write its own function to satisfy our demand of reusing. In the future if the user needs to modify the functions in the class, he should change the function only but not the whole program. For example, we define the shape and color of button in NormalButton.java and SpecialButton.java. When we need to use the button, we can use methods in these two classes. If we want to change shape of button, we just need to adjust methods in NormalButton.java or SpecialButton.java.

Thirdly, without database we store our data and information in the XML files. The programmer can call and use the data stored in XML file for many times

which is flexible and convenient for programming. And the data stored in XML files is more easy to modify, which is convenient for programmer to implement and test the programming. For example, we store the film information in a XML file. When we run our software, the IO find the film information stored in the XML file. If we want to change the film information, we can change the XML files without modifying the program. We also store the information of seat, student, timetable and bank account in different XML files. So we can change these information without modifying codes.

Our system is very flexible and can be used by different users, which means it can adapt to all kinds of conditions.

4.6 Adaptability to changes

Because we use agile method to develop our system, our system has good ability to adapt to changes. And we use SRP principle to develop our system. So if our system meets some changes in requirements, we can just change relative classes without changing other classes.

What's more, the administrator can change film information and time table information at any time in our system, because the movies showing in the theater is always changing, so the system can be used forever. And in our system, we have administrator mode, in that mode, administrator has opportunity to change the score of each movie because score of movies is always changing, and it's very convenient for administrator to update scores of films by using our system.

Secondly, the manager can change the structure of screen by changing the information stored in the data structure.

Thirdly, our system can be used in many different movie companies. Even if different companies have different logos and films, they can just change some specific classes in our system and then the system will be suitable to that company.

4.7 Design Principle

4.7.1 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, which can

lead to highly cohesive software. In the project, we strictly obey SRP principle. Every class in our system has single responsibility.

For example, in the system, there are two classes called Film and TimeTable, both of them have some common information, such as film time and film id. However, the responsibilities of them are different, and they separately focus on one responsibility. Film focuses on the basic information of a movie, and timetable combines the information about film and time, screen and so on.

Our code obeys SRP.

4.7.2 Open-Closed Principle (OCP)

This principle means our software modules (classes, methods) should be open for extension and closed for modification. In our work, there is no need to change the code of module if you want to do extension or modification. If new features should be added to our system, existing code doesn't need to be modified. We just need to add some codes in specific classes. To obey this principle, we make use of it in a more specialist way by passing in parameters or through inheritance.

Many methods in classes need parameter. For example, in Film class there is a method called setFilmDuration(Duation). Different films can use this method to indicate its own duration by passing parameter. Our code obeys OCP.

4.7.3 Don' t Repeat yourself(DRY)

We use DRY principle because we don't want to 'cut and paste' to write the similar code.

For example, in the system, there are two classes, called NormalButton and SpecialButton. Because in each panel, we should have some buttons. And these buttons can be created by methods in NormalButton or SpecialButton.We don't need to write similar code for these buttons. It's very convenient to call these methods. However, we don't use many abstract classes. In the future, we want to transform existing classes to abstract classes.

4.7.4 Dependency-Inversion Principle(DIP)

DIP principle can be stated as high-level modules should not depend on low-level modules, they both should depend on abstractions. In the current system, we use some abstract methods or abstract classes existing in system library, but we don't create many abstract classes. In the future, we hope to use more

abstractions in our system and change our code in order to obey DIP completely.

4.7.5 Interface-Segregation(ISP)

The ISP can be expressed as “Clients should not be forced to depend on methods they do not use”. We should not let our clients depend on methods they do not use. We break our big interface into small and more specific interfaces so that clients will only know the methods they are interested in. When we want to add new information or check former information, we only have simple and convenient interface rather than big interface which is difficult to read.
For example, in our system, every panel have their own interface, so if the user wants to operate or modify one panel, they just need to recognize and manipulate the certain interface without affecting others. We strictly obey ISP.

4.7.6 Liskov Substitution Principle (LSP)

The problem of Liskov Substitution Principle is that a subclass can override a method with one which can do anything, so long as it has the same signature. The principle says that methods should not be overridden in a way that changes assumptions of their behavior.
Frankly, it’s difficult to obey LSP. Because in our system, many child classes have overridden the methods in parent classes. In the future, we hope to refactor our code in order to obey LSP strictly.

5. Implementation and Test

5.1 Implication Strategy

We map our design to code. In our system, we have several subsystems such as administrator mode, customer mode, read or write files and so on. We finish these subsystems one by one. Because we use JAVA to do programming, we define classes and methods and we use references to represent relationships between classes.

5.2 Build Plan

To make sure the whole system is easy to manage and be successful, we had set the program to different states and make the program more complex step

by step. Each build adds functionality to the previous one and contains a few new refined components so that to be manageable and has few integration and problems. And because we have three sprint cycles, we have three builds. Each build will generate a version of our software.

BUILD	Description
1	Show the information of the movies and select seats in screen
2	Pay for tickets by using bank system and print tickets
3	Create statistic report and update film information by administrator

5.3 Pair Programming

In the project, we use pair programming. There are many advantages of pair programming such as cultivating ownership of code, detecting errors in code efficiently. Everyone in our group participates in programming. When two group members sit together and use one computer to do programming, they can discuss with each other and propose new ideas. For example, in order to achieve a function two people in pair programming may have different methods. And they will discuss with each other, finally they will choose a better method to achieve the function. Pair programming is very beneficial for developing software.

5.4 Test

5.4.1 Test Strategy

– 40% of the tests will be automated.
- The remainder will be manual.
- This is to check if the user interface is correct.
- Each subject use case should be tested for its normal flow and behavior and also for two alternative flows.
- Make sure that they cover incorrect user input.

– Success criteria - 83% of test cases passed.
– No high priority defects unresolved.

5.4.2 Test Techniques

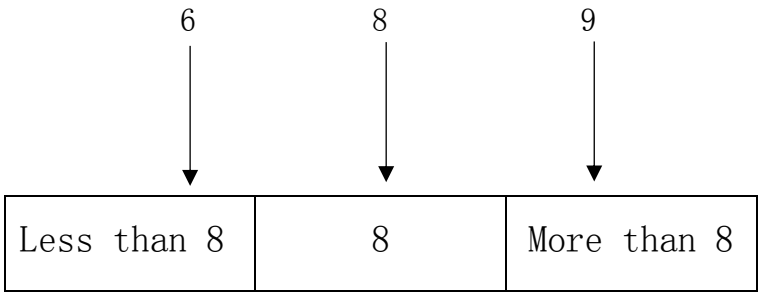
Aiming at demonstrating that software meets its requirements of both developer and customer and exposing the defects. Overall, the goal of testing is to build the confidence and make the software good enough for operational use, we choose some test techniques including Black-box testing, White-box testing, Regression testing, Junit testing.

Black-box testing

Black-box testing mainly focus on the functional requirements. It is a technique to attempt to find errors, such as interface errors, behavior errors and so on. In other words, we test our software according to the output but we do not care how the program realizes it.

Partition testing

Partition testing is used in both component testing and system testing.
Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each class member. Test cases should be chosen from each partition. We use partition testing to test all user inputs. Take the bank account input as an example.
We assume that bank account has 8 digits. We divide the input range into three parts.



We separately input 6 digits, 8 digits, 9 digits, and we find that our system can check the error. If we input 6 digits or 9 digits, the system can generate an error message. If we input 8 digits and relative password is correct, the confirmation message can be generated.

Scenario-based testing

Scenario-based testing is used in system testing. It is a test based on a hypothetical story used to help a person think through a complex problem or system. Observer can check that the requirement has been satisfied. We check our system if meets all requirements in backlog.

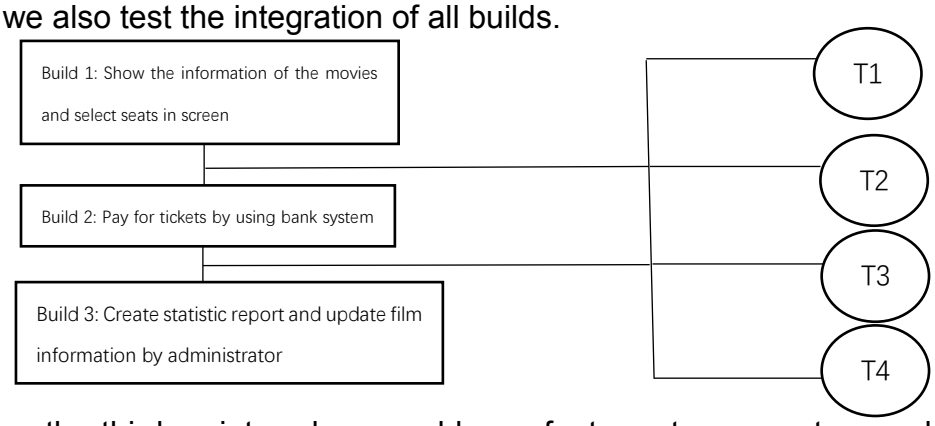
Appendix---Partition testing and Scenario-based testing

White-box testing

White-box testing is a method of testing the internal program logic. as opposed to its functionality (i.e. black-box testing). Its goal is to ensure that all statements and conditions have been executed at least once. In the project, we use basic path testing and looping testing to test our code.

Regression testing

Regression testing is applied at the integration level of the software testing process. We have three builds. At the end of each sprint cycle, we test all builds and we also test the integration of all builds.



During the third sprint cycle, we add new features to our system and we test Build 1, Build 2, Build 3 and integration of three builds.

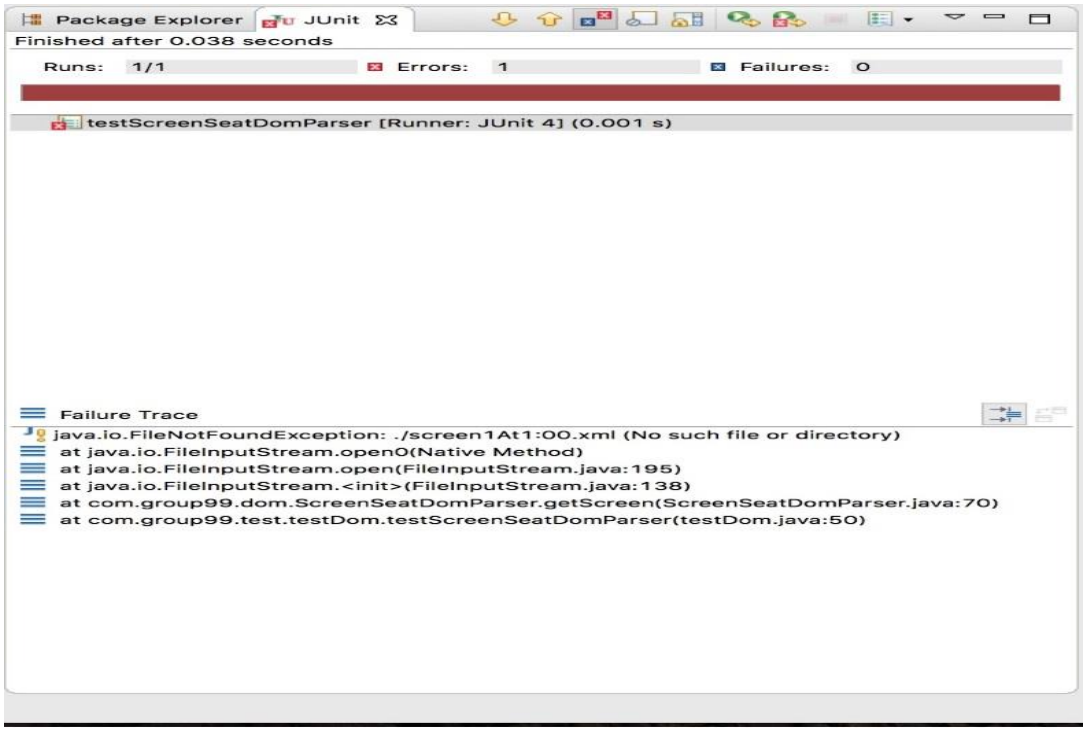
5.4.3 The using of TDD

TDD is a simple, short-cycled mechanism used for writing tests prior to complete the production code. At first, we write a specification, in code and in the form of a unit test. Then, we demonstrate test failure, and write code to meet the specification. Finally, what we should do is to refactor the code, which ensures that the system still has an optimally clean code based on running all tests within the entire system. Junit is a simple unit-testing framework for supporting TDD.

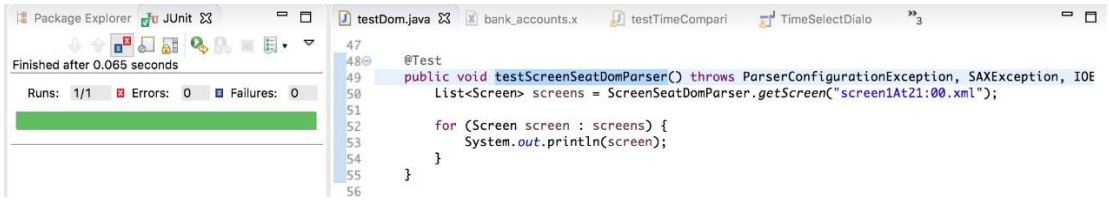
```
@Test
public void testScreenSeatDomParser() throws ParserConfigurationException, SAXException, IOException {
    List<Screen> screens = ScreenSeatDomParser.getScreen("screen1At1:00.xml");

    for (Screen screen : screens) {
        System.out.println(screen);
    }
}
```

Assertion failure:



Assertion success:



5.4.4 Test case

Appendix---test cases

5.4.5 Evaluation of test

We use different techniques to test our system. And we implement 13 test cases to test, and the test results show that our system has a high mistake tolerance rate, which indicates that the system we designed is robust and reliable.

6. Conclusion

Our group 99 completes the process of the software engineering in six

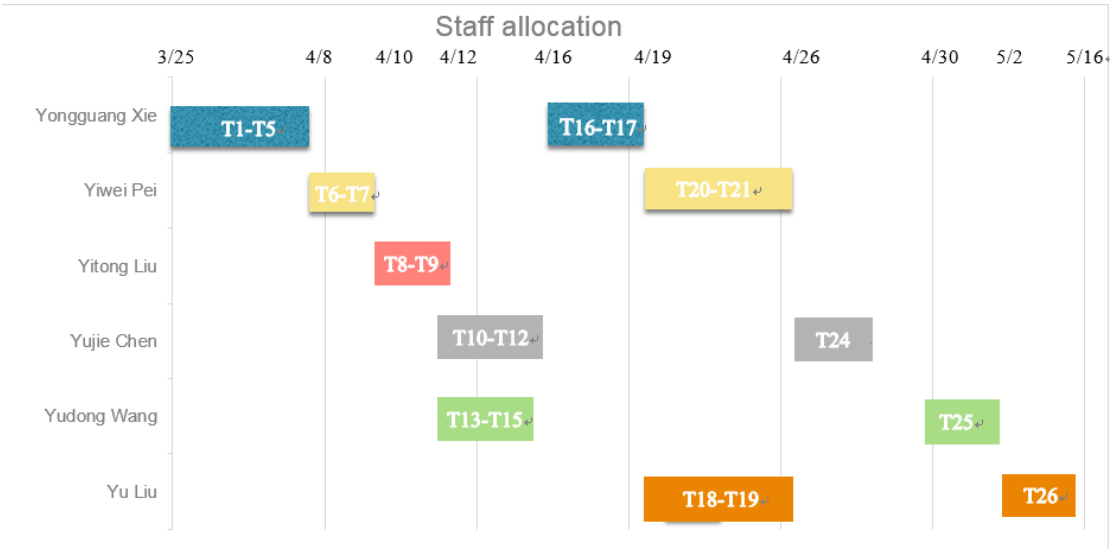
weeks. Besides, our group achieves the goal of designing and developing the Kiosk system by strictly following the agile method. Scrum approach is also in cluded.
We discuss the coursework handout in detail and outline the functionality of sy stem. The concurrent pair programming, the regular group meetings during S print cycle and the help delivered between each other make every member pa rticipate in the coursework efficiently. All the group members try their best to w ork on the five stages and solve the problems.
Finally, our system not only realizes the basic function such as the choosing t he movies, choosing the screen and seats and buying different types of ticket s, but also helps the administrator log in and then update film information.
We all gain much experience from this coursework and learn lots of skills.

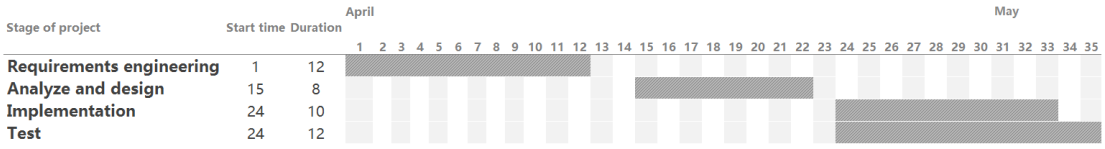
7. Reference

1. RobertC. Martin，Agile Software Development: Principles, Patterns and Practices
2. Crawl D., Altintas I. (2008), A Provenance-Based Fault Tolerance Mechanism for Scientific Workflows, 《Lecture Notes in Computer Science》
3. Chapter 3,4,6,7,8,22 - “Software Engineering” textbook by Ian Sommerville
4. Chapter 4,5 - “Head First Object Oriented Analysis & Design” textbook by Brett McLaughlin et al
5. Agile Java™: Crafting Code with Test-Driven Development, Jeff Langr
6. RN Charette(1989). Software engineering risk analysis and management

Appendix

Appendix---Staff Allocation and Gantt Chart





Return

Appendix---Group Photo

Pair programming Group Meetings

RETURN

Appendix---Task Chart

Task	Duration(days)	Dependence	Milestones	Deliverables
Requirement Engineering				
T1. Background reading	2 days			
T2. Carry out a questionnaire survey	2 days			
T3. Interview with students and movie fans	2 days		M1	
T4. Find users of our system	1 days	T1,T2,T3(M1)		
T5. Find functional requirements and non-functional requirements	3 days	T1,T2,T3(M1)	M2	
T6. Create product backlog	2 days	T6(M2)	M3	Requirement backlog
T7. Estimate stories	1 days	T7(M3)		
T8 Logical interface	1 days			
T9 Physical interface	1 days	T9	M4	Interface prototype
Analyze and Design				
T10 Identify Entity class, Boundary class, Control class	2 days		M5	
T11 Identify attributes and operations	2 days	T11(M5)		
T12 Identify class relationships	1 days	T11(M5)	M6	
T13 Add constraints	1 days	T11(M5)		
T14 Draw concept diagram	2 days	T13(M6)	M7	
Implementation				
T15 Build plan	2 days	T15(M7)	M8	
T16 Choose movies and show time.	1 days		M9	Code, Javadoc
T17 Design layout of each screen	3 days		M10	Code, Javadoc
T18 Choose seat	5 days	T18(M10)	M11	Code, Javadoc
T19 Buy movie tickets	3 days	T19(M11)	M12	Code, Javadoc
T20 Print tickets	3 days	T20(M12)	M13	Code, Javadoc
T21 Create statistical report	5 days		M14	Code, Javadoc
Test				

T22 TDD test	2 days		M15	
T23 Write box testing	2 days			
T24 Black box testing	2 days			
T25 Integration testing	2 days			
T26 improve the software	3 days		M16	Improved software

RETURN

Appendix---Entity class, Boundary class, Control class

Entity class

Name	Description
Film.java	Identify each film, containing all kinds of information about each film, such as id, name, time and so on.
TimeTable.java	Identify timetable, each timetable contains the information about screen, time and movies, which can guide audiences to choose.
Administrator.java	Identify administrator, an instance of administrator can update the information of the film and timetable and so on.
BankAccount.java	Identify bank account, which is used to pay.
Ticket.java	Identify film ticket, there are three kinds of ticket, each ticket has unique number.
Student.java	Identify student, students can enjoy discount of the film, and each students has a unique student ID.
Screen.java	Identify screen, each screen has their own layout.

Boundary class

Name	Description
WelcomePanel.java	the welcomeinterface, containing some welcome sentence, such as “hello”

MoviesIntroPanel.java	the MoviesIntro interface, containing some basic information about movies
InfoPanel.java	the Infointerface, containing some details about each movies, each movies has an unique Info interface
ScreenSelectPanel.java	the ScreenSelect interface, which is used to choose screen for audiences
TimeSelectDialog.java	the TimeSelectDialog interface, which is a dialog used to choose time
ClientNormalButtonUI.java	define the normal button, used to back or continuous
ClientSpecialButtonUI.java	define the special button used to min or exit
ClientBackgroundPanel.java	define the background
ModifyMenuPanel.java	the ModifyMenuPanel interface, which is used to choose films
PaymentDialogPanel.java	The PaymentDialogPanel interface, which is used to payment
Screen1SeatSelectPanel.java	a kind of screen, which has a unique layout
Screen2SeatSelectPanel.java	a kind of screen, which has a unique layout
Screen3SeatSelectPanel.java	a kind of screen, which has a unique layout
SeatSelectedButtonUI.java	define a kind of button, which is used in the dialog, having orange color
SeatSelectingButtonUI.java	define a kind of button, which is used to select a seat
SeatTypeSelectDialog.java	the dialog used to seat, the selecting buttons are in it
StatisticsPanel.java	used by the administrator, to show the sell result of a day
StudentInfoCheckDialog.java	used to check the student to determine whether the user can buy the student ticket or not
StudentInfoCheckHintDialog.java	used to check the student keyword
AdministratorLoginPanel.java	used to log for administrator
AdministratorMenuPanel.java	used to choose function for administrator
CartDialog.java	used to calculate the result of the selling for each day
CartDialogPanel.java	used to show the statistical result

Control class

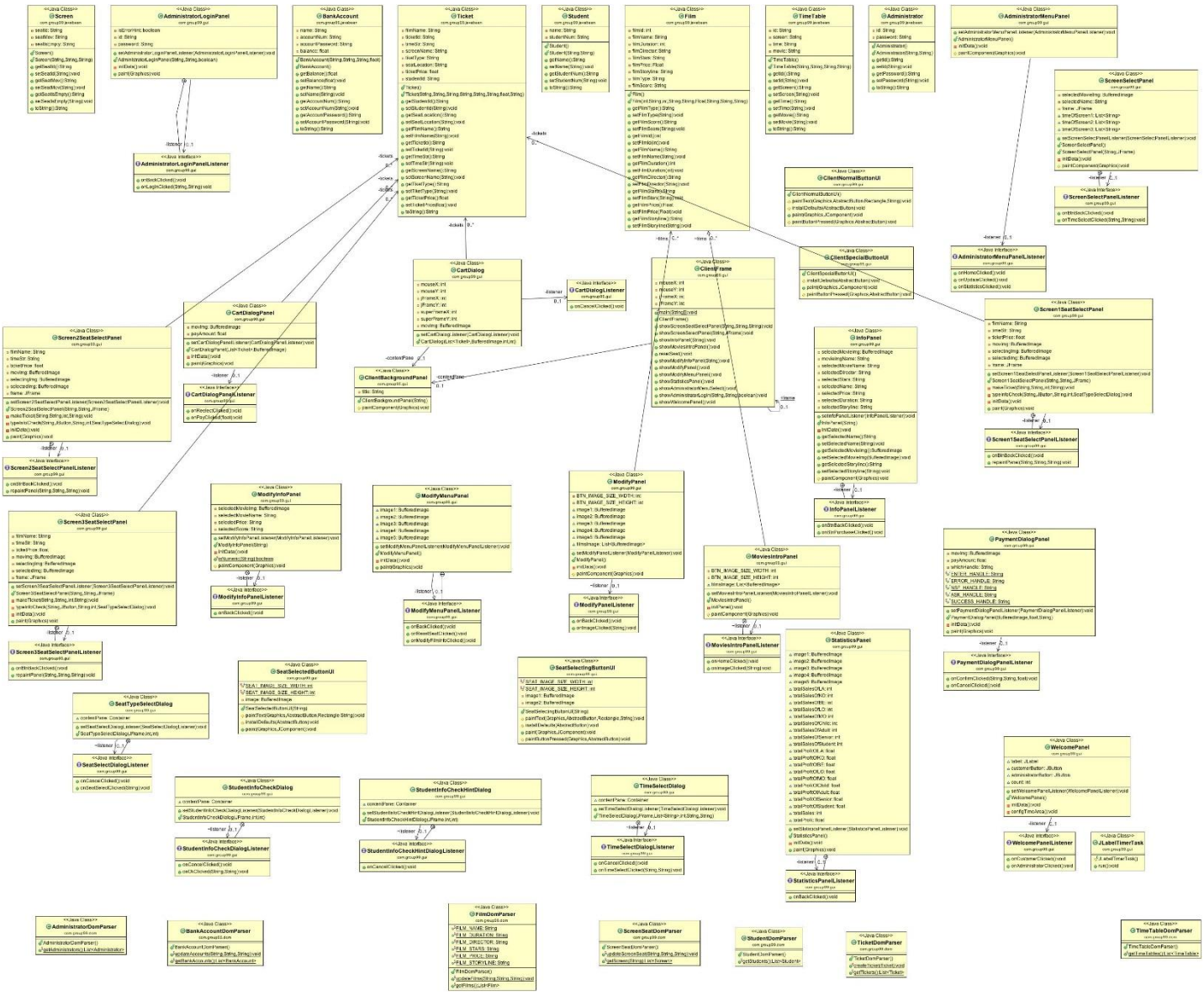
Name	Description
ClientFrame.java	used to realize all interface
FilmDomParser.java	used to read/write the film information from/to file
TimeTableDomParser.java	used to read/write the timetable information from/to file
AdministratorDomParser.java	used to read/write the administrator information from/to file
BankAccountDomParser.java	used to read/write the bank account information from/to file
ScreenSeatDomParser.java	used to read/write the seat information from/to file
StudentDomParser.java	used to read/write the student information from/to file
TicketDomParser.java	used to read/write the ticket information from/to file

RETURN

Appendix---Logical Interface

RETURN

Appendix---Complete Class Diagram



RETURN

Appendix--- Attributes and Operations of all classes

Class	AdministratorDomParser
Methods	List<Administrator> getAdministrators()
Class	BankAccountDomParser
Methods	updateAccounts(String, String, String)
	List<BankAccount> getBankAccounts()
Class	FilmDomParser
Methods	updateFilms(String, String, String)
	List<Film> getFilms()
Class	ScreenSeatDomParser
Methods	updateScreenSeat(String, String, String)
	List<Screen> getScreen(String)
Class	StudentDomParser
Methods	List<Student> getStudents()
Class	TicketDomParser
Methods	createTicket(Ticket) List<Ticket> getTickets()
Class	TimeTableDomParser
Methods	List<TimeTable> getTimeTables()

Class	AdministratorLoginPanel
Attributes	listener isErrorHint
	id
Methods	password setAdministratorLoginPanelListener(AdministratorLoginPanelListener)

Class	AdministratorLoginPanel
	AdministratorLoginPanel(String, String, boolean)
	initData() paint(Graphics)
Class	**AdministratorMenuPanel**
Methods	setAdministratorMenuPanelListener(AdministratorMenuPanelListener) AdministratorMenuPanel()
	initData() paintComponent(Graphics)
Class	**CartDialog**
Attributes	contentPane
	mouseX mouseY
	jFrameX
	jFrameY listener
	superFrameX
	superFrameY
	movImg
	tickets
Methods	setCartDialogListener(CartDialogListener)
	CartDialog(List<Ticket>, BufferedImage, int, int)
Class	**CartDialogPanel**
Attributes	movImg tickets
	payAmount
	listener

Class	AdministratorLoginPanel
Methods	setCartDialogPanelListener(CartDialogPanelListener)
	CartDialogPanel(List<Ticket>, BufferedImage)
	initData()
	paint(Graphics)

Class	ClientBackgroundPanel
Attributes	Title
Methods	ClientBackgroundPanel(String)
	paintComponent(Graphics)

Class	ClientFrame
Attributes	contentPane
	mouseX
	mouseY
	jFrameX
	jFrameY
	frame
Methods	main(String[])
	ClientFrame()
	showScreenSeatSelectPanel(String, String, String)
	showScreenSelectPanel(String, JFrame)
	showInfoPanel(String)
	showMoviesIntroPanel()
	resetSeat()
	showModifyInfoPanel(String)
	showModifyPanel()
	showModifyMenuPanel()

Class	ClientFrame
	showStatisticsPanel()
	showAdministratorMenuSelect() showAdministratorLogin(String, String, boolean)
	showWelcomePanel()

Class	ClientNormalButtonUI
Methods	paintText(Graphics,AbstractButton,Rectangle, String)
	installDefaults(AbstractButton)
	paint(Graphics, JComponent) paintButtonPressed(Graphics, AbstractButton)

Class	ClientSpecialButtonUI
Methods	installDefaults(AbstractButton)
	paintButtonPressed(Graphics, AbstractButton)
	paint(Graphics, JComponent)

Class	InfoPanel
Attributes	selectedMovieImg movieImgName
	selectedMovieName
	selectedDirector selectedStars
	selectedName selectedPrice selectedDuration
	selectedStoryline
	listener

Class	InfoPanel
Methods	setInfoPanelListener(InfoPanelListener)
	InfoPanel(String) initData()
	getSelectedName()
	setSelectedName(String) getSelectedMovieImg()
	setSelectedMovieImg(BufferedImage)
	getSelectedStoryline() setSelectedStoryline(String)
	paintComponent(Graphics)

Class	ModifyInfoPanel
Attributes	selectedMovieImg
	selectedMovieName
	selectedPrice
	selectedScore
	listener
Methods	setModifyInfoPanelListener(ModifyInfoPanelListener)
	ModifyInfoPanel(String) initData() isNumeric(String)
	paintComponent(Graphics)

Class	ModifyMenuPanel
Attributes	image1
	image2
	image3

Class	ModifyMenuPanel
	image4
	image5
	listener
Methods	setModifyMenuPanelListener(ModifyMenuPanelListener)
	ModifyMenuPanel()
	initData()
	paint(Graphics)

Class	ModifyPanel
Attributes	BTN_IMAGE_SIZE_WIDTH
	BTN_IMAGE_SIZE_HEIGHT
	image1
	image2
	image3
	image4
	image5
	films
	filmsImage
Methods	listener
	setModifyPanelListener(ModifyPanelListener)
	ModifyPanel()
	initData()
	paintComponent(Graphics)
Class	MoviesIntroPanel
Attributes	BTN_IMAGE_SIZE_WIDTH
	BTN_IMAGE_SIZE_HEIGHT
	films

Class	ModifyPanel
Methods	filmsImage
	listener
	setMoviesIntroPanelListener(MoviesIntroPanelListener)
	MoviesIntroPanel()
	initPanel()
	paintComponent(Graphics)

Class	PaymentDialogPanel
Attributes	movImg
	payAmount
	whichHandle
	ENTER_HANDLE ERROR_HANDLE NSF_HANDLE
	ASK_HANDLE SUCCESS_HANDLE
	listener
Methods	setPaymentDialogPanelListener(PaymentDialogPanelListener)
	PaymentDialogPanel(BufferedImage,float,String)
	initData()
	paint(Graphics)
Class	Screen1SeatSelectPanel
Attributes	filmName
	timeStr ticketPrice
	movImg selectingImg

Class	PaymentDialogPanel
	selectedImg
	frame tickets
	listener
Methods	setScreen1SeatSelectPanelListener(Screen1SeatSelectPanelListener)
	Screen1SeatSelectPanel(String,String,JFrame) makeTicket(String,String,int,String) typeInfoCheck(String,JButton,String,int, SeatTypeSelectDialog) initData() paint(Graphics)
Class	Screen2SeatSelectPanel
Attributes	filmName timeStr
	ticketPrice
	movImg selectingImg
	selectedImg frame tickets
	listener setScreen2SeatSelectPanelListener(Screen1SeatSelectPanelListener)
Methods	Screen2SeatSelectPanel(String,String,JFrame) makeTicket(String,String,int,String)
	typeInfoCheck(String,JButton,String,int, SeatTypeSelectDialog) initData()

Class	PaymentDialogPanel
	paint(Graphics)

Class	Screen3SeatSelectPanel
Attributes	filmName
	timeStr
	ticketPrice movImg
	selectingImg selectedImg frame
	tickets listener
Methods	setScreen3SeatSelectPanelListener(Screen1SeatSelectPanelListener) Screen3SeatSelectPanel(String,String,JFrame)
	makeTicket(String,String,int,String) typeInfoCheck(String,JButton,String,int, SeatTypeSelectDialog)
	initData()
	paint(Graphics)
Class	ScreenSelectPanel
Attributes	selectedMovieImg
	selectedName frame
	timeOfScreen1
	timeOfScreen2 timeOfScreen3
	listener

Class	Screen3SeatSelectPanel
Methods	setScreenSelectPanelListener(ScreenSelectPanelListener) ScreenSelectPanel() ScreenSelectPanel(String,JFrame)
	initData() paintComponent(Graphics)

Class	SeatSelectedButtonUI
Attributes	SEAT_IMAGE_SIZE_WIDTH
	SEAT_IMAGE_SIZE_HEIGHT image
Methods	SeatSelectedButtonUI(String) paintText(Graphics,AbstractButton,Rectangle, String)
	installDefaults(AbstractButton) paint(Graphics, JComponent)
Class	SeatSelectingButtonUI
Attributes	SEAT_IMAGE_SIZE_WIDTH
	SEAT_IMAGE_SIZE_HEIGHT image1
	image2 SeatSelectingButtonUI(String)
Methods	paintText(Graphics,AbstractButton,Rectangle, String) installDefaults(AbstractButton)
	paint(Graphics,JComponent)
	paintButtonPressed(Graphics,AbstractButton)
Class	SeatTypeSelectDialog

Class	SeatSelectedButtonUI
Attributes	listener contentPane
Methods	setSeatSelectDialogListener(SeatSelectDialogListener) SeatTypeSelectDialog(JFrame,int,int)

Class	StatisticsPanel
Attributes	image1
	image2
	image3
	image4
	image5
	totalSalesOfLA
	totalSalesOfKO
	totalSalesOfBE
	totalSalesOfLO
	totalSalesOfMO
	totalSalesOfChild
	totalSalesOfAdult
	totalSalesOfSenior
	totalSalesOfStudent
	totalProfitOfLA
	totalProfitOfKO
	totalProfitOfBE
	totalProfitOfLO
	totalProfitOfMO
	totalProfitOfChild

Class	StatisticsPanel
	totalProfitOfAdult
	totalProfitOfSenior
	totalProfitOfStudent
	totalSales
	totalProfit
	listener
Methods	setStatisticsPanelListener(StatisticsPanelListener)
	StatisticsPanel()
	initData()
	paint(Graphics)

Class	StudentInfoCheckDialog
Attributes	listener
	contentPane
Methods	setStudentInfoCheckDialogListener(StudentInfoCheckDialogListener)
	StudentInfoCheckDialog(JFrame,int,int)
Class	StudentInfoCheckHintDialog
Attributes	listener
	contentPane
Methods	setStudentInfoCheckHintDialogListener(StudentInfoCheckHintDialogListener)
	StudentInfoCheckHintDialog(JFrame,int,int)
Class	TimeSelectDialog
Attributes	listener
	contentPane
Methods	setTimeSelectDialogListener(TimeSelectDialogListener)

Class	StudentInfoCheckDialog
	TimeSelectDialog(JFrame,List<String>,int, String,String)
Class	WelcomePanel
Attributes	listener
	label
	customerButton administratorButton
	count setWelcomePanelListener(WelcomePanelListener)
Methods	WelcomePanel() initData() configTimeArea()

Class	Administrator
Attributes	id
	password Administrator() Administrator(String,String)
Methods	getId() setId(String)
	getPassword()
	setPassword(String) toString()
Class	BankAccount
Attributes	name
	accountNum

Class	Administrator
Methods	accountPassword
	balance
	BankAccount(String,String,String,float)
	BankAccount()
	getBalance()
	setBalance(float)
	getName()
	setName(String)
	getAccountNum()
	setAccountNum(String)
	getAccountPassword()
	setAccountPassword(String)
	toString()

Class	Film
Attributes	filmId
	filmName
	filmDuration
	filmDirector
	filmStars
	filmPrice
	filmStoryline
	filmType
	filmScore
Methods	Film()
	Film(int,String,int,String,String,Float, String,String,String)
	getFilmType()

Class	Film
	setFilmType(String)
	getFilmScore setFilmScore(String)
	getFilmId()
	setFilmId(int) getFilmName()
	setFilmName(String)
	getFilmDuration() setFilmDuration(int)
	getFilmDirector() setFilmDirector(String) getFilmStars()
	setFilmStars(String) getFilmPrice()
	setFilmPrice(Float)
	getFilmStoryline() setFilmStoryline(String)

Class	Screen
Attributes	seatId
	seatMov seatIsEmpty
Methods	Screen() Screen(String,String,String) getSeatId()
	setSeatId(String)

Class	Screen
	getSeatMov()
	setSeatMov(String)
	getSeatIsEmpty()
	setSeatIsEmpty(String)
	toString()
Class	Student
Attributes	name
	studentNum
Methods	Student()
	Student(String,String)
	getName()
	setName(String)
	getStudentNum()
	setStudentNum(String)
	toString()

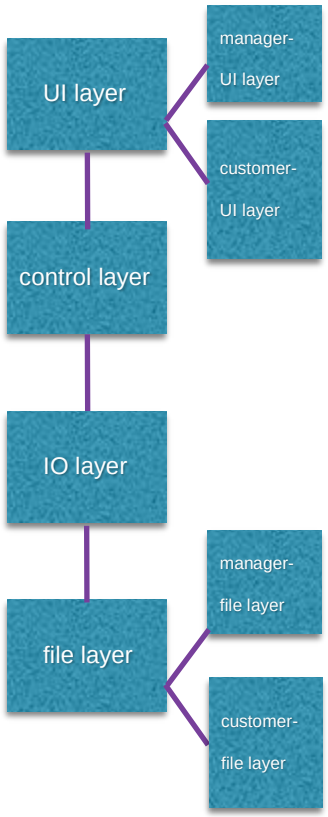
Class	Ticket
Attributes	filmName
	ticketId
	timeStr
	screenName
	tiketType
	seatLocation
	ticketPrice
	studentId
Methods	Ticket()
	Ticket(String,String,String,String,String, String,float,String)

Class	Ticket
	getStudentId()
	setStudentId(String) getSeatLocation()
	setSeatLocation(String)
	getFilmName() setFilmName(String)
	getTicketId()
	setTicketId(String) getTimeStr()
	setTimeStr(String) getScreenName() setScreenName(String)
	getTiketType() setTiketType(String)
	getTicketPrice()
	setTicketPrice(float) toString()

Class	TimeTable
Attributes	id
	screen time movie
Methods	TimeTable() TimeTable(String,String,String,String)
	getId()
	setId(String)
	getScreen()

Class	TimeTable
	setScreen(String)
	getTime() setTime(String)
	getMovie()
	setMovie(String) toString()

Appendix---Architecture Design



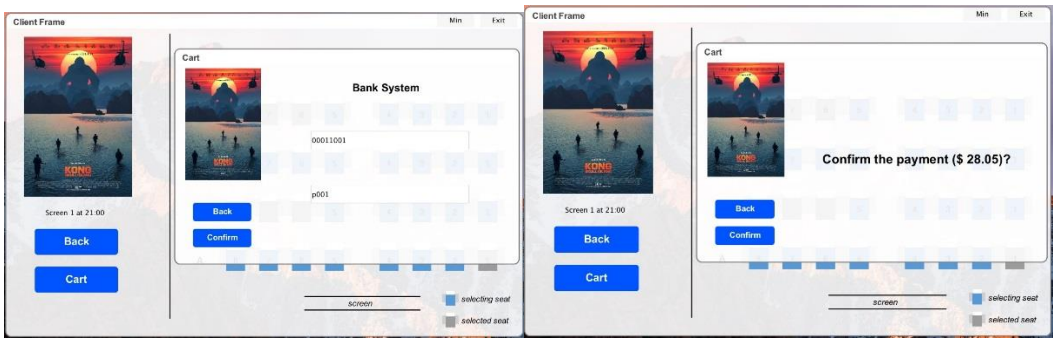
RETURN

Appendix---Partition Testing and Scenario-based testing

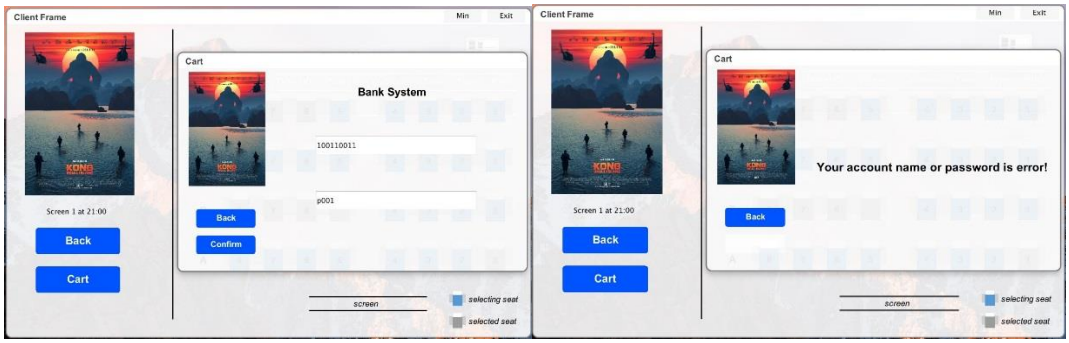
Partitioning testing

Input 6 digits as bank account, the system will generate a error message

Input 8 digits as bank account, the system will not generate error message



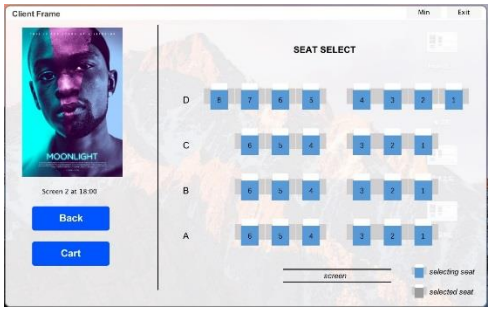
Input 9 digits as bank account, the system will generate a error message



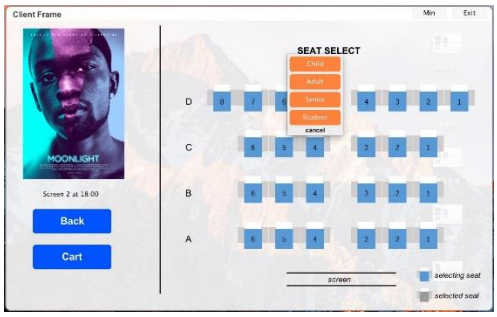
Test case	
Input	Output
6 digits	Error message
8 digits	Right, Confirmation message
9 digits	Error message

Scenario-based Testing

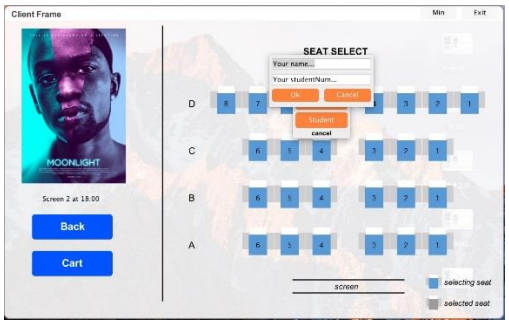
Displaying the complete layout of each screen



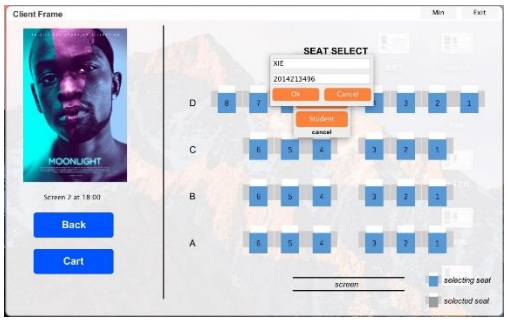
Choosing the certain type of tickets



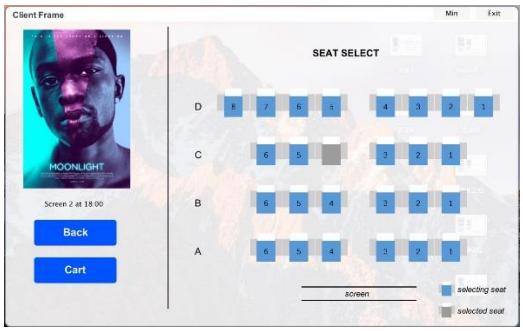
Input the name and student number.



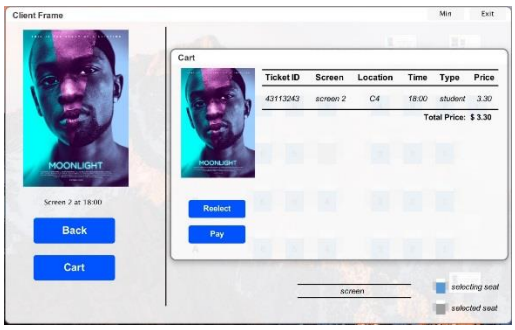
The input is finished



Choosing seats is successful.



Display the information of tickets.



So customers can select seat and buy tickets successfully.

RETURN

Appendix---Test cases

1. Test for input of bank account

Test Case: File Open	#Test Description	Test Cases	Samples Pass/Fail	No.of Bugs	Bug#	Comments
N/A	Create a bank account in system [account number - 23345666 and PIN - 12345544]	setup	-	-	-	Added to system for testing purposes
1.1	Test that the account number 23345666 does exist in the system	1.1	P	0	0	Correct behavior observed
1.2	Test that the account number 23345667 does not exist in the system	1.2	P	0	0	Correct behavior observed
1.3	Test that the PIN 12345548 of account 23345666 is incorrect	1.3	P	0	0	Correct behavior observed
1.4	Test that the PIN 12345544 of account 23345666 is correct	1.4	P	0	0	Correct behavior Observed

2. Test for input of administrator account

Test Case: File Open	**#Test Description**	**Test Cases**	**Samples Pass/Fail**	**No.of Bugs**	**Bug#**	**Comments**
N/A	Create an administrator account in system (account is 86595569)	setup	-	-	-	Added to system for testing purposes
2.1	Test that the account 86595569 does exist in the system	2.1	P	0	0	Correct behavior observed
2.2	Test that the account 86595568 does not exist in the system	2.2	P	0	0	Correct behavior observed

3. Test for seat selecting

Test Class	Case 1(Correct input)	Case 2 (Incorrect input)	Case 3 (Incorrect input)
Input	Click orange button (selectable)	Click grey button. (can't select)	Click "BACK"
Result	It turns to display four types of tickets	No response	It turns to previous interface
Condition	Get into the seat selecting interface	Get into the seat selecting interface.	Get into the seat selecting interface.

Item	Test description	Test cases	Samples Pass/Fail	Number of bugs	Bug	Comment
N/A	Input: Click button	Set up	-	-	-	Testing system
3.1	Test that clicking orange button can display four different types of tickets	3.1	P	0	0	Correct behavior observed
3.2	Test that clicking grey button has no response	3.2	P	0	0	Correct behavior observed
3.3	Test that clicking “BACK” button can use	3.3	P	0	0	Correct behavior observed

4. Test for selecting different types of tickets

Test Class	Case 1(Correct input)	Case 2 (Incorrect input)
Input	Click “child” or “Adult” or “Senior” or “Student” and enter the student number, then click CONFIRM	Click “Student”, and then click “CONFIRM”.
Result	It turns back to the layout of screen, and the color of the seat selected by customers becomes grey	A new interface to ask you to input student number
Condition	Get into the type selecting interface	Get into the type selecting interface

Item	Test description	Test cases	Samples Pass/Fail	Number of bugs	Bug	Comment
N/A	Input: Click button	Set up	-	-	-	Testing system

4.1	Test that click “Child” “Adult” “Senior”	4.1	P	0	0	Correct behavior observed
4.2	Test that click “Student” and input student number	4.2	P	0	0	Correct behavior observed

5. Test for confirmation of transaction

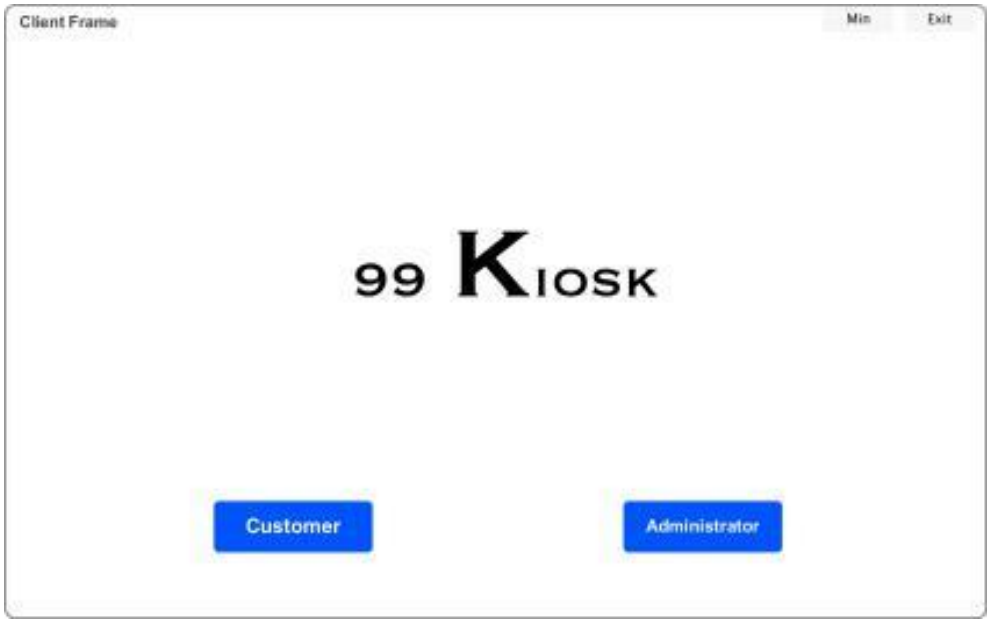
Test Class	Case 1(Correct input)	Case 2 (Incorrect input)
Input	Click “CONFIRM”	Click “BACK”
Result	The system will tell you that purchasing successful.	It turns to previous interface.
Condition	Get into the confirmation message	Get into the confirmation message.

Item	Test description	Test cases	Samples Pass/Fail	Number of bugs	Bug	Comment
N/A	Input: Click button	Set up	-	-	-	Testing system
5.1	Test that click “CONFIRM”	5.1	P	0	0	Correct behavior observed
5.2	Test that click “BACK”	5.2	P	0	0	Correct behavior observed

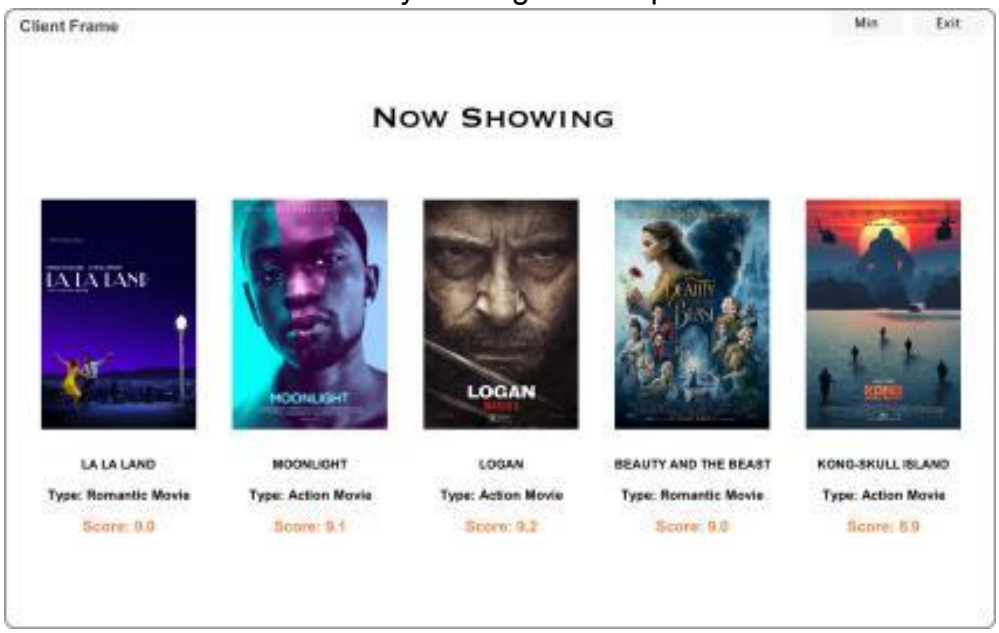
RETURN

Appendix---Main screenshots of the system

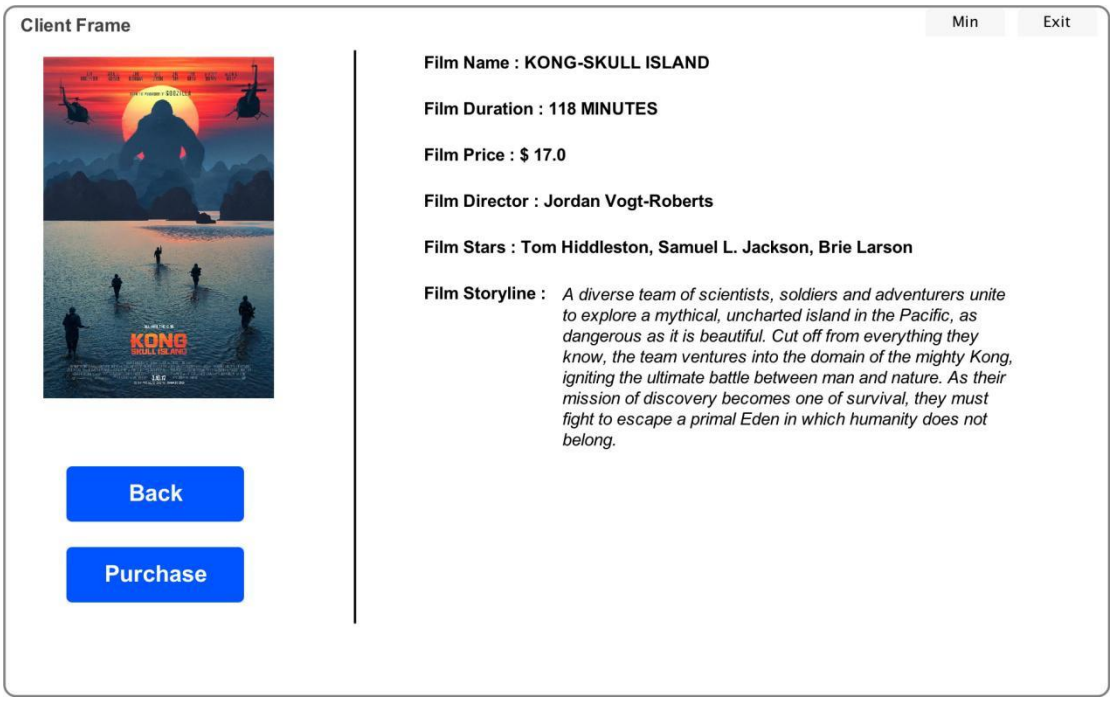
You can choose one mode.



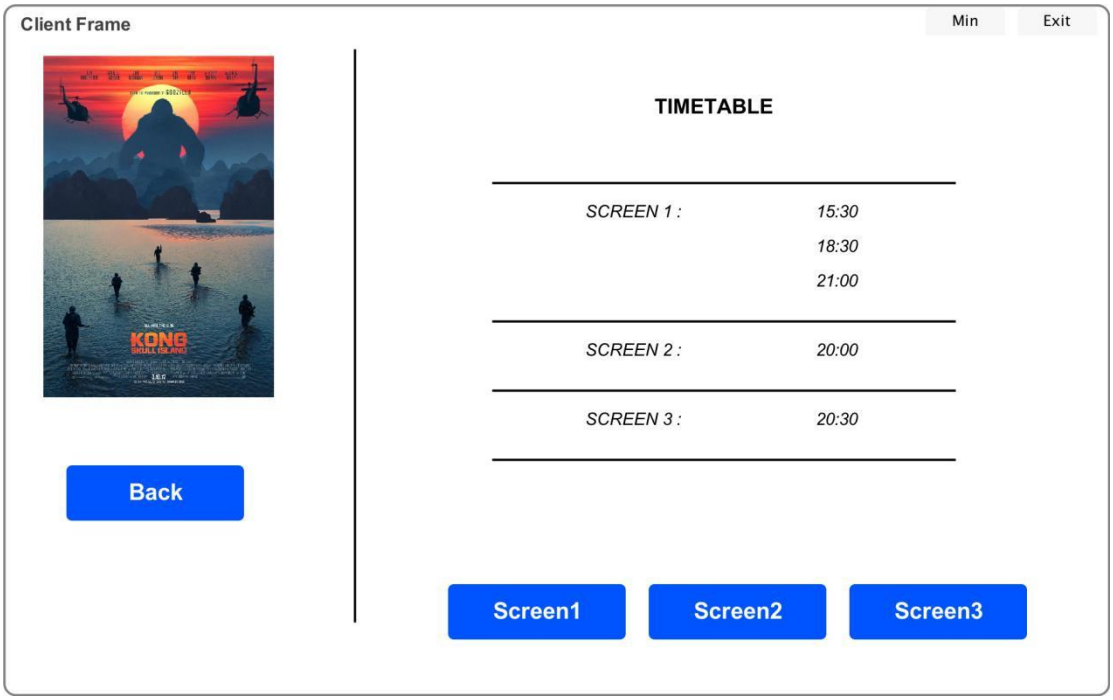
You can choose one movie by clicking relative picture.

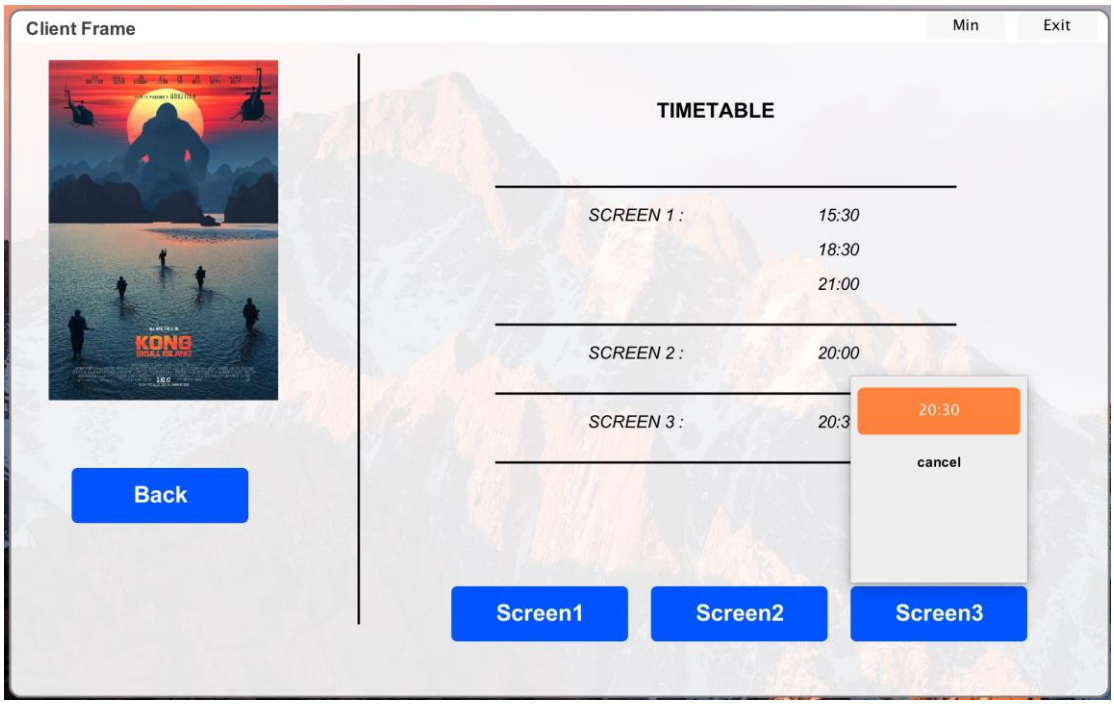


Now you can obtain detailed information of the movie.

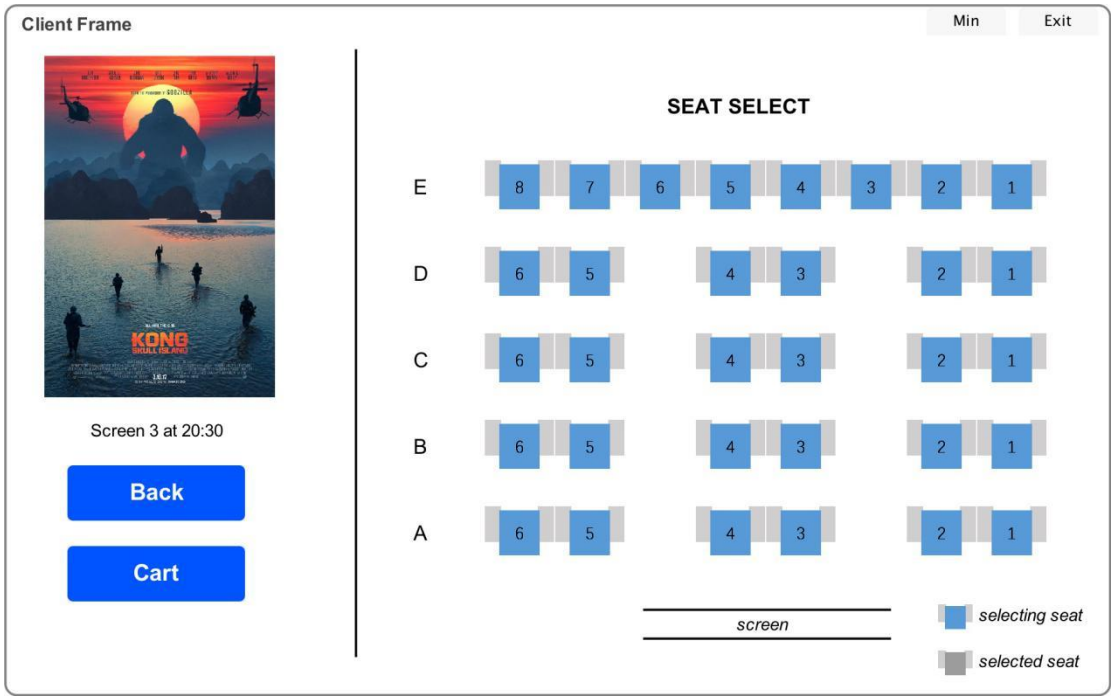


If you click “Purchase”, you can see the time table of the movie and you can also choose screen

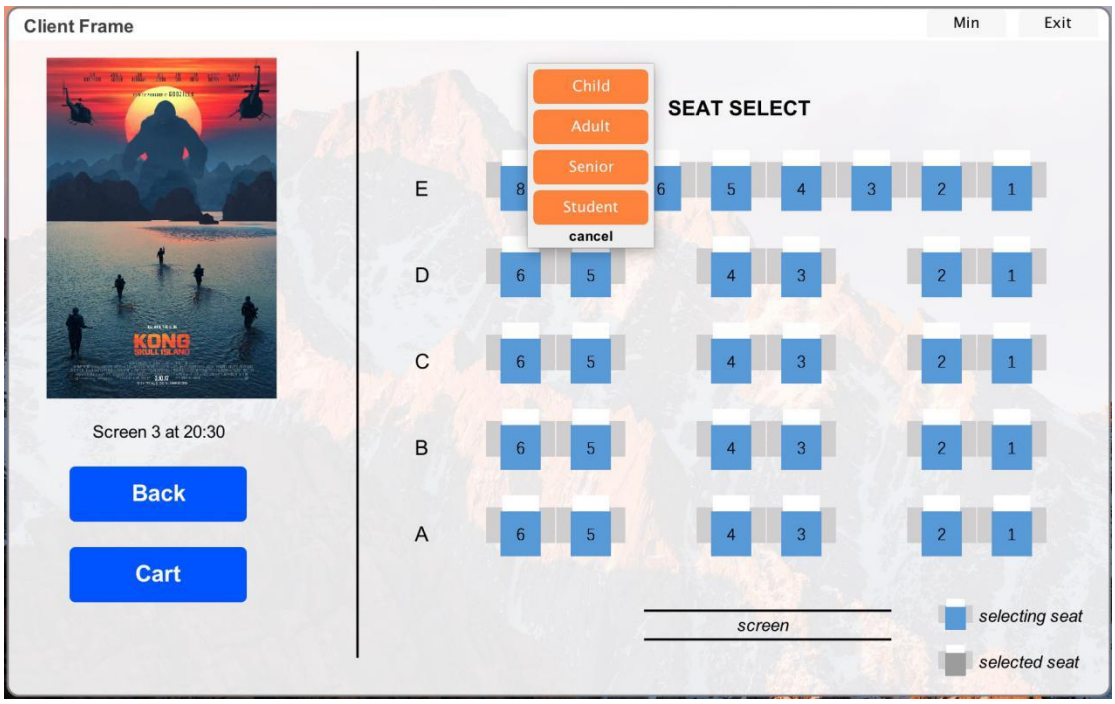




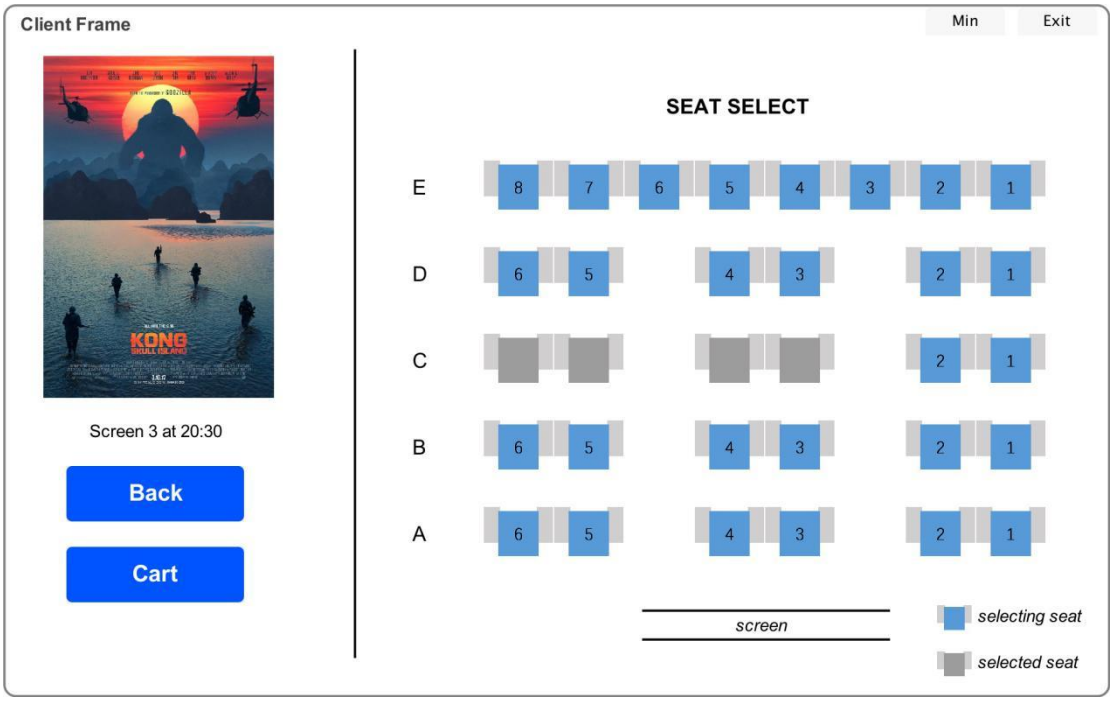
After clicking time button, you will see the layout of the screen and you can click the seat you want.



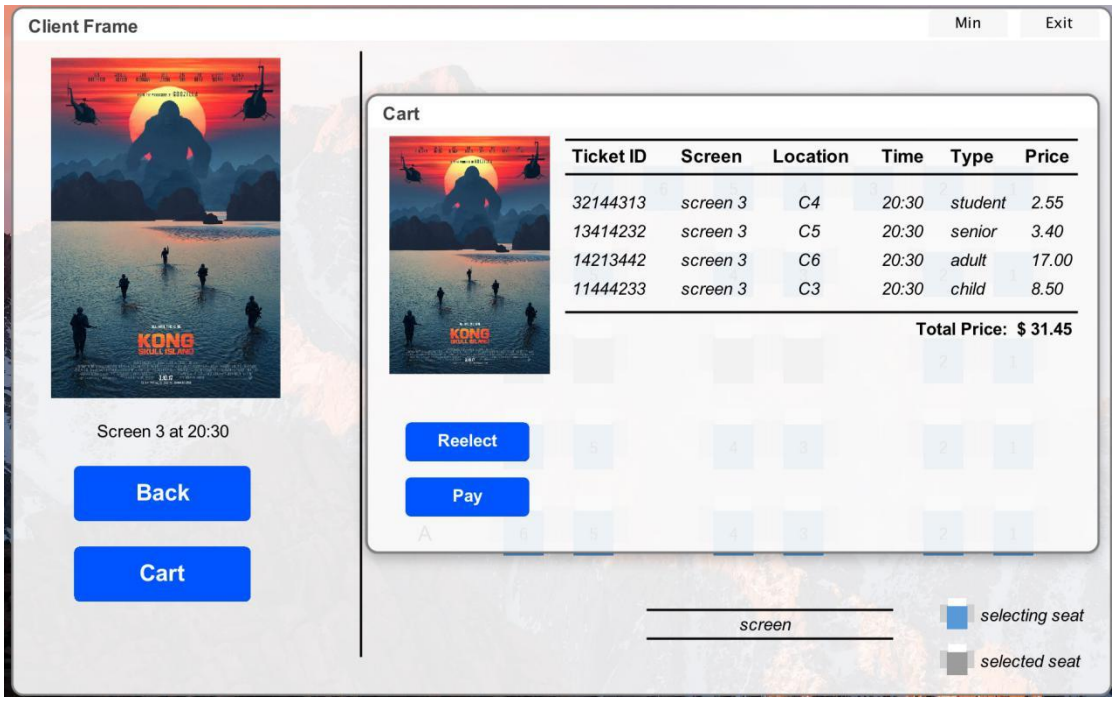
When you click the seat, you can choose the type of ticket.



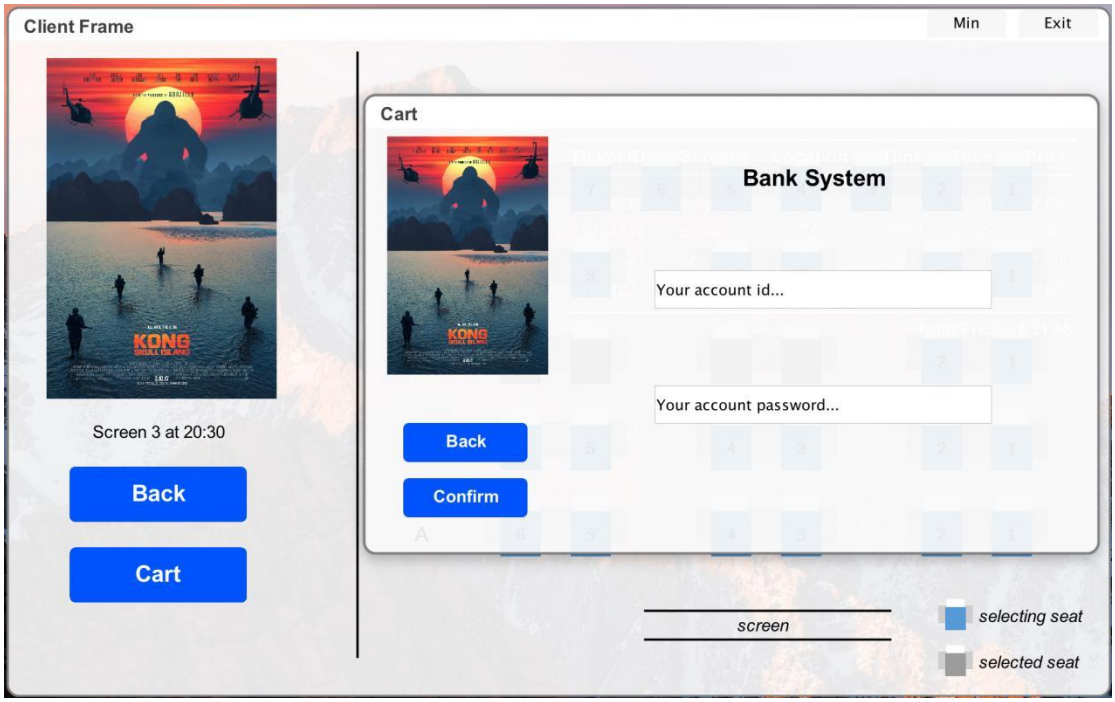
After choosing seats, the color of seat selected by customers will become grey.



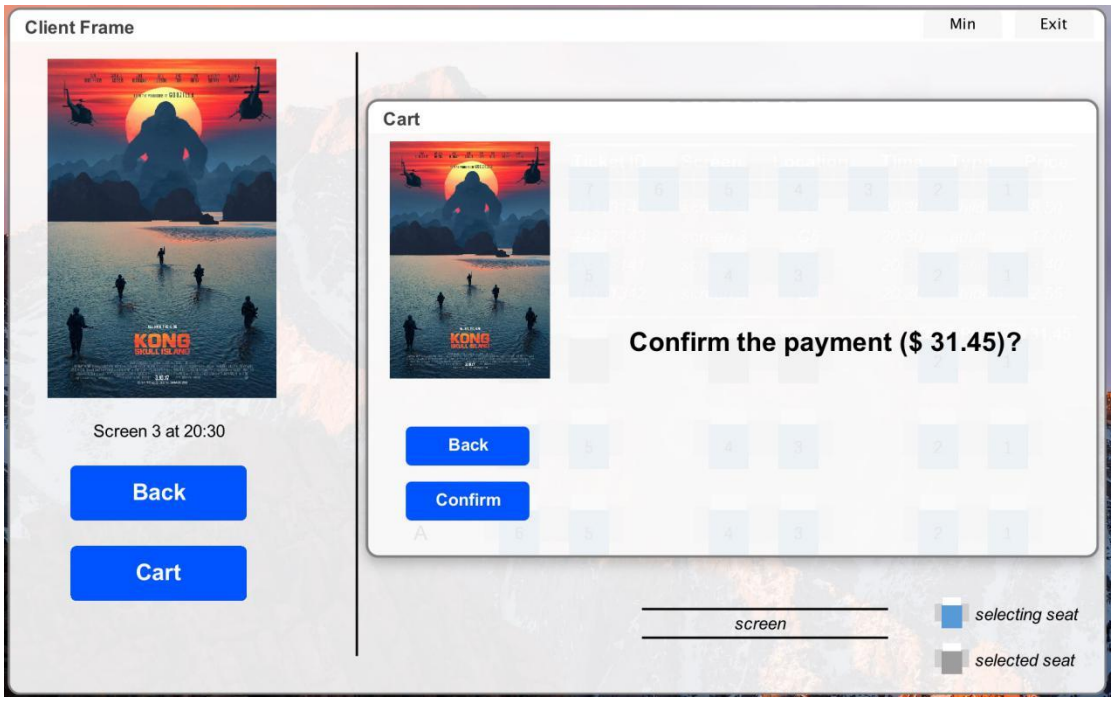
Then if you click “Cart”, you will see all tickets you have ordered.



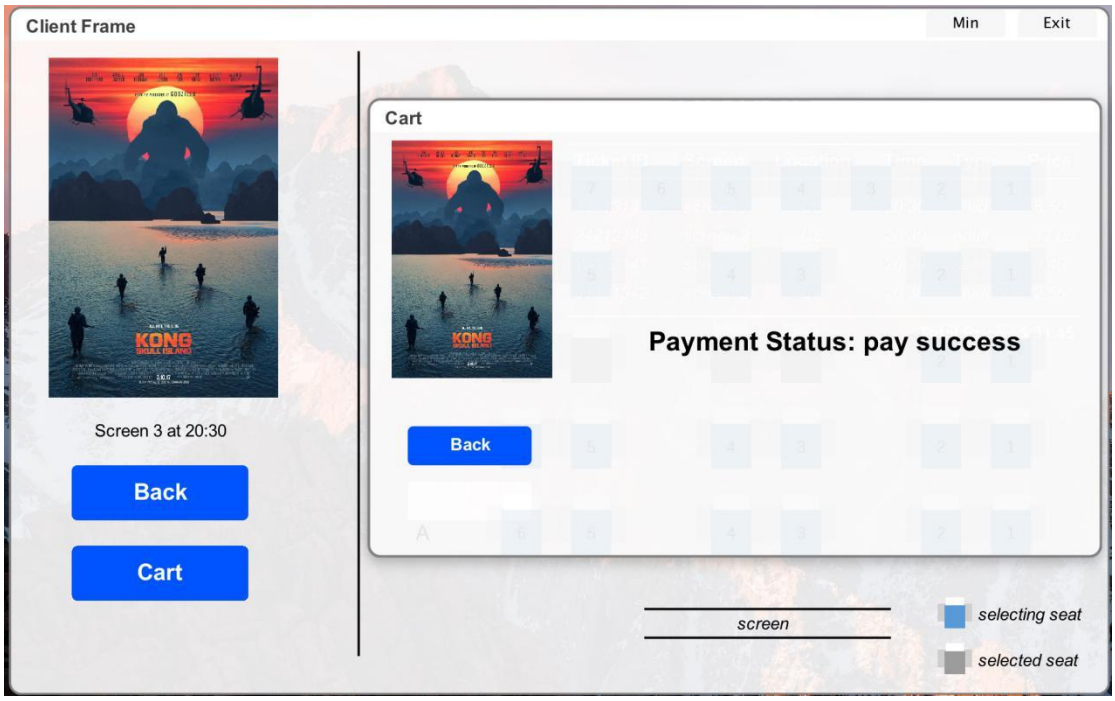
When you click “Pay”, you will need to input your bank account and password.



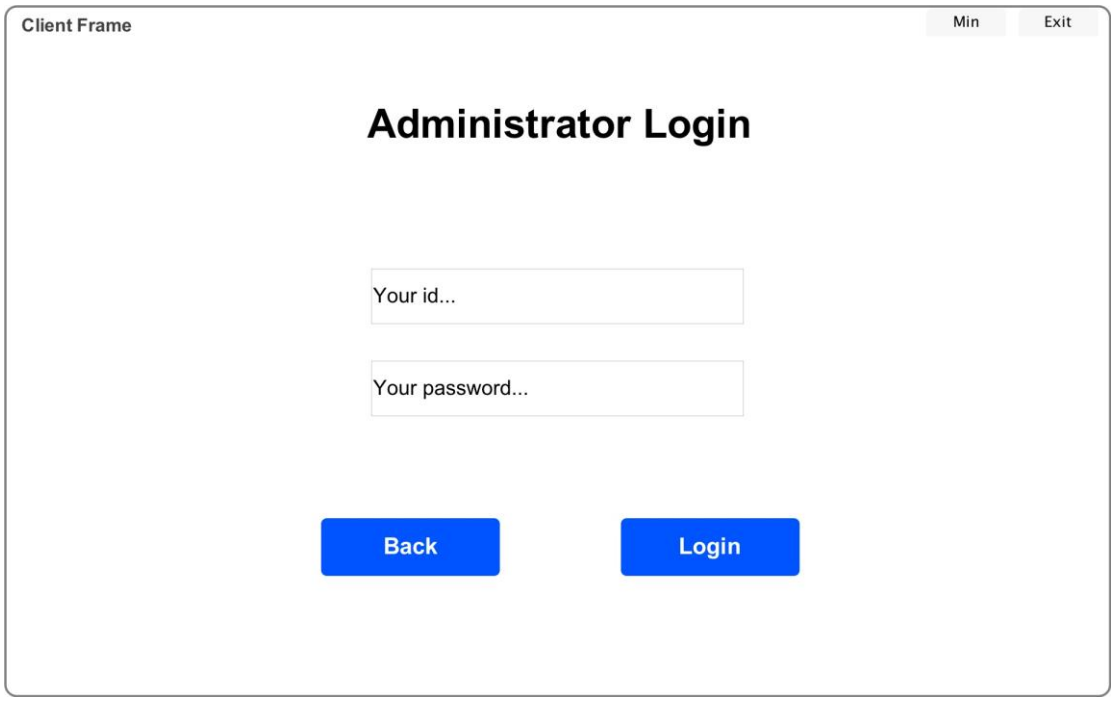
If your bank account and password are correct, you can see a confirmation message.



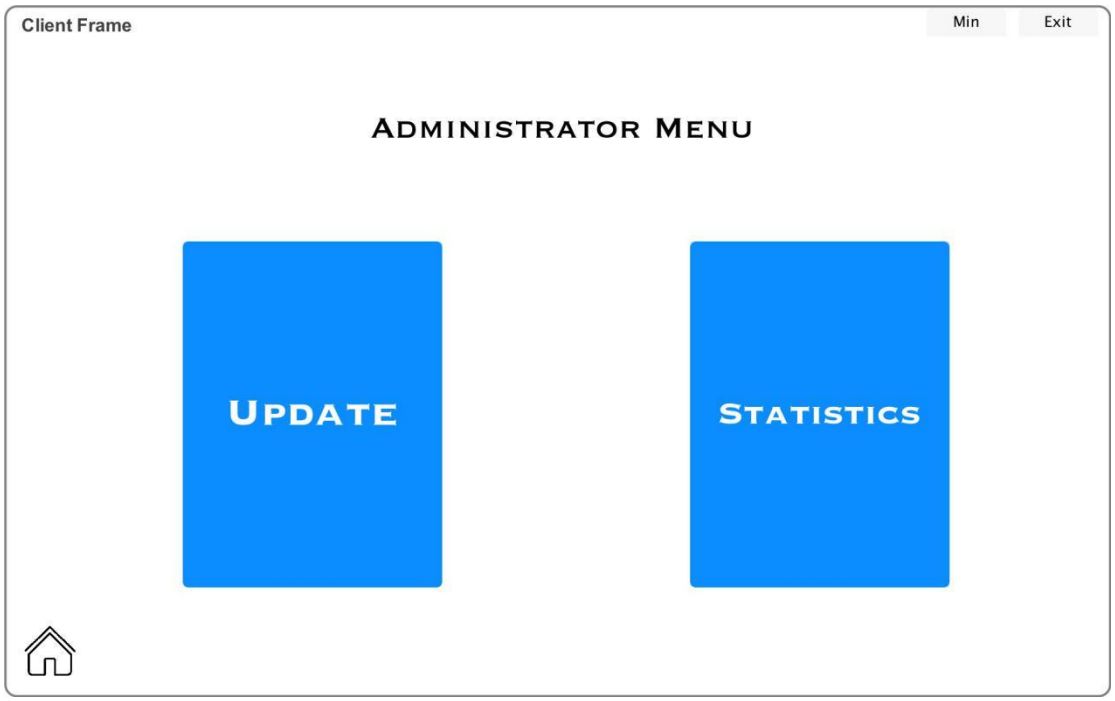
If you click “Confirm”, you can see “Pay successfully” message.

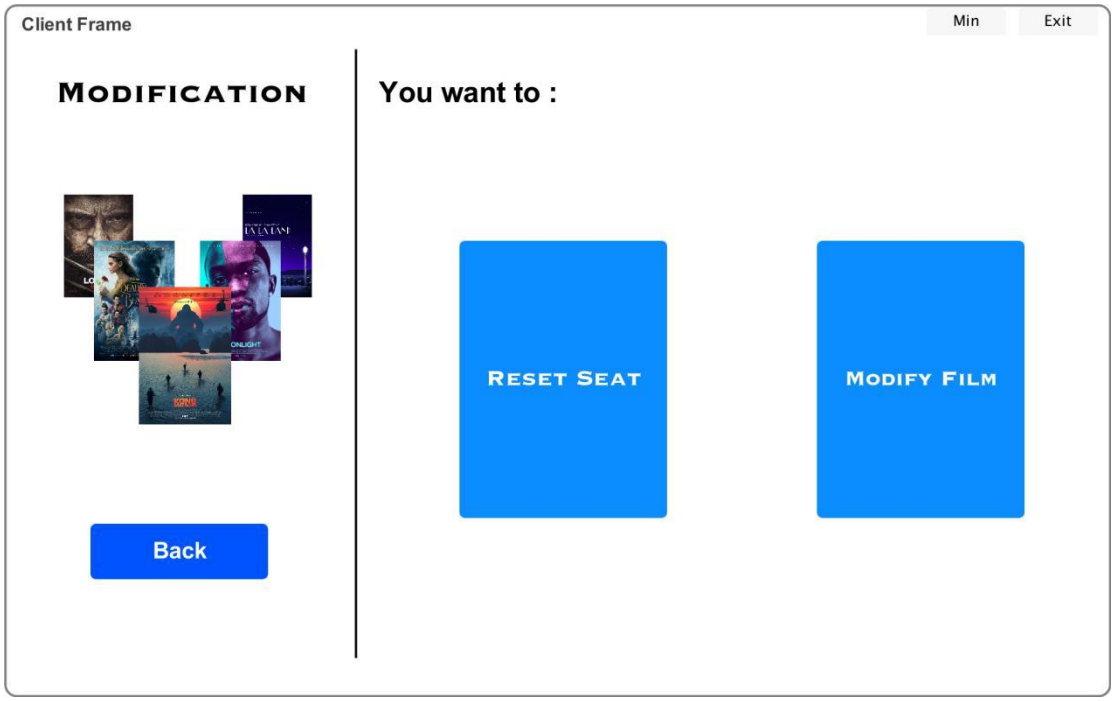


The steps above are the main function of the customer system.
As for administrator mode, first administrator needs to enter the username and the password, and then select one operation.

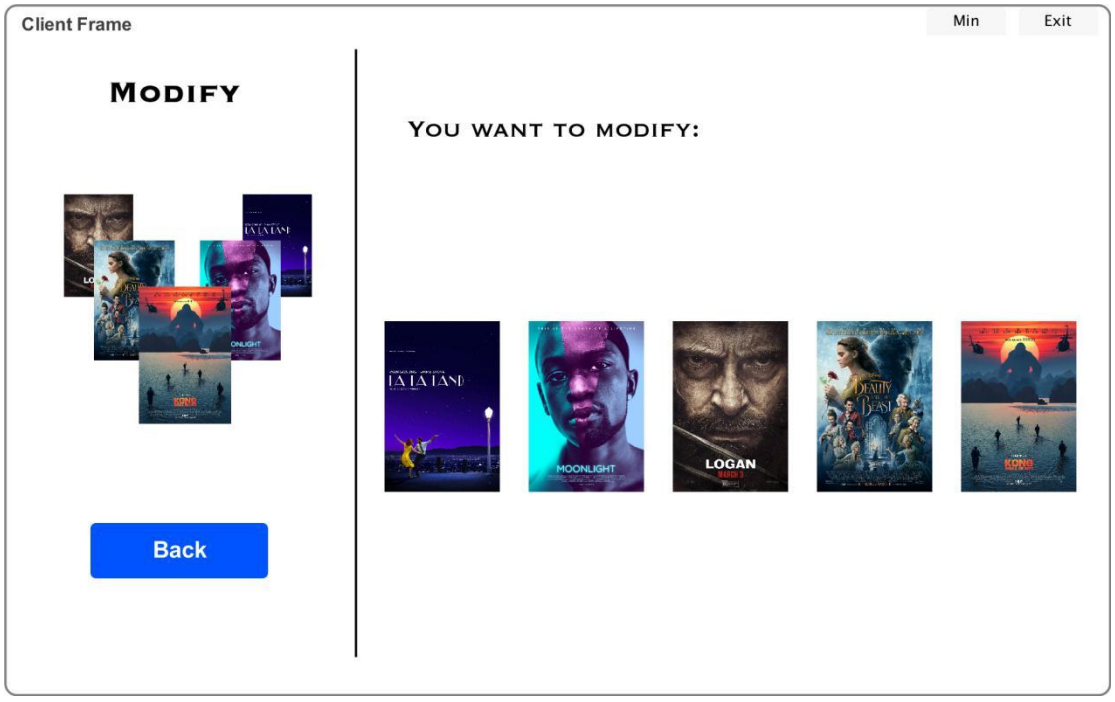


Update is to update the information of the movies. Statistics is to analyze the data of sales of tickets.





You can choose which information of movie you want to change.



Client Frame

Min Exit

Back

Submit

MODIFY INFO

New Price: The old price: 22.0

New Score: The old score: 9.5

This is the statistic report.

Client Frame

Min Exit

STATISTICS

Back

Email

Film Name	Total Sales	Total Profit
LA LA LAND	0	$ 0.00
KONG-SKULL ISLAND	4	$ 39.95
BEAUTY AND THE BEAST	0	$ 0.00
LOGAN	0	$ 0.00
MOONLIGHT	0	$ 0.00

Ticket Type	Total Sales	Total Profit
Adult	2	$ 34.00
Child	0	$ 0.00
Senior	1	$ 3.40
Student	1	$ 2.55

Total Sales	Total Profit
4	$ 39.95

Appendix---Code

```
package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of panel of logging in by administrator.
 * @author group 99.
 *
 */
public class AdministratorLoginPanel extends JPanel {

    private AdministratorLoginPanelListener listener;
    private boolean isErrorHint;
    private String id;
    private String password;
    /**
     * This is the callback interface of administrator's logging in panel.
     * @author group 99
     *
     */
    public interface AdministratorLoginPanelListener{
        public void onBackClicked();
        public void onLoginCicked(String id, String password);
    }
    /**
     * Set AdministratorLoginPanel's listener.
```

```
     * @param listener
     */
    public void
setAdministratorLoginPanelListener(AdministratorLoginPanelListener
listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of AdministratorLoginPanel.
     * @param id The id of administrator.
     * @param password The password of administrator's account.
     * @param isErrorHint Whether the panel generates error message or not.
     */
    public AdministratorLoginPanel(String id, String password, boolean
isErrorHint) {
        this.id = id;
        this.password = password;
        this.isErrorHint = isErrorHint;
        initData();
    }
    /**
     * Initialize the data of AdministratorLoginPanel.
     */
    private void initData() {

        setLayout(null);

        JTextField idTextField = new JTextField(15);
        idTextField.setBounds(310, 190, 330, 55);
        if(id == null){
            idTextField.setText("Your id...");
        }else{
            idTextField.setText(id);
        }
        idTextField.setFont(new Font("Arial", Font.PLAIN, 18));
        idTextField.setOpaque(false);
        add(idTextField);

        JTextField passwordTextField = new JTextField(15);
        passwordTextField.setBounds(310, 270, 330, 55);
        if(password == null){
            passwordTextField.setText("Your password...");
        }else{
            passwordTextField.setText(password);
```

```
        }
        passwordTextField.setFont(new Font("Arial", Font.PLAIN, 18));
        idTextField.setOpaque(false);
        add(passwordTextField);

        JButton backButton = new JButton("Back");
        backButton.setUI(new ClientNormalButtonUI());
        backButton.setForeground(Color.WHITE);
        backButton.setFont(new Font("Arial", Font.BOLD, 20));
        backButton.setBounds(270, 410, 155, 50);
        backButton.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBackClicked();
            }

        });
        add(backButton);

        JButton loginButton = new JButton("Login");
        loginButton.setUI(new ClientNormalButtonUI());
        loginButton.setForeground(Color.WHITE);
        loginButton.setFont(new Font("Arial", Font.BOLD, 20));
        loginButton.setBounds(530, 410, 155, 50);
        loginButton.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onLoginCicked(idTextField.getText(),
passwordTextField.getText());
            }

        });
        add(loginButton);

    }
    /**
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
```

```
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.setFont(new Font("Arial", Font.BOLD, 35));
        graphics2d.drawString("Administrator Login", 310, 80);

        if(isErrorHint){
            graphics2d.setColor(Color.RED);
            graphics2d.setFont(new Font("Arial", Font.PLAIN, 15));
            graphics2d.drawString("Your id or password is error !", 450,
350);
        }

        super.paint(graphics2d);
    }

}



package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JPanel;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;
/**
 * This is the boundary class of panel of menu of administrator mode.
```

```
 * @author group 99.
 *
 */
public class AdministratorMenuPanel extends JPanel {

    private AdministratorMenuPanelListener listener;
    /**
     * This is the callback interface of menu of administrator mode.
     * @author group 99
     *
     */
    public interface AdministratorMenuPanelListener{
        public void onHomeClicked();
        public void onUpdateClicked();
        public void onStatisticsClicked();
    }
    /**
     * Set AdministratorMenuPanel's listener.
     * @param listener A listener of AdministratorMenuPanel.
     */
    public void
setAdministratorMenuPanelListener(AdministratorMenuPanelListener listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of AdministratorMenuPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public AdministratorMenuPanel() throws ParserConfigurationException,
SAXException, IOException {
        initData();
    }
    /**
     * Initialize the data of AdministratorMenuPanel.
     */
    private void initData() throws ParserConfigurationException,
SAXException, IOException {

        setLayout(null);

        BufferedImage image = ImageIO.read(new
FileInputStream("./../../resource/home.png"));
```

```
        JButton homeBack = new JButton();
        homeBack.setBounds(10, 500, 50, 50);
        homeBack.setOpaque(false);
        homeBack.setUI(new BasicButtonUI() {

            @Override
            public void paint(Graphics g, JComponent c) {
                Graphics2D graphics2d = (Graphics2D) g;
                graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                graphics2d.drawImage(image, 0, 0, 50, 50, null);
            }

        });
        homeBack.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onHomeClicked();
            }

        });
        add(homeBack);

        BasicButtonUI buttonUI = new BasicButtonUI(){
            @Override
            public void paint(Graphics g, JComponent c) {
                g.setColor(new Color(30, 144, 255));
                g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
                super.paint(g, c);
            }

            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(255, 130, 71));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
            }
        };

        JButton updateButton = new JButton("Update");
        updateButton.setUI(buttonUI);
        updateButton.setForeground(Color.WHITE);
        updateButton.setFont(new Font("Copperplate", Font.BOLD, 40));
```

```
        updateButton.setBounds(150, 170, 225, 300);
        updateButton.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onUpdateClicked();
            }

        });
        add(updateButton);

        JButton statisticsButton = new JButton("Statistics");
        statisticsButton.setUI(buttonUI);
        statisticsButton.setForeground(Color.WHITE);
        statisticsButton.setFont(new Font("Copperplate", Font.BOLD, 33));
        statisticsButton.setBounds(590, 170, 225, 300);
        statisticsButton.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onStatisticsClicked();
            }

        });
        add(statisticsButton);


    }
    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
    protected void paintComponent(Graphics g) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.setFont(new Font("Copperplate", Font.PLAIN, 32));
        graphics2d.drawString("Administrator Menu", 313, 80);

    }

}
```

```
package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.HeadlessException;
import java.awt.Label;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.image.BufferedImage;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Calendar;
import java.util.List;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;

import org.xml.sax.SAXException;

import com.group99.dom.BankAccountDomParser;
import com.group99.dom.ScreenSeatDomParser;
import com.group99.dom.TicketDomParser;
import com.group99.gui.CartDialogPanel.CartDialogPanelListener;
import com.group99.gui.PaymentDialogPanel.PaymentDialogPanelListener;
import com.group99.javabean.BankAccount;
import com.group99.javabean.Ticket;
/**
 * This is the control class of cart payment.
 * @author group 99
 *
 */
public class CartDialog extends JFrame {

    private ClientBackgroundPanel contentPane;
    private int mouseX, mouseY, jFrameX, jFrameY;
    private CartDialogListener listener;
    private int superFrameX, superFrameY;
    private BufferedImage movImg;
    private List<Ticket> tickets;
    /**
```

```
     * This is the callback interface of cart dialog.
     * @author group 99.
     *
     */
    public interface CartDialogListener {
        public void onCancelClicked();
    }
    /**
     * Set CartDialog's listener.
     * @param listener A listener of CartDialog.
     */
    public void setCartDialogListener(CartDialogListener listener) {
        this.listener = listener;
    }
    /**
     * This is the constructor of CartDialog.
     * @param tickets A List of Ticket.
     * @param movImg The Buffered Image of current movie.
     * @param superFrameX The x location of frame.
     * @param superFrameY The y location of frame.
     * @throws HeadlessException
     */
    public CartDialog(List<Ticket> tickets, BufferedImage movImg, int
superFrameX, int superFrameY)
            throws HeadlessException {

        this.tickets = tickets;
        this.movImg = movImg;
        this.superFrameX = superFrameX;
        this.superFrameY = superFrameY;

        setUndecorated(true);
        setLayout(null);
        setBackground(new Color(0, 0, 0, 0));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(313 + superFrameX, 80 + superFrameY, 640, 400);

        contentPane = new ClientBackgroundPanel("Cart");
        contentPane.addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                setLocation(jFrameX + (e.getXOnScreen() - mouseX), jFrameY +
(e.getYOnScreen() - mouseY));
            }
```

```
        });
        contentPane.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                mouseX = e.getXOnScreen();
                mouseY = e.getYOnScreen();
                jFrameX = getX();
                jFrameY = getY();
            }

        });
        contentPane.setOpaque(false);
        contentPane.setLayout(null);
        setContentPane(contentPane);

        CartDialogPanel cartDialogPanel = new CartDialogPanel(tickets,
movImg);
        cartDialogPanel.setBounds(0, 0, 640, 400);
        cartDialogPanel.setOpaque(false);
        cartDialogPanel.setCartDialogPanelListener(new
CartDialogPanelListener() {

            @Override
            public void onReelectClicked() {
                listener.onCancelClicked();
            }

            @Override
            public void onPayClicked(float payAmount) {
                // TODO Auto-generated method stub
                contentPane.remove(cartDialogPanel);
                contentPane.repaint();

                PaymentDialogPanel paymentDialogPanel = new
PaymentDialogPanel(movImg, payAmount,
                        PaymentDialogPanel.ENTER_HANDLE);
                paymentDialogPanel.setBounds(0, 0, 640, 400);
                paymentDialogPanel.setOpaque(false);
                paymentDialogPanel.setPaymentDialogPanelListener(new
PaymentDialogPanelListener() {

                    @Override
                    public void onConfirmClicked(String accountNum, String
accountPassword, float allPayAmount) {
```

```
                        List<BankAccount> bankAccounts = null;
                        try {
                            bankAccounts =
BankAccountDomParser.getBankAccounts();
                        } catch (ParserConfigurationException | SAXException
| IOException e) {
                            e.printStackTrace();
                        }
                        int count = 0;
                        for (BankAccount bankAccount : bankAccounts) {
                            if
(accountNum.equals(bankAccount.getAccountNum())) {
                                if
(accountPassword.equals(bankAccount.getAccountPassword())) {
                                    if (bankAccount.getBalance() >=
allPayAmount) {

    contentPane.remove(paymentDialogPanel);
                                        contentPane.repaint();
                                        PaymentDialogPanel askDialogPanel =
new PaymentDialogPanel(movImg, allPayAmount,

    PaymentDialogPanel.ASK_HANDLE);
                                        askDialogPanel.setBounds(0, 0, 640,
400);
                                        askDialogPanel.setOpaque(false);

    askDialogPanel.setPaymentDialogPanelListener(new
PaymentDialogPanelListener() {

                                            @Override
                                            public void
onConfirmClicked(String accountNum, String accountPassword,
                                                    float payAmount) {
                                                try {

    BankAccountDomParser.updateAccounts(bankAccount.getAccountNum(),
                                                        "balance",

    String.valueOf(bankAccount.getBalance() - payAmount));
                                                } catch
(ParserConfigurationException | SAXException | IOException
                                                        | TransformerException
e) {
```

```
                                                    e.printStackTrace();
                                                }
                                                String fileName =
tickets.get(0).getScreenName().toLowerCase().replace(" ", "") + "At"
                                                        +
tickets.get(0).getTimeStr().split(":")[0] + "_" +
tickets.get(0).getTimeStr().split(":")[1] + ".xml";
                                                try {
                                                    for (Ticket ticket :
tickets) {


    TicketDomParser.createTicket(ticket);


    ScreenSeatDomParser.updateScreenSeat(fileName,

    ticket.getSeatLocation(), "false");
                                                    }
                                                } catch
(ParserConfigurationException | SAXException | IOException
                                                        | TransformerException
e) {
                                                    e.printStackTrace();
                                                }


    contentPane.remove(askDialogPanel);
                                                contentPane.repaint();

                                                for(Ticket ticket : tickets){
                                                    FileWriter fw;
                                                    try {
                                                        fw = new
FileWriter("./TICKET_ID " + ticket.getTicketId() + ".txt");
                                                        BufferedWriter bufw =
new BufferedWriter(fw);
                                                        bufw.write("Ticket ID:
" + "   |   " + ticket.getTicketId());
                                                        bufw.newLine();
                                                        bufw.write("Film Name:
" + "   |   " + ticket.getFilmName());
                                                        bufw.newLine();
                                                        bufw.write("Screen
```

```
Name: " + "|   " + ticket.getScreenName());
                                                    bufw.newLine();
                                                    bufw.write("Seat
Location: " + "|   " + ticket.getSeatLocation());
                                                    bufw.newLine();
                                                    bufw.write("Time: " +
"       |   " + ticket.getTimeStr());
                                                    bufw.newLine();
                                                    bufw.write("Tiket
Type: " + "|   " + ticket.getTiketType());
                                                    bufw.newLine();

    if("student".equals(ticket.getTiketType())){

    bufw.write("Student ID: " + "  |   " + ticket.getStudentId());
                                                        bufw.newLine();
                                                    }
                                                    bufw.write("Ticket
Price: " + "    |   " + ticket.getTicketPrice());
                                                    bufw.newLine();
                                                    bufw.flush();
                                                    bufw.close();
                                                } catch (IOException e) {
                                                    e.printStackTrace();
                                                }

                                            }

                                            PaymentDialogPanel
successPaymentDialogPanel = new PaymentDialogPanel(
                                                    movImg, payAmount,
PaymentDialogPanel.SUCCESS_HANDLE);

    successPaymentDialogPanel.setBounds(0, 0, 640, 400);

    successPaymentDialogPanel.setOpaque(false);

    successPaymentDialogPanel.setPaymentDialogPanelListener(
                                                    new
PaymentDialogPanelListener() {

                                                        @Override
                                                        public void
onConfirmClicked(String accountNum,
```

```
                                                                String
accountPassword, float payAmount) {
                                                        }

                                                        @Override
                                                        public void
onCancelClicked() {

    listener.onCancelClicked();
                                                        }
                                                    });

    contentPane.add(successPaymentDialogPanel);
                                            contentPane.repaint();
                                        }

                                        @Override
                                        public void onCancelClicked() {
                                            listener.onCancelClicked();
                                        }
                                    });
                                    contentPane.add(askDialogPanel);
                                    contentPane.repaint();
                                } else {

    contentPane.remove(paymentDialogPanel);
                                    contentPane.repaint();
                                    PaymentDialogPanel nsfDialogPanel =
new PaymentDialogPanel(movImg, allPayAmount,

    PaymentDialogPanel.NSF_HANDLE);
                                    nsfDialogPanel.setBounds(0, 0, 640,
400);
                                    nsfDialogPanel.setOpaque(false);

    nsfDialogPanel.setPaymentDialogPanelListener(new
PaymentDialogPanelListener() {

                                        @Override
                                        public void
onConfirmClicked(String accountNum, String accountPassword,
                                                float payAmount) {

                                        }
```

```
                                        @Override
                                        public void onCancelClicked() {
                                            listener.onCancelClicked();
                                        }
                                    });
                                    contentPane.add(nsfDialogPanel);
                                    contentPane.repaint();
                                }
                                return;
                            }
                        }
                        if (count == bankAccounts.size() - 1) {
                            JLabel errorLabel = new JLabel("Your account
number or password is error !");
                            errorLabel.setFont(new Font("Arial",
Font.PLAIN, 11));
                            errorLabel.setForeground(Color.RED);
                            errorLabel.setOpaque(false);
                            errorLabel.setBounds(330, 300, 300, 20);
                            contentPane.add(errorLabel);
                            contentPane.repaint();
                        }
                        count++;
                    }
                }

                @Override
                public void onCancelClicked() {
                    listener.onCancelClicked();
                }
            });
            contentPane.add(paymentDialogPanel);
            contentPane.repaint();
        }
    });

    contentPane.add(cartDialogPanel);
    contentPane.repaint();
}

}

package com.group99.gui;
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JPanel;

import com.group99.javabean.Ticket;

/**
 * This is the boundary class of panel of cart dialog.
 * @author group 99
 *
 */
public class CartDialogPanel extends JPanel {

    private BufferedImage movImg;
    private List<Ticket> tickets;
    private float payAmount = 0;

    private CartDialogPanelListener listener;
    /**
     * This is the callback interface of CartDialogPanel.
     * @author group 99
     *
     */
    public interface CartDialogPanelListener{
        public void onReelectClicked();
        public void onPayClicked(float totalPrice);
    }
    /**
     * Set CartDialogPanel's listener.
     * @param listener A listener of CartDialogPanel.
     */
    public void setCartDialogPanelListener(CartDialogPanelListener
listener){
        this.listener = listener;
```

```
}
/**
 * This is the constructor of CartDialogPanel.
 * @param tickets A List of Ticket.
 * @param movImg The Buffered Image of current movie.
 */
public CartDialogPanel(List<Ticket> tickets, BufferedImage movImg) {

    this.tickets = tickets;
    this.movImg = movImg;
    initData();
}
/**
 * Initialize the data of CartDialogPanel.
 */
private void initData() {

    setLayout(null);

    JButton btnCancel = new JButton("Reelect");
    btnCancel.setFont(new Font("Arial", Font.BOLD, 15));
    btnCancel.setForeground(Color.WHITE);
    btnCancel.setOpaque(true);
    btnCancel.setBounds(35, 286, 108, 34);
    btnCancel.setUI(new ClientNormalButtonUI());
    btnCancel.addMouseListener(new MouseAdapter() {

        @Override
        public void mouseClicked(MouseEvent e) {
            listener.onReelectClicked();
        }

    });
    add(btnCancel);

    for(Ticket ticket : tickets){
        payAmount = payAmount + ticket.getTicketPrice();
    }

    JButton btnPay = new JButton("Pay");
    btnPay.setFont(new Font("Arial", Font.BOLD, 15));
    btnPay.setForeground(Color.WHITE);
    btnPay.setOpaque(true);
    btnPay.setBounds(35, 334, 108, 34);
```

```
        btnPay.setUI(new ClientNormalButtonUI());
        btnPay.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onPayClicked(payAmount);
            }

        });
        add(btnPay);

    }
    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
    public void paint(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        graphics2d.drawImage(movImg, 21, 37, 140, 207, null);

        graphics2d.drawLine(175, 40, 620, 40);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 16));
        graphics2d.drawString("Ticket ID", 180, 60);
        graphics2d.drawString("Screen", 275, 60);
        graphics2d.drawString("Location", 355, 60);
        graphics2d.drawString("Time", 450, 60);
        graphics2d.drawString("Type", 510, 60);
        graphics2d.drawString("Price", 575, 60);
        graphics2d.drawLine(175, 70, 620, 70);

        int i = 0;
        float totalPrice = 0;
        graphics2d.setFont(new Font("Arial", Font.ITALIC, 15));
        for(Ticket ticket : tickets){
            totalPrice = totalPrice + ticket.getTicketPrice();
            graphics2d.drawString(ticket.getTicketId(), 180, 100 + (25*i));
            graphics2d.drawString(ticket.getScreenName(), 275, 100 +
(25*i));
            graphics2d.drawString(ticket.getSeatLocation(), 377, 100 +
(25*i));
            graphics2d.drawString(ticket.getTimeStr(), 450, 100 + (25*i));
```

```
            graphics2d.drawString(ticket.getTiketType(), 510, 100 + (25*i));

    graphics2d.drawString(String.format("%.2f",ticket.getTicketPrice()),
580, 100 + (25*i));
            i++;
        }
        graphics2d.drawLine(175, 90 + (25*i), 620, 90 + (25*i));
        graphics2d.setFont(new Font("Arial", Font.BOLD, 15));
        graphics2d.drawString("Total Price:  $ " +
String.format("%.2f",totalPrice), 480, 110 + (25*i));
        super.paint(graphics2d);

    }



}

package com.group99.gui;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;

import javax.swing.JPanel;
/**
 * This is the boundary class of client background panel.
 * @author group 99.
 *
 */
public class ClientBackgroundPanel extends JPanel {

    private String title;

    public ClientBackgroundPanel(String title){
        this.title = title;
    }
    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
```

```
    protected void paintComponent(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.setColor(new Color(255, 255, 255, 230));
        graphics2d.fillRoundRect(1, 1, getWidth() - 2, getHeight() - 2, 20,
20);

        graphics2d.setColor(Color.WHITE);
        graphics2d.setClip(0, 0, getWidth(), 30);
        graphics2d.fillRoundRect(1, 3, getWidth() - 2, getHeight() - 1, 20,
20);
        graphics2d.setClip(null);

        graphics2d.setStroke(new BasicStroke(2));
        graphics2d.setColor(Color.GRAY);
        graphics2d.drawRoundRect(1, 1, getWidth() - 2, getHeight() - 2, 20,
20);

        graphics2d.setFont(new Font("Arial", Font.BOLD, 16));
        graphics2d.setColor(Color.DARK_GRAY);
        graphics2d.drawString(title, 15, 23);

    }

}

package com.group99.gui;

import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;

import org.xml.sax.SAXException;

import com.group99.dom.AdministratorDomParser;
import com.group99.dom.ScreenSeatDomParser;
import
```

```
com.group99.gui.AdministratorLoginPanel.AdministratorLoginPanelListener;
import
com.group99.gui.AdministratorMenuPanel.AdministratorMenuPanelListener;
import com.group99.gui.InfoPanel.InfoPanelListener;
import com.group99.gui.ModifyInfoPanel.ModifyInfoPanelListener;
import com.group99.gui.ModifyMenuPanel.ModifyMenuPanelListener;
import com.group99.gui.ModifyPanel.ModifyPanelListener;
import com.group99.gui.MoviesIntroPanel.MoviesIntroPanelListener;
import
com.group99.gui.Screen1SeatSelectPanel.Screen1SeatSelectPanelListener;
import
com.group99.gui.Screen2SeatSelectPanel.Screen2SeatSelectPanelListener;
import
com.group99.gui.Screen3SeatSelectPanel.Screen3SeatSelectPanelListener;
import com.group99.gui.ScreenSelectPanel.ScreenSelectPanelListener;
import com.group99.gui.StatisticsPanel.StatisticsPanelListener;
import com.group99.gui.WelcomePanel.WelcomePanelListener;
import com.group99.javabean.Administrator;
import com.group99.javabean.Screen;

import javax.swing.AbstractButton;
import javax.swing.JButton;

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.io.IOException;
import java.util.List;
import java.awt.Font;
import javax.swing.JLabel;
/**
 * This is the control class of kiosk's client.
 */
public class ClientFrame extends JFrame {

    private ClientBackgroundPanel contentPane;
    private int mouseX, mouseY, jFrameX, jFrameY;
    public static ClientFrame frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
```

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

    /**
     * Create kiosk's client.
     * @throws IOException
     * @throws SAXException
     * @throws ParserConfigurationException
     */
    public ClientFrame() throws ParserConfigurationException, SAXException,
IOException {

        setUndecorated(true);
        setBackground(new Color(0,0,0,0));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 960, 600);



        contentPane = new ClientBackgroundPanel("Client Frame");
        contentPane.addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                setLocation(jFrameX + (e.getXOnScreen() -mouseX), jFrameY +
(e.getYOnScreen() -mouseY));
            }
        });
        contentPane.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                mouseX = e.getXOnScreen();
                mouseY = e.getYOnScreen();
                jFrameX = getX();
                jFrameY = getY();
            }
```

```
        });
        contentPane.setOpaque(false);
        setContentPane(contentPane);

        JButton btnExit = new JButton("Exit");
        btnExit.setBounds(880, 3, 69, 24);
        btnExit.setUI(new ClientSpecialButtonUI(){

            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(250, 0, 0, 200));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight() + 5, 10,
10);
            }

        });
        btnExit.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                System.exit(0);
            }
        });

        JButton btnMin = new JButton("Min");
        btnMin.setBounds(800, 3, 69, 24);
        btnMin.setUI(new ClientSpecialButtonUI());
        btnMin.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                setExtendedState(JFrame.ICONIFIED);
            }
        });

        contentPane.setLayout(null);
        contentPane.add(btnMin);
        contentPane.add(btnExit);

        showWelcomePanel();
    }
    /**
     * Display the layout of certain screen selected by customers and
customers can also buy tickets by clicking related seats.
     * @param movName The name of movie selected by customers.
```

```
     * @param screenName The screen name selected by customers.
     * @param timeStr The time selected by customers.
     * @throws IOException
     * @throws ParserConfigurationException
     * @throws SAXException
     */
    public void showScreenSeatSelectPanel(String movName, String
screenName,String timeStr) throws IOException,
ParserConfigurationException, SAXException{

        if("1".equals(screenName)){
            Screen1SeatSelectPanel screen1SeatSelectPanel = new
Screen1SeatSelectPanel(movName, timeStr, frame);
            screen1SeatSelectPanel.setBounds(5, 35, 956, 562);
            screen1SeatSelectPanel.setOpaque(false);
            screen1SeatSelectPanel.setScreen1SeatSelectPanelListener(new
Screen1SeatSelectPanelListener() {

                @Override
                public void onBtnBackClicked() {
                    contentPane.remove(screen1SeatSelectPanel);
                    contentPane.repaint();
                    try {
                        showScreenSelectPanel(movName, frame);
                    } catch (ParserConfigurationException e) {
                        e.printStackTrace();
                    } catch (SAXException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                @Override
                public void repaintPanel(String movName, String screenName,
String timeStr) {
                    contentPane.remove(screen1SeatSelectPanel);
                    contentPane.repaint();
                    try {
                        showScreenSeatSelectPanel(movName, screenName,
timeStr);
                    } catch (IOException | ParserConfigurationException |
SAXException e) {
                        e.printStackTrace();
```

```
                }
            }
        });
        contentPane.add(screen1SeatSelectPanel);
    }else if("2".equals(screenName)){
        Screen2SeatSelectPanel screen2SeatSelectPanel = new
Screen2SeatSelectPanel(movName, timeStr, frame);
        screen2SeatSelectPanel.setBounds(5, 35, 956, 562);
        screen2SeatSelectPanel.setOpaque(false);
        screen2SeatSelectPanel.setScreen2SeatSelectPanelListener(new
Screen2SeatSelectPanelListener() {
            @Override
            public void onBtnBackClicked() {
                contentPane.remove(screen2SeatSelectPanel);
                contentPane.repaint();
                try {
                    showScreenSelectPanel(movName, frame);
                } catch (ParserConfigurationException e) {
                    e.printStackTrace();
                } catch (SAXException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void repaintPanel(String movName, String screenName,
String timeStr) {
                contentPane.remove(screen2SeatSelectPanel);
                contentPane.repaint();
                try {
                    showScreenSeatSelectPanel(movName, screenName,
timeStr);
                } catch (IOException | ParserConfigurationException |
SAXException e) {
                    e.printStackTrace();
                }
            }
        });
        contentPane.add(screen2SeatSelectPanel);
    }else if("3".equals(screenName)){
        Screen3SeatSelectPanel screen3SeatSelectPanel = new
Screen3SeatSelectPanel(movName, timeStr, frame);
```

```
            screen3SeatSelectPanel.setBounds(5, 35, 956, 562);
            screen3SeatSelectPanel.setOpaque(false);
            screen3SeatSelectPanel.setScreen3SeatSelectPanelListener(new
Screen3SeatSelectPanelListener() {
                @Override
                public void onBtnBackClicked() {
                    contentPane.remove(screen3SeatSelectPanel);
                    contentPane.repaint();
                    try {
                        showScreenSelectPanel(movName, frame);
                    } catch (ParserConfigurationException e) {
                        e.printStackTrace();
                    } catch (SAXException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                @Override
                public void repaintPanel(String movName, String screenName,
String timeStr) {
                    contentPane.remove(screen3SeatSelectPanel);
                    contentPane.repaint();
                    try {
                        showScreenSeatSelectPanel(movName, screenName,
timeStr);
                    } catch (IOException | ParserConfigurationException |
SAXException e) {
                        e.printStackTrace();
                    }
                }
            });
            contentPane.add(screen3SeatSelectPanel);
        }

    }

    /**
     * Display timetable of films, the screen and related time will be
selected by customers.
     * @param selectName The name of film selected by customers.
     * @param frame Current frame.
     * @throws ParserConfigurationException
```

```
     * @throws SAXException
     * @throws IOException
     */
    public void showScreenSelectPanel(String selectName,JFrame frame)
throws ParserConfigurationException, SAXException, IOException{
        ScreenSelectPanel screenSelectPanel = new
ScreenSelectPanel(selectName, frame);
        screenSelectPanel.setBounds(5, 35, 956, 562);
        screenSelectPanel.setOpaque(false);

        screenSelectPanel.setScreenSelectPanelListener(new
ScreenSelectPanelListener() {

            @Override
            public void onBtnBackClicked() {
                contentPane.remove(screenSelectPanel);
                contentPane.repaint();
                try {
                    showMoviesIntroPanel();
                } catch (ParserConfigurationException e) {
                    e.printStackTrace();
                } catch (SAXException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onTimeSelectClicked(String screenName, String
timeStr) {
                try {
                    contentPane.remove(screenSelectPanel);
                    contentPane.repaint();

    showScreenSeatSelectPanel(selectName,screenName,timeStr);
                } catch (IOException | ParserConfigurationException |
SAXException e) {
                    e.printStackTrace();
                }
            }
        });

        contentPane.add(screenSelectPanel);
```

```
    }
    /**
     * Display detailed information of films.
     * @param selectedMovieName The name of movie selected by customers.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public void showInfoPanel(String selectedMovieName) throws
ParserConfigurationException, SAXException, IOException{
        InfoPanel infoPanel = new InfoPanel(selectedMovieName);
        infoPanel.setBounds(5, 35, 956, 562);
        infoPanel.setOpaque(false);

        infoPanel.setInfoPanelListener(new InfoPanelListener() {

            @Override
            public void onBtnPurchaseClicked() {
                contentPane.remove(infoPanel);
                contentPane.repaint();
                try {
                    showScreenSelectPanel(selectedMovieName, frame);
                } catch (ParserConfigurationException e) {
                    e.printStackTrace();
                } catch (SAXException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onBtnBackClicked() {
                contentPane.remove(infoPanel);
                contentPane.repaint();
                try {
                    showMoviesIntroPanel();
                } catch (ParserConfigurationException e) {
                    e.printStackTrace();
                } catch (SAXException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
```

```
            }
        });
        contentPane.add(infoPanel);
    }
    /**
     * Display brief information of films and select certain movies.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public void showMoviesIntroPanel() throws ParserConfigurationException,
SAXException, IOException{
        MoviesIntroPanel moviesInroPanel = new MoviesIntroPanel();
        moviesInroPanel.setBounds(5, 35, 956, 562);
        moviesInroPanel.setOpaque(false);
        moviesInroPanel.setMoviesIntroPanelListener(new
MoviesIntroPanelListener(){
            @Override
            public void onImageClicked(String filmName) {
                contentPane.remove(moviesInroPanel);
                contentPane.repaint();
                try {
                    showInfoPanel(filmName);
                } catch (ParserConfigurationException e) {
                    e.printStackTrace();
                } catch (SAXException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onHomeClicked() {
                contentPane.remove(moviesInroPanel);
                contentPane.repaint();
                showWelcomePanel();
            }
        });
        contentPane.add(moviesInroPanel);

    }
    /**
     * Set all seats in all screens to empty.
```

```
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     * @throws TransformerException
     */
    public void resetSeat() throws ParserConfigurationException,
SAXException, IOException, TransformerException{

        String[] seatFilesName = {

    "screen1At10_00.xml","screen1At12_30.xml","screen1At15_30.xml","screen1
At18_30.xml","screen1At21_00.xml",

    "screen2At10_30.xml","screen2At13_00.xml","screen2At16_00.xml","screen2
At18_00.xml","screen2At20_00.xml",

    "screen3At10_30.xml","screen3At13_00.xml","screen3At15_30.xml","screen3
At18_00.xml","screen3At20_30.xml",
                };

        for(int i = 0; i < seatFilesName.length; i ++){
            List<Screen> screens1 =
ScreenSeatDomParser.getScreen(seatFilesName[i]);
            for (Screen screen : screens1) {
                ScreenSeatDomParser.updateScreenSeat(seatFilesName[i],
screen.getSeatId(), "true");
            }
        }

    }
    /**
     * Display the information of modifying films.
     * @param selectedMovieName The name of movie whose information is
updated.
     */
    public void showModifyInfoPanel(String selectedMovieName){
        try {
            ModifyInfoPanel modifyInfoPanel = new
ModifyInfoPanel(selectedMovieName);
            modifyInfoPanel.setOpaque(false);
            modifyInfoPanel.setBounds(5, 35, 956, 562);
            modifyInfoPanel.setModifyInfoPanelListener(new
ModifyInfoPanelListener() {
                @Override
```

```
                public void onBackClicked() {
                    contentPane.remove(modifyInfoPanel);
                    contentPane.repaint();
                    showModifyPanel();
                }
            });
            contentPane.add(modifyInfoPanel);
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            e.printStackTrace();
        }

    }
    /**
     * Display several movies and select which film information will be
updated.
     */
    public void showModifyPanel() {
        try {
            ModifyPanel modifyPanel = new ModifyPanel();
            modifyPanel.setOpaque(false);
            modifyPanel.setBounds(5, 35, 956, 562);
            modifyPanel.setModifyPanelListener(new ModifyPanelListener() {

                @Override
                public void onImageClicked(String filmName) {
                    contentPane.remove(modifyPanel);
                    contentPane.repaint();
                    showModifyInfoPanel(filmName);
                }

                @Override
                public void onBackClicked() {
                    contentPane.remove(modifyPanel);
                    contentPane.repaint();
                    try {
                        showModifyMenuPanel();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
            contentPane.add(modifyPanel);
```

```
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            e.printStackTrace();
        }

    }
    /**
     * Display the menu of modifying mode.
     * @throws IOException
     */
    public void showModifyMenuPanel() throws IOException{
        ModifyMenuPanel modifyMenuPanel = new ModifyMenuPanel();
        modifyMenuPanel.setOpaque(false);
        modifyMenuPanel.setBounds(5, 35, 956, 562);
        modifyMenuPanel.setModifyMenuPanelListener(new
ModifyMenuPanelListener() {

            @Override
            public void onBackClicked() {
                contentPane.remove(modifyMenuPanel);
                contentPane.repaint();
                try {
                    showAdministratorMenuSelect();
                } catch (ParserConfigurationException | SAXException |
IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onResetSeatClicked() {
                try {
                    resetSeat();
                } catch (ParserConfigurationException | SAXException |
IOException | TransformerException e) {
                    e.printStackTrace();
                }

                JLabel resetSucessLabel = new JLabel("Reset
successfully !");
                resetSucessLabel.setFont(new Font("Arial", Font.PLAIN, 15));
                resetSucessLabel.setOpaque(false);
                resetSucessLabel.setForeground(Color.RED);
                resetSucessLabel.setBounds(410, 430, 180, 20);
```

```
                modifyMenuPanel.add(resetSucessLabel);
                modifyMenuPanel.repaint();
            }

            @Override
            public void onModifyFilmInfoClicked() {
                contentPane.remove(modifyMenuPanel);
                contentPane.repaint();
                showModifyPanel();
            }
        });
        add(modifyMenuPanel);
    }
    /**
     * Display the statistical report.
     * @throws IOException
     */
    public void showStatisticsPanel() throws IOException{
        StatisticsPanel statisticsPanel = new StatisticsPanel();
        statisticsPanel.setOpaque(false);
        statisticsPanel.setBounds(5, 35, 956, 562);
        statisticsPanel.setStatisticsPanelListener(new
StatisticsPanelListener() {

            @Override
            public void onBackClicked() {
                contentPane.remove(statisticsPanel);
                contentPane.repaint();
                try {
                    showAdministratorMenuSelect();
                } catch (ParserConfigurationException | SAXException |
IOException e) {
                    e.printStackTrace();
                }
            }
        });

        add(statisticsPanel);
    }
    /**
     * The main menu of administrator mode.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
```

```
     */
    public void showAdministratorMenuSelect() throws
ParserConfigurationException, SAXException, IOException{
        AdministratorMenuPanel administratorMenuPanel = new
AdministratorMenuPanel();
        administratorMenuPanel.setBounds(5, 35, 956, 562);
        administratorMenuPanel.setOpaque(false);
        administratorMenuPanel.setAdministratorMenuPanelListener(new
AdministratorMenuPanelListener() {

            @Override
            public void onUpdateClicked() {
                contentPane.remove(administratorMenuPanel);
                contentPane.repaint();
                try {
                    showModifyMenuPanel();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onStatisticsClicked() {
                contentPane.remove(administratorMenuPanel);
                contentPane.repaint();
                try {
                    showStatisticsPanel();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onHomeClicked() {
                contentPane.remove(administratorMenuPanel);
                contentPane.repaint();
                showWelcomePanel();
            }
        });

        contentPane.add(administratorMenuPanel);
    }
    /**
     * Display the interface of logging in by administrator.
```

```
     * @param id The id of administrator.
     * @param password The password of administrator's account.
     * @param isErrorHint Whether the panel generate error message or not.
     */
    public void showAdministratorLogin(String id, String password, boolean
isErrorHint) {
        AdministratorLoginPanel administratorLoginPanel = new
AdministratorLoginPanel(id, password, isErrorHint);
        administratorLoginPanel.setBounds(5, 35, 956, 562);
        administratorLoginPanel.setOpaque(false);
        administratorLoginPanel.setAdministratorLoginPanelListener(new
AdministratorLoginPanelListener() {

            @Override
            public void onLoginCicked(String id, String password) {
                try {
                    List<Administrator> administrators =
AdministratorDomParser.getAdministrators();
                    int count = 0;
                    for(Administrator administrator : administrators){
                        if(id.equals(administrator.getId())){

    if(password.equals(administrator.getPassword())){
                                contentPane.remove(administratorLoginPanel);
                                contentPane.repaint();
                                showAdministratorMenuSelect();
                                return;
                            }
                        }
                        if(count == administrators.size() - 1){
                            contentPane.remove(administratorLoginPanel);
                            contentPane.repaint();
                            showAdministratorLogin(id, password, true);
                            return;
                        }
                        count++;
                    }
                } catch (ParserConfigurationException | SAXException |
IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
```

```
            public void onBackClicked() {
                contentPane.remove(administratorLoginPanel);
                contentPane.repaint();
                showWelcomePanel();
            }
        });

        contentPane.add(administratorLoginPanel);
    }
    /**
     * Display the welcome interface. Customer mode or Administrator mode
can be selected.
     */
    public void showWelcomePanel() {
        WelcomePanel welcomePanel = new WelcomePanel();
        welcomePanel.setBounds(5, 35, 956, 562);
        welcomePanel.setOpaque(false);
        welcomePanel.setWelcomePanelListener(new WelcomePanelListener() {

            @Override
            public void onCustomerClicked() {
                contentPane.remove(welcomePanel);
                contentPane.repaint();
                try {
                    showMoviesIntroPanel();
                } catch (ParserConfigurationException | SAXException |
IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onAdministratorClicked() {
                contentPane.remove(welcomePanel);
                contentPane.repaint();
                showAdministratorLogin(null, null, false);
            }
        });

        contentPane.add(welcomePanel);
    }
}

package com.group99.gui;
```

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;

import javax.swing.AbstractButton;
import javax.swing.JComponent;
import javax.swing.LookAndFeel;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of normal client button UI.
 * @author group 99
 *
 */
public class ClientNormalButtonUI extends BasicButtonUI {

    @Override
    protected void paintText(Graphics g, AbstractButton b, Rectangle
textRect, String text) {
        super.paintText(g, b, textRect, text);
    }
    /**
     * Override the method of void installDefaults(AbstractButton b) to set
the LookAndFeel.
     */
    @Override
    protected void installDefaults(AbstractButton b) {
        super.installDefaults(b);
        LookAndFeel.installProperty(b, "opaque", Boolean.FALSE);
    }
    /**
     * Override the method of void paint(Graphics g, JComponent c) to set
the view when button isn't pressed.
     */
    @Override
    public void paint(Graphics g, JComponent c) {
        g.setColor(new Color(0, 91, 255));
        g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
        super.paint(g, c);
    }
    /**
     * Override the method of void paintButtonPressed(Graphics g,
AbstractButton b) to set the view when button is pressed.
     */
```

```
    @Override
    protected void paintButtonPressed(Graphics g, AbstractButton b) {
        g.setColor(new Color(255, 130, 71));
        g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
    }

}

package com.group99.gui;

import java.awt.Color;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Rectangle;

import javax.swing.AbstractButton;
import javax.swing.ButtonModel;
import javax.swing.JComponent;
import javax.swing.LookAndFeel;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of special client button UI.
 * @author group 99
 *
 */
public class ClientSpecialButtonUI extends BasicButtonUI {

    /**
     * Override the method of void installDefaults(AbstractButton b) to set
the LookAndFeel.
     */
    @Override
    protected void installDefaults(AbstractButton b) {
        super.installDefaults(b);
        LookAndFeel.installProperty(b, "opaque", Boolean.FALSE);
    }
    /**
     * Override the method of void paint(Graphics g, JComponent c) to set
the view when button isn't pressed.
     */
    @Override
    public void paint(Graphics g, JComponent c) {
        g.setColor(new Color(245, 245, 245, 200));
        g.fillRoundRect(0, 0, c.getWidth(), c.getHeight() + 5, 10, 10);
```

```
        super.paint(g, c);
    }
    /**
     * Override the method of void paintButtonPressed(Graphics g,
AbstractButton b) to set the view when button is pressed.
     */
    @Override
    protected void paintButtonPressed(Graphics g, AbstractButton b) {
        g.setColor(new Color(0, 91, 255, 200));
        g.fillRoundRect(0, 0, b.getWidth(), b.getHeight() + 5, 10, 10);
    }

}

package com.group99.gui;

import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.text.AttributedCharacterIterator;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JTextPane;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.javabean.Film;
/**
 * This is the boundary class of panel of movie information.
 * @author group 99
 *
```

```
 */
public class InfoPanel extends JPanel {
    private BufferedImage selectedMovieImg;
    private String movieImgName;
    private String selectedMovieName;
    private String selectedDirector;
    private String selectedStars;
    private String selectedName;
    private String selectedPrice;
    private String selectedDuration;
    private String selectedStoryline;

    private InfoPanelListener listener;
    /**
     * This is the callback interface of InfoPanel.
     * @author group 99
     *
     */
    public interface InfoPanelListener{
        public void onBtnBackClicked();
        public void onBtnPurchaseClicked();
    }
    /**
     * Set InfoPanel's listener.
     * @param listener A listener of InfoPanel.
     */
    public void setInfoPanelListener(InfoPanelListener listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of InfoPanel.
     * @param selectedMovieName The name of selecting movie.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public InfoPanel(String selectedMovieName) throws
ParserConfigurationException, SAXException, IOException {

        this.selectedMovieName = selectedMovieName;
        initData();

    }
    /**
```

```
     * Initialize the data of InfoPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    private void initData() throws ParserConfigurationException,
SAXException, IOException {

        setLayout(null);

        List<Film> films = FilmDomParser.getFilms();

        for(Film film : films){
            if(film.getFilmName().equals(selectedMovieName)){
                movieImgName = selectedMovieName;
                selectedDirector = film.getFilmDirector();
                selectedStars = film.getFilmStars();
                selectedStoryline = film.getFilmStoryline();
                selectedName = film.getFilmName();
                selectedPrice = film.getFilmPrice().toString();
                selectedDuration = film.getFilmDuration() + "";
                selectedMovieImg = ImageIO.read(new
FileInputStream("./../../resource/" + movieImgName + ".jpg"));
            }
        }

        JTextPane storyline = new JTextPane();
        storyline.setEditable(false);
        storyline.setFont(new Font("Arial", Font.ITALIC, 15));
        storyline.setAlignmentY(Component.BOTTOM_ALIGNMENT);
        storyline.setOpaque(false);
        storyline.setText(selectedStoryline);
        storyline.setBounds(480, 207, 403, 350);
        add(storyline);

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBtnBackClicked();
            }
        });
```

```
        btnBack.setOpaque(true);
        btnBack.setBounds(50, 365, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        add(btnBack);

        JButton btnPurchase = new JButton("Purchase");
        btnPurchase.setForeground(Color.WHITE);
        btnPurchase.setFont(new Font("Arial", Font.BOLD, 20));
        btnPurchase.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBtnPurchaseClicked();
            }
        });
        btnPurchase.setOpaque(true);
        btnPurchase.setBounds(50, 435, 154, 48);
        btnPurchase.setUI(new ClientNormalButtonUI());
        add(btnPurchase);

    }

    public String getSelectedName() {
        return selectedName;
    }

    public void setSelectedName(String selectedName) {
        this.selectedName = selectedName;
    }

    public BufferedImage getSelectedMovieImg() {
        return selectedMovieImg;
    }

    public void setSelectedMovieImg(BufferedImage selectedMovieImg) {
        this.selectedMovieImg = selectedMovieImg;
    }

    public String getSelectedStoryline() {
        return selectedStoryline;
    }

    public void setSelectedStoryline(String selectedStoryline) {
        this.selectedStoryline = selectedStoryline;
    }
```

```
    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
    protected void paintComponent(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.drawImage(selectedMovieImg, 30, 10, 200, 296, null);
        graphics2d.drawLine(300, 5, 300, 500);

        graphics2d.setFont(new Font("Arial", Font.BOLD, 15));
        graphics2d.drawString("Film Name : " + selectedName, 360, 20);
        graphics2d.drawString("Film Duration : " + selectedDuration + "
MINUTES", 360, 60);
        graphics2d.drawString("Film Price : $ " + selectedPrice, 360, 100);
        graphics2d.drawString("Film Director : " + selectedDirector, 360,
140);
        graphics2d.drawString("Film Stars : " + selectedStars, 360, 180);
        graphics2d.drawString("Film Storyline : ", 360, 220);

    }
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.text.AttributedCharacterIterator;
import java.util.List;
import java.util.regex.Pattern;

import javax.imageio.ImageIO;
```

```
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.javabean.Film;
/**
 * This is the boundary class of panel of modifying information of movie.
 * @author group 99
 *
 */
public class ModifyInfoPanel extends JPanel {
    private BufferedImage selectedMovieImg;
    private String selectedMovieName;
    private String selectedPrice;
    private String selectedScore;

    private ModifyInfoPanelListener listener;
    /**
     * This is the callback interface of ModifyInfoPanel.
     * @author group 99
     *
     */
    public interface ModifyInfoPanelListener {
        public void onBackClicked();
    }
    /**
     * Set ModifyInfoPanel's listener.
     * @param listener A listener of ModifyInfoPanel.
     */
    public void setModifyInfoPanelListener(ModifyInfoPanelListener
listener) {
        this.listener = listener;
    }
    /**
     * This is the constructor of ModifyInfoPanel.
     * @param selectedMovieName The name of selecting movie.
     * @throws ParserConfigurationException
```

```
     * @throws SAXException
     * @throws IOException
     */
    public ModifyInfoPanel(String selectedMovieName) throws
ParserConfigurationException, SAXException, IOException {

        this.selectedMovieName = selectedMovieName;
        initData();

    }
    /**
     * Initialize the data of ModifyInfoPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    private void initData() throws ParserConfigurationException,
SAXException, IOException {

        List<Film> films = FilmDomParser.getFilms();

        for(Film film : films){
            if(film.getFilmName().equals(selectedMovieName)){
                selectedPrice = film.getFilmPrice().toString();
                selectedScore = film.getFilmScore();
            }
        }

        selectedMovieImg = ImageIO.read(new
FileInputStream("./../../resource/" + selectedMovieName + ".jpg"));

        setLayout(null);

        JTextField priceTextField = new JTextField(15);
        priceTextField.setBounds(500, 207, 330, 55);
        priceTextField.setText("The old price: " + selectedPrice);
        priceTextField.setFont(new Font("Arial", Font.PLAIN, 19));
        priceTextField.setOpaque(false);
        add(priceTextField);

        JTextField scoreTextField = new JTextField(15);
        scoreTextField.setBounds(500, 327, 330, 55);
        scoreTextField.setText("The old score: " + selectedScore);
        scoreTextField.setFont(new Font("Arial", Font.PLAIN, 19));
```

```
        scoreTextField.setOpaque(false);
        add(scoreTextField);

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBackClicked();
            }
        });
        btnBack.setOpaque(true);
        btnBack.setBounds(50, 365, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        add(btnBack);

        JLabel errorLabel = new JLabel("The price must be number.The score
must be less than 10 !");
        errorLabel.setOpaque(false);
        errorLabel.setFont(new Font("Arial", Font.PLAIN, 15));
        errorLabel.setForeground(Color.RED);
        errorLabel.setBounds(502, 400, 450, 20);

        JButton btnSubmit = new JButton("Submit");
        btnSubmit.setForeground(Color.WHITE);
        btnSubmit.setFont(new Font("Arial", Font.BOLD, 20));
        btnSubmit.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                if(isNumeric(scoreTextField.getText()) &&
isNumeric(priceTextField.getText())){
                    if(Float.parseFloat(scoreTextField.getText()) <= 10){
                        try {
                            FilmDomParser.updateFilms(selectedMovieName,
"filmPrice", priceTextField.getText());
                            FilmDomParser.updateFilms(selectedMovieName,
"filmScore", scoreTextField.getText());
                            JLabel successLabel = new JLabel("Update
successfully ! ");
                            successLabel.setOpaque(false);
                            successLabel.setFont(new Font("Arial",
Font.PLAIN, 15));
                            successLabel.setForeground(Color.RED);
```

```
                        successLabel.setBounds(58, 510, 450, 20);
                        add(successLabel);
                        repaint();
                    } catch (ParserConfigurationException | SAXException
| IOException | TransformerException e1) {
                        e1.printStackTrace();
                    }
                }else{
                    add(errorLabel);
                    repaint();
                }
            }else{
                add(errorLabel);
                repaint();
            }
        }
    });
    btnSubmit.setOpaque(true);
    btnSubmit.setBounds(50, 435, 154, 48);
    btnSubmit.setUI(new ClientNormalButtonUI());
    add(btnSubmit);

}

public static boolean isNumeric(String str){
    if (null == str || "".equals(str)) {
        return false;
    }
    Pattern pattern = Pattern.compile("^[-\\+]?[.\\d]*$");
    return pattern.matcher(str).matches();
}
/**
 * Override the method of void paintComponent(Graphics g).
 */
@Override
protected void paintComponent(Graphics g) {

    Graphics2D graphics2d = (Graphics2D) g;
    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
    graphics2d.drawImage(selectedMovieImg, 30, 10, 200, 296, null);
    graphics2d.drawLine(300, 5, 300, 500);

    graphics2d.setFont(new Font("Copperplate", Font.BOLD, 32));
```

```
        graphics2d.drawString("Modify Info", 510, 50);

        graphics2d.setFont(new Font("Arial", Font.BOLD, 20));
        graphics2d.drawString("New Price: ", 380, 240);

        graphics2d.drawString("New Score: ", 380, 360);

        super.paintComponent(graphics2d);
    }
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JPanel;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.TicketDomParser;
import com.group99.javabean.Ticket;

/**
 * This is the boundary class of panel of menu of modifying movie.
 * @author group 99
 *
 */
public class ModifyMenuPanel extends JPanel {
```

```
    BufferedImage image1, image2, image3, image4, image5;

    private ModifyMenuPanelListener listener;
    /**
     * This is the callback interface of ModifyMenuPanel.
     * @author group 99
     *
     */
    public interface ModifyMenuPanelListener{
        public void onBackClicked();
        public void onResetSeatClicked();
        public void onModifyFilmInfoClicked();
    }
    /**
     * Set ModifyMenuPanel's listener.
     * @param listener A listener of ModifyMenuPanel.
     */
    public void setModifyMenuPanelListener(ModifyMenuPanelListener
listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of ModifyMenuPanel.
     * @throws IOException
     */
    public ModifyMenuPanel() throws IOException {
        initData();
    }
    /**
     * Initialize the data of ModifyMenuPanel.
     * @throws IOException
     */
    private void initData() throws IOException {

        image1 = ImageIO.read(new FileInputStream("./../../resource/LA LA
LAND.jpg"));
        image2 = ImageIO.read(new FileInputStream("./../../resource/" +
"KONG-SKULL ISLAND" + ".jpg"));
        image3 = ImageIO.read(new FileInputStream("./../../resource/" +
"BEAUTY AND THE BEAST" + ".jpg"));
        image4 = ImageIO.read(new FileInputStream("./../../resource/" +
"LOGAN" + ".jpg"));
        image5 = ImageIO.read(new FileInputStream("./../../resource/" +
```

```
"MOONLIGHT" + ".jpg"));

        setLayout(null);

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.setOpaque(true);
        btnBack.setBounds(70, 415, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        btnBack.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBackClicked();
            }

        });
        add(btnBack);

        BasicButtonUI buttonUI = new BasicButtonUI(){
            @Override
            public void paint(Graphics g, JComponent c) {
                g.setColor(new Color(30, 144, 255));
                g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
                super.paint(g, c);
            }

            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(255, 130, 71));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
            }
        };

        JButton updateButton = new JButton("Reset Seat");
        updateButton.setUI(buttonUI);
        updateButton.setForeground(Color.WHITE);
        updateButton.setFont(new Font("Copperplate", Font.BOLD, 23));
        updateButton.setBounds(390, 170, 180, 240);
        updateButton.addMouseListener(new MouseAdapter() {

            @Override
```

```
            public void mouseClicked(MouseEvent e) {
                listener.onResetSeatClicked();
            }

        });
        add(updateButton);

        JButton modifyButton = new JButton("Modify Film");
        modifyButton.setUI(buttonUI);
        modifyButton.setForeground(Color.WHITE);
        modifyButton.setFont(new Font("Copperplate", Font.BOLD, 22));
        modifyButton.setBounds(700, 170, 180, 240);
        modifyButton.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onModifyFilmInfoClicked();;
            }

        });
        add(modifyButton);

    }
    /**
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.setFont(new Font("Copperplate", Font.BOLD, 32));
        graphics2d.drawString("Modification", 40, 50);

        graphics2d.drawImage(image4, 47, 130, 60, 89, null);
        graphics2d.drawImage(image1, 202, 130, 60, 89, null);
        graphics2d.drawImage(image3, 73, 170, 70, 104, null);
        graphics2d.drawImage(image5, 165, 170, 70, 104, null);
        graphics2d.drawImage(image2, 112, 210, 80, 119, null);
        graphics2d.drawLine(300, 5, 300, 530);

        graphics2d.setFont(new Font("Arial", Font.BOLD, 25));
```

```
        graphics2d.drawString("You want to : ", 320, 50);

        super.paint(graphics2d);


    }



}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.javabean.Film;
/**
 * This is the boundary class of panel of modifying movie.
 * @author group 99
 *
 */
public class ModifyPanel extends JPanel {
```

```
    private int BTN_IMAGE_SIZE_WIDTH = 100;
    private int BTN_IMAGE_SIZE_HEIGHT = 148;
    BufferedImage image1, image2, image3, image4, image5;

    List<Film> films = null;
    List<BufferedImage> filmsImage = new ArrayList<>();
    ModifyPanelListener listener;
    /**
     * This is the callback interface of ModifyPanel.
     * @author group 99
     *
     */
    public interface ModifyPanelListener{
        public void onBackClicked();
        public void onImageClicked(String filmName);
    }
    /**
     * Set ModifyPanel's listener.
     * @param listener A listener of ModifyPanel.
     */
    public void setModifyPanelListener(ModifyPanelListener listener){
        this.listener = listener;
    }

    /**
     * This is the constructor of ModifyPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public ModifyPanel() throws ParserConfigurationException, SAXException,
IOException {
        initData();
    }
    /**
     * Initialize the data of ModifyPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    private void initData() throws ParserConfigurationException,
SAXException, IOException {
```

```
        image1 = ImageIO.read(new FileInputStream("./../../resource/LA LA
LAND.jpg"));
        image2 = ImageIO.read(new FileInputStream("./../../resource/" +
"KONG-SKULL ISLAND" + ".jpg"));
        image3 = ImageIO.read(new FileInputStream("./../../resource/" +
"BEAUTY AND THE BEAST" + ".jpg"));
        image4 = ImageIO.read(new FileInputStream("./../../resource/" +
"LOGAN" + ".jpg"));
        image5 = ImageIO.read(new FileInputStream("./../../resource/" +
"MOONLIGHT" + ".jpg"));

        int count = 0;
        List<Film> films = FilmDomParser.getFilms();

        setLayout(null);

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.setOpaque(true);
        btnBack.setBounds(70, 415, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        btnBack.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBackClicked();
            }

        });
        add(btnBack);

        for(Film film : films){
            String filmName = film.getFilmName();
            String filmType = film.getFilmType();
            String filmScore = film.getFilmScore();
            BufferedImage image = ImageIO.read(new
FileInputStream("./../../resource/" + filmName + ".jpg"));

            JButton button = new JButton();
            button.setUI(new BasicButtonUI(){
                @Override
                public void paint(Graphics g, JComponent c) {
                    Graphics2D graphics2d = (Graphics2D) g;
```

```
    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                    graphics2d.drawImage(image, 0, 0, BTN_IMAGE_SIZE_WIDTH,
BTN_IMAGE_SIZE_HEIGHT, null);
                }
            });
            button.setBounds(325 + count*125, 240, BTN_IMAGE_SIZE_WIDTH,
BTN_IMAGE_SIZE_HEIGHT);
            button.addMouseListener(new MouseAdapter(){
                @Override
                public void mouseClicked(MouseEvent e) {
                    listener.onImageClicked(filmName);
                }
            });
            add(button);
            count++;
        }
    }
    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
    protected void paintComponent(Graphics g) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.setFont(new Font("Copperplate", Font.BOLD, 32));
        graphics2d.drawString("Modify", 85, 50);

        graphics2d.drawImage(image4, 47, 130, 60, 89, null);
        graphics2d.drawImage(image1, 202, 130, 60, 89, null);
        graphics2d.drawImage(image3, 73, 170, 70, 104, null);
        graphics2d.drawImage(image5, 165, 170, 70, 104, null);
        graphics2d.drawImage(image2, 112, 210, 80, 119, null);
        graphics2d.drawLine(300, 5, 300, 530);
        graphics2d.setFont(new Font("Copperplate", Font.PLAIN, 25));
        graphics2d.drawString("You want to modify: ", 345, 80);

        super.paintComponents(graphics2d);
    }

}
```

```
package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.javabean.Film;
/**
 * This is the boundary class of panel of movies' brief information.
 * @author group 99
 *
 */
public class MoviesIntroPanel extends JPanel {

    private int BTN_IMAGE_SIZE_WIDTH = 150;
    private int BTN_IMAGE_SIZE_HEIGHT = 222;

    List<Film> films = null;
    List<BufferedImage> filmsImage = new ArrayList<>();
    MoviesIntroPanelListener listener;
    /**
     * This is the callback interface of MoviesIntroPanel.
     * @author group 99
```

```
     *
     */
    public interface MoviesIntroPanelListener{
        public void onHomeClicked();
        public void onImageClicked(String filmName);
    }
    /**
     * Set MoviesIntroPanel's listener.
     * @param listener A listener of MoviesIntroPanel.
     */
    public void setMoviesIntroPanelListener(MoviesIntroPanelListener
listener){
        this.listener = listener;
    }

    /**
     * This is the constructor of MoviesIntroPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public MoviesIntroPanel() throws ParserConfigurationException,
SAXException, IOException {
        initPanel();
    }
    /**
     * Initialize the data of MoviesIntroPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    private void initPanel() throws ParserConfigurationException,
SAXException, IOException {

        int count = 0;
        List<Film> films = FilmDomParser.getFilms();

        setLayout(null);

        for(Film film : films){
            String filmName = film.getFilmName();
            String filmType = film.getFilmType();
            String filmScore = film.getFilmScore();
            BufferedImage image = ImageIO.read(new
```

```
FileInputStream("././../resource/" + filmName + ".jpg"));

            BufferedImage homeImage = ImageIO.read(new
FileInputStream("././../resource/home.png"));
            JButton homeBack = new JButton();
            homeBack.setBounds(10, 500, 50, 50);
            homeBack.setOpaque(false);
            homeBack.setUI(new BasicButtonUI() {

                @Override
                public void paint(Graphics g, JComponent c) {
                    Graphics2D graphics2d = (Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                    graphics2d.drawImage(homeImage, 0, 0, 50, 50, null);
                }

            });
            homeBack.addMouseListener(new MouseAdapter() {

                @Override
                public void mouseClicked(MouseEvent e) {
                    listener.onHomeClicked();
                }

            });
            add(homeBack);

            JButton button = new JButton();
            button.setUI(new BasicButtonUI(){
                @Override
                public void paint(Graphics g, JComponent c) {
                    Graphics2D graphics2d = (Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                    graphics2d.drawImage(image, 0, 0, BTN_IMAGE_SIZE_WIDTH,
BTN_IMAGE_SIZE_HEIGHT, null);
                }
            });
            button.setBounds(31 + count*185, 150, BTN_IMAGE_SIZE_WIDTH,
BTN_IMAGE_SIZE_HEIGHT);
            button.addMouseListener(new MouseAdapter(){
```

```
                @Override
                public void mouseClicked(MouseEvent e) {
                    listener.onImageClicked(filmName);
                }
            });
            add(button);

            JLabel labelFilmName = new JLabel();
            labelFilmName.setFont(new Font("Arial", Font.BOLD, 12));
            labelFilmName.setText(filmName);
            labelFilmName.setHorizontalAlignment(JLabel.CENTER);
            labelFilmName.setOpaque(false);
            labelFilmName.setBounds(26 + count*185, 400, 160, 12);
            add(labelFilmName);

            JLabel labelFilmType = new JLabel();
            labelFilmType.setFont(new Font("Arial", Font.BOLD, 13));
            labelFilmType.setText("Type: " + filmType);
            labelFilmType.setHorizontalAlignment(JLabel.CENTER);
            labelFilmType.setOpaque(false);
            labelFilmType.setBounds(26 + count*185, 430, 160, 12);
            add(labelFilmType);

            JLabel labelFilmScore = new JLabel();
            labelFilmScore.setFont(new Font("Arial", Font.BOLD, 15));
            labelFilmScore.setText("Score: " + filmScore);
            labelFilmScore.setForeground(new Color(255, 130, 71));
            labelFilmScore.setHorizontalAlignment(JLabel.CENTER);
            labelFilmScore.setOpaque(false);
            labelFilmScore.setBounds(26 + count*185, 460, 160, 12);
            add(labelFilmScore);

            count++;
        }
    }
    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
    protected void paintComponent(Graphics g) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
```

```
        graphics2d.setFont(new Font("Copperplate", Font.PLAIN, 35));
        graphics2d.drawString("Now Showing", 350, 80);

    }

}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JPanel;
import javax.swing.JTextField;

import com.group99.javabean.Ticket;

/**
 * This is the boundary class of panel of payment dialog.
 * @author group 99
 *
 */
public class PaymentDialogPanel extends JPanel {

    private BufferedImage movImg;
    private float payAmount = 0;
    private String whichHandle;

    public static final String ENTER_HANDLE = "enter interface";
    public static final String ERROR_HANDLE = "error interface";
    public static final String NSF_HANDLE = "not sufficient funds
interface";
    public static final String ASK_HANDLE = "ask interface";
    public static final String SUCCESS_HANDLE = "successful payment
interface";
```

```
    private PaymentDialogPanelListener listener;
    /**
     * This is the callback interface of PaymentDialogPanel.
     * @author group 99
     *
     */
    public interface PaymentDialogPanelListener{
        public void onConfirmClicked(String accountNum, String
accountPassword, float payAmount);
        public void onCancelClicked();
    }
    /**
     * Set PaymentDialogPanel's listener.
     * @param listener A listener of PaymentDialogPanel.
     */
    public void setPaymentDialogPanelListener(PaymentDialogPanelListener
listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of PaymentDialogPanel.
     * @param movImg The name of selecting movie.
     * @param payAmount The pay amount.
     * @param whichHandle  Which handle you want to do.
     */
    public PaymentDialogPanel(BufferedImage movImg, float payAmount, String
whichHandle) {

        this.movImg = movImg;
        this.payAmount = payAmount;
        this.whichHandle = whichHandle;
        initData();
    }
    /**
     * Initialize the data of PaymentDialogPanel.
     */
    private void initData() {

        setLayout(null);

        JButton btnCancel = new JButton("Back");
        btnCancel.setFont(new Font("Arial", Font.BOLD, 15));
        btnCancel.setForeground(Color.WHITE);
```

```
        btnCancel.setOpaque(true);
        btnCancel.setBounds(35, 286, 108, 34);
        btnCancel.setUI(new ClientNormalButtonUI());
        btnCancel.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onCancelClicked();
            }

        });
        add(btnCancel);

        JButton btnConfirm = new JButton("Confirm");
        btnConfirm.setFont(new Font("Arial", Font.BOLD, 15));
        btnConfirm.setForeground(Color.WHITE);
        btnConfirm.setOpaque(true);
        btnConfirm.setBounds(35, 334, 108, 34);
        btnConfirm.setUI(new ClientNormalButtonUI());

        if("enter interface".equals(whichHandle)){

            JTextField idTextField = new JTextField(10);;
            JTextField passwordTextField = new JTextField(10);

            idTextField.setBounds(250, 150, 300, 40);
            idTextField.setText("Your account id...");
            idTextField.setOpaque(false);
            add(idTextField);

            passwordTextField.setBounds(250, 250, 300, 40);
            passwordTextField.setText("Your account password...");
            passwordTextField.setOpaque(false);
            add(passwordTextField);

            btnConfirm.addMouseListener(new MouseAdapter() {

                @Override
                public void mouseClicked(MouseEvent e) {
                    listener.onConfirmClicked(idTextField.getText(),
passwordTextField.getText(), payAmount);
                }

            });
```

```
            add(btnConfirm);
        }else if("ask interface".equals(whichHandle)){
            btnConfirm.addMouseListener(new MouseAdapter() {

                @Override
                public void mouseClicked(MouseEvent e) {
                    listener.onConfirmClicked(null, null, payAmount);
                }

            });
            add(btnConfirm);
        }

    }
    /**
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        graphics2d.drawImage(movImg, 21, 37, 140, 207, null);

        if("enter interface".equals(whichHandle)){
            graphics2d.setFont(new Font("Arial", Font.BOLD, 20));
            graphics2d.drawString("Bank System", 330, 80);
        }else if("error interface".equals(whichHandle)){
            graphics2d.setFont(new Font("Arial", Font.BOLD, 21));
            graphics2d.drawString("Your account name or password is error!",
200, 220);
        }else if("not sufficient funds interface".equals(whichHandle)){
            graphics2d.setFont(new Font("Arial", Font.BOLD, 23));
            graphics2d.drawString("Your balance is insufficient!", 245,
220);
        }else if("ask interface".equals(whichHandle)){
            graphics2d.setFont(new Font("Arial", Font.BOLD, 23));
            graphics2d.drawString("Confirm the payment ($ " +
String.format("%.2f",payAmount) + ")?", 230, 220);
        }else if("successful payment interface".equals(whichHandle)){
            graphics2d.setFont(new Font("Arial", Font.BOLD, 23));
            graphics2d.drawString("Payment Status: pay success", 245, 220);
        }
```

```
            super.paint(graphics2d);
        }



}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.dom.ScreenSeatDomParser;
import com.group99.dom.StudentDomParser;
import com.group99.dom.TicketDomParser;
import com.group99.gui.CartDialog.CartDialogListener;
import com.group99.gui.SeatTypeSelectDialog.SeatSelectDialogListener;
import
com.group99.gui.StudentInfoCheckDialog.StudentInfoCheckDialogListener;
import
com.group99.gui.StudentInfoCheckHintDialog.StudentInfoCheckHintDialogListen
er;
```

```
import com.group99.javabean.Film;
import com.group99.javabean.Screen;
import com.group99.javabean.Student;
import com.group99.javabean.Ticket;
/**
 * This is the boundary class of panel of selecting seat from screen 1.
 * @author group 99.
 *
 */
public class Screen1SeatSelectPanel extends JPanel {

    private String filmName;
    private String timeStr;
    private float ticketPrice;
    private BufferedImage movImg;
    private BufferedImage selectingImg;
    private BufferedImage selectedImg;
    private JFrame frame;

    private List<Ticket> tickets = null;

    private Screen1SeatSelectPanelListener listener;
    /**
     * This is the callback interface of Screen1SeatSelectPanel.
     * @author group 99
     *
     */
    public interface Screen1SeatSelectPanelListener{
        public void onBtnBackClicked();
        public void repaintPanel(String movName, String screenName, String
timeStr);
    }
    /**
     * Set Screen1SeatSelectPanel's listener.
     * @param listener A listener of Screen1SeatSelectPanel.
     */
    public void
setScreen1SeatSelectPanelListener(Screen1SeatSelectPanelListener listener){
        this.listener = listener;
    }

    /**
     * This is the constructor of Screen1SeatSelectPanel.
     * @param filmName The name of selecting movie.
```

```
     * @param timeStr The time you selecting.
     * @param frame The current frame.
     * @throws IOException
     * @throws ParserConfigurationException
     * @throws SAXException
     */
    public Screen1SeatSelectPanel(String filmName, String timeStr, JFrame
frame) throws IOException, ParserConfigurationException, SAXException {
        this.filmName = filmName;
        this.timeStr = timeStr;
        this.frame = frame;
        initData();
    }
    /**
     * Make a object of Ticket.
     * @param ticketType The type of Ticket.
     * @param row The row of seat.
     * @param column The column of seat.
     * @param studentId The student id of Ticket.
     */
    private void makeTicket(String ticketType, String row, int column,
String studentId){
        Ticket ticket = new Ticket();
        ticket.setFilmName(filmName);
        ticket.setScreenName("screen 1");

        String ticketId = "";

        for(int i = 0; i < 8; i++){
            int randomNum = (int)(Math.random()*4) + 1;
            ticketId = ticketId + randomNum;
        }

        try {
            int count = 0;
            List<Ticket> allTickets = TicketDomParser.getTickets();
            for(Ticket currentTicket : allTickets){
                if(ticketId.equals(currentTicket.getTicketId())){
                    makeTicket(ticketType, row, column, studentId);
                    return;
                }
                if(count == allTickets.size() - 1){
                    ticket.setTicketId(ticketId);
                }
```

```
                count ++;
            }


        } catch (ParserConfigurationException | SAXException | IOException
e1) {
            e1.printStackTrace();
        }

        ticket.setSeatLocation(row + column);

        ticket.setTiketType(ticketType);

        List<Film> films = null;
        try {
            films = FilmDomParser.getFilms();
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        for(Film film : films){
            if(filmName.equals(film.getFilmName()))
                ticketPrice = film.getFilmPrice();
        }

        if("child".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.5f);
        }else if("senior".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.2f);
        }else if("student".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.15f);
        }else{
            ticket.setTicketPrice(ticketPrice);
        }
        ticket.setTimeStr(timeStr);
        ticket.setStudentId(studentId);
        tickets.add(ticket);
    }
    /**
     * To check what type of the ticket and the student information.
     * @param ticketType The type of ticket.
     * @param button The current button.
     * @param row The row of current seat.
```

```
     * @param column The column of current seat.
     * @param seatSelectDialog The current dialog of selecting seat.
     */
    private void typeInfoCheck(String ticketType, JButton button,String
row, int column, SeatTypeSelectDialog seatSelectDialog){
        if("student".equals(ticketType)){
            StudentInfoCheckDialog studentInfoCheckDialog = new
StudentInfoCheckDialog(frame, button.getX(), button.getY());
            studentInfoCheckDialog.setAlwaysOnTop(true);
            studentInfoCheckDialog.setVisible(true);
            studentInfoCheckDialog.setStudentInfoCheckDialogListener(new
StudentInfoCheckDialogListener() {

                @Override
                public void onOkClicked(String studentName, String
studentNum) {

                    List<Student> students = null;
                    try {
                        students = StudentDomParser.getStudents();
                    } catch (ParserConfigurationException | SAXException |
IOException e) {
                        e.printStackTrace();
                    }
                    int i = 0;
                    for(Student student : students){
                        if(studentName.equals(student.getName())){
                            if(studentNum.equals(student.getStudentNum())){
                                studentInfoCheckDialog.dispose();
                                button.setEnabled(false);
                                try {
                                    button.setUI(new
SeatSelectedButtonUI(column + ""){
                                        @Override
                                        public void paint(Graphics g,
JComponent c) {
                                            Graphics2D graphics2d =
(Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                                            graphics2d.drawImage(selectedImg,
0, 0, 60, 53, null);
                                        }
```

```
                                    });
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                                makeTicket(ticketType, row, column,
studentNum);
                                seatSelectDialog.dispose();
                                return;
                            }
                        }
                        if((students.size() - 1) == i){
                            StudentInfoCheckHintDialog
studentInfoCheckHintDialog = new StudentInfoCheckHintDialog(frame,
button.getX(),button.getY());
                            studentInfoCheckHintDialog.setAlwaysOnTop(true);
                            studentInfoCheckHintDialog.setVisible(true);

    studentInfoCheckHintDialog.setStudentInfoCheckHintDialogListener(new
StudentInfoCheckHintDialogListener() {

                                @Override
                                public void onCancelClicked() {
                                    studentInfoCheckHintDialog.dispose();
                                }
                            });
                            return;
                        }
                        i++;
                    }

                }

                @Override
                public void onCancelClicked() {
                    studentInfoCheckDialog.dispose();
                }
            });
        }else{
            button.setEnabled(false);
            try {
                button.setUI(new SeatSelectedButtonUI(column + ""){
                    @Override
                    public void paint(Graphics g, JComponent c) {
                        Graphics2D graphics2d = (Graphics2D) g;
```

```
    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                        graphics2d.drawImage(selectedImg, 0, 0, 60, 53,
null);
                    }
                });
            } catch (IOException e) {
                e.printStackTrace();
            }
            makeTicket(ticketType, row, column, " ");
            seatSelectDialog.dispose();
        }
    }
    /**
     * Initialize the data of Screen1SeatSelectPanel.
     * @throws IOException
     * @throws ParserConfigurationException
     * @throws SAXException
     */
    private void initData() throws IOException,
ParserConfigurationException, SAXException {

        tickets = new ArrayList<Ticket>();

        movImg = ImageIO.read(new FileInputStream("./../../resource/" +
filmName + ".jpg"));
        selectingImg = ImageIO.read(new FileInputStream("./../../resource/"
+ "0" + ".png"));
        selectedImg = ImageIO.read(new FileInputStream("./../../resource/"
+ "s0" + ".png"));

        setLayout(null);

        List<Screen> screens = ScreenSeatDomParser.getScreen("screen1At" +
timeStr.split(":")[0] + "_" + timeStr.split(":")[1] + ".xml");
        for(Screen screen : screens){
            String row = screen.getSeatId().substring(0, 1);
            int column = Integer.parseInt(screen.getSeatId().substring(1,
2));
            if("D".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
```

```
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, "D",
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                    button.setEnabled(false);
                }
                if(column <= 4){
                    button.setBounds(920 - 61*column, 120, 60, 53);
                }else if(column > 4){
                    button.setBounds(880 - 61*column, 120, 60, 53);
                }

                add(button);
            }else if("C".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());
```

```
                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, "C",
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 4){
                    button.setBounds(920 - 61*column, 210, 60, 53);
                }else if(column > 4){
                    button.setBounds(880 - 61*column, 210, 60, 53);
                }
                add(button);
            }else if("B".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());
                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }
```

```
                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, "B",
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 4){
                    button.setBounds(920 - 61*column, 300, 60, 53);
                }else if(column > 4){
                    button.setBounds(880 - 61*column, 300, 60, 53);
                }
                add(button);
            }else if("A".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(),button.getY());
                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, "A",
column, seatSelectDialog);
                                }
                            });
```

```
                }
            });
        }else if("false".equals(screen.getSeatIsEmpty())){
            button.setUI(new SeatSelectedButtonUI(column + ""));
        }
        if(column <= 4){
            button.setBounds(920 - 61*column, 390, 60, 53);
        }else if(column > 4){
            button.setBounds(880 - 61*column, 390, 60, 53);
        }
        add(button);
    }
}

JButton btnBack = new JButton("Back");
btnBack.setFont(new Font("Arial", Font.BOLD, 20));
btnBack.setForeground(Color.WHITE);
btnBack.setOpaque(true);
btnBack.setBounds(50, 365, 154, 48);
btnBack.setUI(new ClientNormalButtonUI());
btnBack.addMouseListener(new MouseAdapter() {

    @Override
    public void mouseClicked(MouseEvent e) {
        listener.onBtnBackClicked();
    }

});
add(btnBack);

JButton btnCart = new JButton("Cart");
btnCart.setFont(new Font("Arial", Font.BOLD, 20));
btnCart.setForeground(Color.WHITE);
btnCart.setOpaque(true);
btnCart.setBounds(50, 435, 154, 48);
btnCart.setUI(new ClientNormalButtonUI());
btnCart.addMouseListener(new MouseAdapter() {

    @Override
    public void mouseClicked(MouseEvent e) {

        if(tickets.isEmpty()){
            return;
        }
```

```
                CartDialog cartDialog = new CartDialog(tickets, movImg,
frame.getX(), frame.getY());
                cartDialog.setVisible(true);
                cartDialog.setAlwaysOnTop(true);
                cartDialog.setCartDialogListener(new CartDialogListener() {

                    @Override
                    public void onCancelClicked() {
                        // TODO Auto-generated method stub
                        tickets.clear();
                        cartDialog.dispose();
                        listener.repaintPanel(filmName, "1", timeStr);
                    }
                });
            }
        });
        add(btnCart);

    }
    /**
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.drawImage(movImg, 30, 10, 200, 296, null);
        graphics2d.drawString("Screen 1 at " + timeStr, 70, 340);
        graphics2d.drawLine(300, 5, 300, 530);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 18));
        graphics2d.drawString("SEAT SELECT", 570, 60);

        graphics2d.setFont(new Font("Arial", Font.PLAIN, 18));
        graphics2d.drawString("D", 350, 160);
        graphics2d.drawString("C", 350, 250);
        graphics2d.drawString("B", 350, 340);
        graphics2d.drawString("A", 350, 430);

        graphics2d.drawLine(550, 490, 763, 490);
        graphics2d.setFont(new Font("Arial", Font.ITALIC, 15));
        graphics2d.drawString("screen", 630, 508);
```

```
        graphics2d.drawLine(550, 515, 763, 515);

        graphics2d.drawImage(selectingImg, 805, 480, 30, 27, null);
        graphics2d.drawString("selecting seat", 840, 500);
        graphics2d.drawImage(selectedImg, 805, 520, 30, 27, null);
        graphics2d.drawString("selected seat", 840, 540);

        super.paint(graphics2d);

    }


}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.dom.ScreenSeatDomParser;
import com.group99.dom.StudentDomParser;
import com.group99.gui.CartDialog.CartDialogListener;
import com.group99.gui.ScreenSelectPanel.ScreenSelectPanelListener;
import com.group99.gui.SeatTypeSelectDialog.SeatSelectDialogListener;
```

```
import
com.group99.gui.StudentInfoCheckDialog.StudentInfoCheckDialogListener;
import
com.group99.gui.StudentInfoCheckHintDialog.StudentInfoCheckHintDialogListen
er;
import com.group99.javabean.Film;
import com.group99.javabean.Screen;
import com.group99.javabean.Student;
import com.group99.javabean.Ticket;
/**
 * This is the boundary class of panel of selecting seat from screen 2.
 * @author group 99.
 *
 */
public class Screen2SeatSelectPanel extends JPanel {

    private String filmName;
    private String timeStr;
    private float ticketPrice;
    private BufferedImage movImg;
    private BufferedImage selectingImg;
    private BufferedImage selectedImg;
    private JFrame frame;

    private List<Ticket> tickets = null;
    private Screen2SeatSelectPanelListener listener;
    /**
     * This is the callback interface of Screen2SeatSelectPanel.
     * @author group 99
     *
     */
    public interface Screen2SeatSelectPanelListener{
        public void onBtnBackClicked();
        public void repaintPanel(String movName, String screenName, String
timeStr);
    }
    /**
     * Set Screen2SeatSelectPanel's listener.
     * @param listener A listener of Screen2SeatSelectPanel.
     */
    public void
setScreen2SeatSelectPanelListener(Screen2SeatSelectPanelListener listener){
        this.listener = listener;
    }
```

```
    /**
     * This is the constructor of Screen2SeatSelectPanel.
     * @param filmName The name of selecting movie.
     * @param timeStr The time you selecting.
     * @param frame The current frame.
     * @throws IOException
     * @throws ParserConfigurationException
     * @throws SAXException
     */
    public Screen2SeatSelectPanel(String filmName, String timeStr,JFrame
frame) throws IOException, ParserConfigurationException, SAXException {
        this.filmName = filmName;
        this.timeStr = timeStr;
        this.frame = frame;
        initData();
    }
    /**
     * Make a object of Ticket.
     * @param ticketType The type of Ticket.
     * @param row The row of seat.
     * @param column The column of seat.
     * @param studentId The student id of Ticket.
     */
    private void makeTicket(String ticketType, String row, int column,
String studentId){
        Ticket ticket = new Ticket();
        ticket.setFilmName(filmName);
        ticket.setScreenName("screen 2");

        String ticketId = "";

        for(int i = 0; i < 8; i++){
            int randomNum = (int)(Math.random()*4) + 1;
            ticketId = ticketId + randomNum;
        }

        ticket.setTicketId(ticketId);

        ticket.setSeatLocation(row + column);

        ticket.setTiketType(ticketType);

        List<Film> films = null;
```

```
        try {
            films = FilmDomParser.getFilms();
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        for(Film film : films){
            if(filmName.equals(film.getFilmName()))
                ticketPrice = film.getFilmPrice();
        }

        if("child".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.5f);
        }else if("senior".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.2f);
        }else if("student".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.15f);
        }else{
            ticket.setTicketPrice(ticketPrice);
        }
        ticket.setTimeStr(timeStr);
        ticket.setStudentId(studentId);
        tickets.add(ticket);
    }
    /**
     * To check what type of the ticket and the student information.
     * @param ticketType The type of ticket.
     * @param button The current button.
     * @param row The row of current seat.
     * @param column The column of current seat.
     * @param seatSelectDialog The current dialog of selecting seat.
     */
    private void typeInfoCheck(String ticketType, JButton button,String
row, int column, SeatTypeSelectDialog seatSelectDialog){
        if("student".equals(ticketType)){
            StudentInfoCheckDialog studentInfoCheckDialog = new
StudentInfoCheckDialog(frame, button.getX(), button.getY());
            studentInfoCheckDialog.setAlwaysOnTop(true);
            studentInfoCheckDialog.setVisible(true);
            studentInfoCheckDialog.setStudentInfoCheckDialogListener(new
StudentInfoCheckDialogListener() {

                @Override
```

```
                public void onOkClicked(String studentName, String
studentNum) {

                    List<Student> students = null;
                    try {
                        students = StudentDomParser.getStudents();
                    } catch (ParserConfigurationException | SAXException |
IOException e) {
                        e.printStackTrace();
                    }
                    int i = 0;
                    for(Student student : students){
                        if(studentName.equals(student.getName())){
                            if(studentNum.equals(student.getStudentNum())){
                                studentInfoCheckDialog.dispose();
                                button.setEnabled(false);
                                try {
                                    button.setUI(new
SeatSelectedButtonUI(column + ""){
                                        @Override
                                        public void paint(Graphics g,
JComponent c) {
                                            Graphics2D graphics2d =
(Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                                            graphics2d.drawImage(selectedImg,
0, 0, 60, 53, null);
                                        }
                                    });
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                                makeTicket(ticketType, row, column,
studentNum);
                                seatSelectDialog.dispose();
                                return;
                            }
                        }
                        if((students.size() - 1) == i){
                            StudentInfoCheckHintDialog
studentInfoCheckHintDialog = new StudentInfoCheckHintDialog(frame,
button.getX(),button.getY());
```

```
                            studentInfoCheckHintDialog.setAlwaysOnTop(true);
                            studentInfoCheckHintDialog.setVisible(true);

    studentInfoCheckHintDialog.setStudentInfoCheckHintDialogListener(new
StudentInfoCheckHintDialogListener() {

                                @Override
                                public void onCancelClicked() {
                                    studentInfoCheckHintDialog.dispose();
                                }
                            });
                            return;
                        }
                        i++;
                    }

                }

                @Override
                public void onCancelClicked() {
                    studentInfoCheckDialog.dispose();
                }
            });
        }else{
            button.setEnabled(false);
            try {
                button.setUI(new SeatSelectedButtonUI(column + ""){
                    @Override
                    public void paint(Graphics g, JComponent c) {
                        Graphics2D graphics2d = (Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                        graphics2d.drawImage(selectedImg, 0, 0, 60, 53,
null);
                    }
                });
            } catch (IOException e) {
                e.printStackTrace();
            }
            makeTicket(ticketType, row, column, " ");
            seatSelectDialog.dispose();
        }
    }
```

```
    /**
     * Initialize the data of Screen2SeatSelectPanel.
     * @throws IOException
     * @throws ParserConfigurationException
     * @throws SAXException
     */
    private void initData() throws IOException,
ParserConfigurationException, SAXException {

        tickets = new ArrayList<Ticket>();
        movImg = ImageIO.read(new FileInputStream("./../../../resource/" +
filmName + ".jpg"));
        selectingImg = ImageIO.read(new FileInputStream("./../../../resource/"
+ "0" + ".png"));
        selectedImg = ImageIO.read(new FileInputStream("./../../../resource/"
+ "s0" + ".png"));

        setLayout(null);

        List<Screen> screens = ScreenSeatDomParser.getScreen("screen2At" +
timeStr.split(":")[0] + "_" + timeStr.split(":")[1] + ".xml");
        for(Screen screen : screens){
            String row = screen.getSeatId().substring(0, 1);
            int column = Integer.parseInt(screen.getSeatId().substring(1,
2));
            if("D".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }
```

```
                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });

                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                    button.setEnabled(false);
                }
                if(column <= 4){
                    button.setBounds(920 - 61*column, 120, 60, 53);
                }else if(column > 4){
                    button.setBounds(880 - 61*column, 120, 60, 53);
                }

                add(button);
            }else if("C".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
```

```
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 3){
                    button.setBounds(920 - 61*(column + 1), 210, 60, 53);
                }else if(column > 3){
                    button.setBounds(880 - 61*(column + 1), 210, 60, 53);
                }
                add(button);
            }else if("B".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
```

```
                }
                if(column <= 3){
                    button.setBounds(920 - 61*(column + 1), 300, 60, 53);
                }else if(column > 3){
                    button.setBounds(880 - 61*(column + 1), 300, 60, 53);
                }
                add(button);
            }else if("A".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 3){
                    button.setBounds(920 - 61*(column + 1), 390, 60, 53);
                }else if(column > 3){
                    button.setBounds(880 - 61*(column + 1), 390, 60, 53);
                }
                add(button);
```

```
            }
        }

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.setOpaque(true);
        btnBack.setBounds(50, 365, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        btnBack.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBtnBackClicked();
            }

        });
        add(btnBack);

        JButton btnCart = new JButton("Cart");
        btnCart.setFont(new Font("Arial", Font.BOLD, 20));
        btnCart.setForeground(Color.WHITE);
        btnCart.setOpaque(true);
        btnCart.setBounds(50, 435, 154, 48);
        btnCart.setUI(new ClientNormalButtonUI());
        btnCart.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {

                if(tickets.isEmpty()){
                    return;
                }

                CartDialog cartDialog = new CartDialog(tickets, movImg,
frame.getX(), frame.getY());
                cartDialog.setVisible(true);
                cartDialog.setAlwaysOnTop(true);
                cartDialog.setCartDialogListener(new CartDialogListener() {

                    @Override
                    public void onCancelClicked() {
                        // TODO Auto-generated method stub
                        tickets.clear();
```

```
                        cartDialog.dispose();
                        listener.repaintPanel(filmName, "2", timeStr);
                    }
                });
            }

        });
        add(btnCart);

    }
    /**
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.drawImage(movImg, 30, 10, 200, 296, null);
        graphics2d.drawString("Screen 2 at " + timeStr, 70, 340);
        graphics2d.drawLine(300, 5, 300, 530);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 18));
        graphics2d.drawString("SEAT SELECT", 570, 60);

        graphics2d.setFont(new Font("Arial", Font.PLAIN, 18));
        graphics2d.drawString("D", 350, 160);
        graphics2d.drawString("C", 350, 250);
        graphics2d.drawString("B", 350, 340);
        graphics2d.drawString("A", 350, 430);

        graphics2d.drawLine(550, 490, 763, 490);
        graphics2d.setFont(new Font("Arial", Font.ITALIC, 15));
        graphics2d.drawString("screen", 630, 508);
        graphics2d.drawLine(550, 515, 763, 515);

        graphics2d.drawImage(selectingImg, 805, 480, 30, 27, null);
        graphics2d.drawString("selecting seat", 840, 500);
        graphics2d.drawImage(selectedImg, 805, 520, 30, 27, null);
        graphics2d.drawString("selected seat", 840, 540);

        super.paint(graphics2d);

    }
```

```
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.dom.ScreenSeatDomParser;
import com.group99.dom.StudentDomParser;
import com.group99.gui.CartDialog.CartDialogListener;
import com.group99.gui.ScreenSelectPanel.ScreenSelectPanelListener;
import com.group99.gui.SeatTypeSelectDialog.SeatSelectDialogListener;
import
com.group99.gui.StudentInfoCheckDialog.StudentInfoCheckDialogListener;
import
com.group99.gui.StudentInfoCheckHintDialog.StudentInfoCheckHintDialogListen
er;
import com.group99.javabean.Film;
import com.group99.javabean.Screen;
import com.group99.javabean.Student;
import com.group99.javabean.Ticket;
/**
```

```
 * This is the boundary class of panel of selecting seat from screen 3.
 * @author group 99.
 *
 */
public class Screen3SeatSelectPanel extends JPanel {

    private String filmName;
    private String timeStr;
    private float ticketPrice;
    private BufferedImage movImg;
    private BufferedImage selectingImg;
    private BufferedImage selectedImg;
    private JFrame frame;

    private List<Ticket> tickets = null;
    private Screen3SeatSelectPanelListener listener;
    /**
     * This is the callback interface of Screen3SeatSelectPanel.
     * @author group 99
     *
     */
    public interface Screen3SeatSelectPanelListener{
        public void onBtnBackClicked();
        public void repaintPanel(String movName, String screenName, String
timeStr);
    }
    /**
     * Set Screen3SeatSelectPanel's listener.
     * @param listener A listener of Screen3SeatSelectPanel.
     */
    public void
setScreen3SeatSelectPanelListener(Screen3SeatSelectPanelListener listener){
        this.listener = listener;
    }

    /**
     * This is the constructor of Screen3SeatSelectPanel.
     * @param filmName The name of selecting movie.
     * @param timeStr The time you selecting.
     * @param frame The current frame.
     * @throws IOException
     * @throws ParserConfigurationException
     * @throws SAXException
     */
```

```
    public Screen3SeatSelectPanel(String filmName, String timeStr, JFrame
frame) throws IOException, ParserConfigurationException, SAXException {
        this.filmName = filmName;
        this.timeStr = timeStr;
        this.frame = frame;
        initData();
    }
    /**
     * Make a object of Ticket.
     * @param ticketType The type of Ticket.
     * @param row The row of seat.
     * @param column The column of seat.
     * @param studentId The student id of Ticket.
     */
    private void makeTicket(String ticketType, String row, int column,
String studentId){
        Ticket ticket = new Ticket();
        ticket.setFilmName(filmName);
        ticket.setScreenName("screen 3");

        String ticketId = "";

        for(int i = 0; i < 8; i++){
            int randomNum = (int)(Math.random()*4) + 1;
            ticketId = ticketId + randomNum;
        }

        ticket.setTicketId(ticketId);

        ticket.setSeatLocation(row + column);

        ticket.setTiketType(ticketType);

        List<Film> films = null;
        try {
            films = FilmDomParser.getFilms();
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        for(Film film : films){
            if(filmName.equals(film.getFilmName()))
                ticketPrice = film.getFilmPrice();
```

```
        }

        if("child".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.5f);
        }else if("senior".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.2f);
        }else if("student".equals(ticketType)){
            ticket.setTicketPrice(ticketPrice*0.15f);
        }else{
            ticket.setTicketPrice(ticketPrice);
        }
        ticket.setTimeStr(timeStr);
        ticket.setStudentId(studentId);
        tickets.add(ticket);
    }
    /**
     * To check what type of the ticket and the student information.
     * @param ticketType The type of ticket.
     * @param button The current button.
     * @param row The row of current seat.
     * @param column The column of current seat.
     * @param seatSelectDialog The current dialog of selecting seat.
     */
    private void typeInfoCheck(String ticketType, JButton button,String
row, int column, SeatTypeSelectDialog seatSelectDialog){
        if("student".equals(ticketType)){
            StudentInfoCheckDialog studentInfoCheckDialog = new
StudentInfoCheckDialog(frame, button.getX(), button.getY());
            studentInfoCheckDialog.setAlwaysOnTop(true);
            studentInfoCheckDialog.setVisible(true);
            studentInfoCheckDialog.setStudentInfoCheckDialogListener(new
StudentInfoCheckDialogListener() {

                @Override
                public void onOkClicked(String studentName, String
studentNum) {

                    List<Student> students = null;
                    try {
                        students = StudentDomParser.getStudents();
                    } catch (ParserConfigurationException | SAXException |
IOException e) {
                        e.printStackTrace();
                    }
```

```
                    int i = 0;
                    for(Student student : students){
                        if(studentName.equals(student.getName())){
                            if(studentNum.equals(student.getStudentNum())){
                                studentInfoCheckDialog.dispose();
                                button.setEnabled(false);
                                try {
                                    button.setUI(new
SeatSelectedButtonUI(column + ""){
                                        @Override
                                        public void paint(Graphics g,
JComponent c) {
                                            Graphics2D graphics2d =
(Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                                            graphics2d.drawImage(selectedImg,
0, 0, 60, 53, null);
                                        }
                                    });
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                                makeTicket(ticketType, row, column,
studentNum);
                                seatSelectDialog.dispose();
                                return;
                            }
                        }
                        if((students.size() - 1) == i){
                            StudentInfoCheckHintDialog
studentInfoCheckHintDialog = new StudentInfoCheckHintDialog(frame,
button.getX(),button.getY());
                            studentInfoCheckHintDialog.setAlwaysOnTop(true);
                            studentInfoCheckHintDialog.setVisible(true);

    studentInfoCheckHintDialog.setStudentInfoCheckHintDialogListener(new
StudentInfoCheckHintDialogListener() {

                                @Override
                                public void onCancelClicked() {
                                    studentInfoCheckHintDialog.dispose();
                                }
```

```
                        });
                        return;
                    }
                    i++;
                }

            }

            @Override
            public void onCancelClicked() {
                studentInfoCheckDialog.dispose();
            }
        });
    }else{
        button.setEnabled(false);
        try {
            button.setUI(new SeatSelectedButtonUI(column + ""){
                @Override
                public void paint(Graphics g, JComponent c) {
                    Graphics2D graphics2d = (Graphics2D) g;

    graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                    graphics2d.drawImage(selectedImg, 0, 0, 60, 53,
null);
                }
            });
        } catch (IOException e) {
            e.printStackTrace();
        }
        makeTicket(ticketType, row, column, " ");
        seatSelectDialog.dispose();
    }
}
/**
 * Initialize the data of Screen3SeatSelectPanel.
 * @throws IOException
 * @throws ParserConfigurationException
 * @throws SAXException
 */
private void initData() throws IOException,
ParserConfigurationException, SAXException {

    tickets = new ArrayList<Ticket>();
```

```
        movImg = ImageIO.read(new FileInputStream("././../resource/" +
filmName + ".jpg"));
        selectingImg = ImageIO.read(new FileInputStream("././../resource/"
+ "0" + ".png"));
        selectedImg = ImageIO.read(new FileInputStream("././../resource/"
+ "s0" + ".png"));

        setLayout(null);

        List<Screen> screens = ScreenSeatDomParser.getScreen("screen3At" +
timeStr.split(":")[0] + "_" + timeStr.split(":")[1] + ".xml");
        for(Screen screen : screens){
            String row = screen.getSeatId().substring(0, 1);
            int column = Integer.parseInt(screen.getSeatId().substring(1,
2));

            if("E".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
```

```
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                    button.setEnabled(false);
                }
                button.setBounds(900 - 61*column, 90, 60, 53);
                add(button);
            }else if("D".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                    button.setEnabled(false);
                }
                if(column <= 2 && column >= 1){
                    button.setBounds(900 - 61*column, 165, 60, 53);
                }else if(column >= 3 && column <= 4){
                    button.setBounds(900 - 61*(column + 1), 165, 60, 53);
                }else if(column >= 5 && column <= 6){
```

```
                    button.setBounds(900 - 61*(column + 2), 165, 60, 53);
                }
                add(button);
            }else if("C".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 2 && column >= 1){
                    button.setBounds(900 - 61*column, 240, 60, 53);
                }else if(column >= 3 && column <= 4){
                    button.setBounds(900 - 61*(column + 1), 240, 60, 53);
                }else if(column >= 5 && column <= 6){
                    button.setBounds(900 - 61*(column + 2), 240, 60, 53);
                }
                add(button);
            }else if("B".equals(row)){
                JButton button = new JButton();
```

```
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 2 && column >= 1){
                    button.setBounds(900 - 61*column, 315, 60, 53);
                }else if(column >= 3 && column <= 4){
                    button.setBounds(900 - 61*(column + 1), 315, 60, 53);
                }else if(column >= 5 && column <= 6){
                    button.setBounds(900 - 61*(column + 2), 315, 60, 53);
                }
                add(button);
            }else if("A".equals(row)){
                JButton button = new JButton();
                if("true".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectingButtonUI(column + ""));
                    button.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent e) {
```

```
                            SeatTypeSelectDialog seatSelectDialog = new
SeatTypeSelectDialog(frame,button.getX(), button.getY());

                            seatSelectDialog.setAlwaysOnTop(true);
                            seatSelectDialog.setVisible(true);
                            seatSelectDialog.setSeatSelectDialogListener(new
SeatSelectDialogListener() {
                                @Override
                                public void onCancelClicked() {
                                    seatSelectDialog.dispose();
                                }

                                @Override
                                public void onSeatSelectClicked(String
ticketType) {
                                    typeInfoCheck(ticketType, button, row,
column, seatSelectDialog);
                                }
                            });
                        }
                    });
                }else if("false".equals(screen.getSeatIsEmpty())){
                    button.setUI(new SeatSelectedButtonUI(column + ""));
                }
                if(column <= 2 && column >= 1){
                    button.setBounds(900 - 61*column, 390, 60, 53);
                }else if(column >= 3 && column <= 4){
                    button.setBounds(900 - 61*(column + 1), 390, 60, 53);
                }else if(column >= 5 && column <= 6){
                    button.setBounds(900 - 61*(column + 2), 390, 60, 53);
                }
                add(button);
            }
        }

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.setOpaque(true);
        btnBack.setBounds(50, 365, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        btnBack.addMouseListener(new MouseAdapter() {

            @Override
```

```
            public void mouseClicked(MouseEvent e) {
                listener.onBtnBackClicked();
            }

        });
        add(btnBack);

        JButton btnCart = new JButton("Cart");
        btnCart.setFont(new Font("Arial", Font.BOLD, 20));
        btnCart.setForeground(Color.WHITE);
        btnCart.setOpaque(true);
        btnCart.setBounds(50, 435, 154, 48);
        btnCart.setUI(new ClientNormalButtonUI());
        btnCart.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {

                if(tickets.isEmpty()){
                    return;
                }

                CartDialog cartDialog = new CartDialog(tickets, movImg,
frame.getX(), frame.getY());
                cartDialog.setVisible(true);
                cartDialog.setAlwaysOnTop(true);
                cartDialog.setCartDialogListener(new CartDialogListener() {

                    @Override
                    public void onCancelClicked() {
                        // TODO Auto-generated method stub

                        tickets.clear();
                        cartDialog.dispose();
                        listener.repaintPanel(filmName, "3", timeStr);
                    }
                });
            }
        });

        add(btnCart);

    }
    /**
```

```
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.drawImage(movImg, 30, 10, 200, 296, null);
        graphics2d.setFont(new Font("Arial", Font.PLAIN, 15));

        graphics2d.drawString("Screen 3 at " + timeStr, 70, 340);

        graphics2d.drawLine(300, 5, 300, 530);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 18));
        graphics2d.drawString("SEAT SELECT", 570, 60);

        graphics2d.setFont(new Font("Arial", Font.PLAIN, 18));
        graphics2d.drawString("E", 350, 130);
        graphics2d.drawString("D", 350, 205);
        graphics2d.drawString("C", 350, 280);
        graphics2d.drawString("B", 350, 355);
        graphics2d.drawString("A", 350, 430);

        graphics2d.drawLine(550, 490, 763, 490);
        graphics2d.setFont(new Font("Arial", Font.ITALIC, 15));
        graphics2d.drawString("screen", 630, 508);
        graphics2d.drawLine(550, 515, 763, 515);

        graphics2d.drawImage(selectingImg, 805, 480, 30, 27, null);
        graphics2d.drawString("selecting seat", 840, 500);
        graphics2d.drawImage(selectedImg, 805, 520, 30, 27, null);
        graphics2d.drawString("selected seat", 840, 540);

        super.paint(graphics2d);

    }


}

package com.group99.gui;

import java.awt.Color;
```

```
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseWheelEvent;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.FilmDomParser;
import com.group99.dom.TimeTableDomParser;
import com.group99.gui.TimeSelectDialog.TimeSelectDialogListener;
import com.group99.javabean.Film;
import com.group99.javabean.TimeTable;

/**
 * This is the boundary class of panel of selecting screen.
 * @author group 99
 *
 */
public class ScreenSelectPanel extends JPanel {

    private BufferedImage selectedMovieImg;
    private String selectedName;
    private JFrame frame;
    List<String> timeOfScreen1 = null, timeOfScreen2 = null, timeOfScreen3
= null;

    private ScreenSelectPanelListener listener;
    /**
```

```
     * This is the callback interface of ScreenSelectPanel.
     * @author group 99
     *
     */
    public interface ScreenSelectPanelListener{
        public void onBtnBackClicked();
        public void onTimeSelectClicked(String screenName, String timeStr);
    }
    /**
     * Set ScreenSelectPanel's listener.
     * @param listener A listener of ScreenSelectPanel.
     */
    public void setScreenSelectPanelListener(ScreenSelectPanelListener
listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of ScreenSelectPanel.
     */
    public ScreenSelectPanel() {
    }
    /**
     * This is the constructor of ScreenSelectPanel.
     * @param selectedName The name of selecting movie.
     * @param frame The current frame.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public ScreenSelectPanel(String selectedName, JFrame frame) throws
ParserConfigurationException, SAXException, IOException{
        this.selectedName = selectedName;
        this.frame = frame;
        initData();
    }
    /**
     * Initialize the data of ScreenSelectPanel.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    private void initData() throws ParserConfigurationException,
SAXException, IOException {
```

```
        setLayout(null);

        List<Film> films = FilmDomParser.getFilms();

        for(Film film : films){
            if(film.getFilmName().equals(selectedName)){
                selectedMovieImg = ImageIO.read(new
FileInputStream("./../../resource/" + selectedName + ".jpg"));
            }
        }

        List<TimeTable> tables = TimeTableDomParser.getTimeTables();
        timeOfScreen1 = new ArrayList<String>();
        timeOfScreen2 = new ArrayList<String>();
        timeOfScreen3 = new ArrayList<String>();

        for(TimeTable table : tables){
            if(table.getMovie().equals(selectedName) &&
table.getScreen().equals("1")){
                timeOfScreen1.add(table.getTime());
            }else if(table.getMovie().equals(selectedName) &&
table.getScreen().equals("2")){
                timeOfScreen2.add(table.getTime());
            }else if(table.getMovie().equals(selectedName) &&
table.getScreen().equals("3")){
                timeOfScreen3.add(table.getTime());
            }
        }

        JButton btnScreen1 = new JButton("Screen1");
        if(timeOfScreen1.isEmpty()){
            btnScreen1.setEnabled(false);

            btnScreen1.setUI(new BasicButtonUI(){
                @Override
                public void paint(Graphics g, JComponent c) {
                    g.setColor(new Color(169, 169, 169));
                    g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10,
10);
                    super.paint(g, c);
                }
            });
        }else{
            btnScreen1.setUI(new ClientNormalButtonUI());
```

```
            btnScreen1.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    TimeSelectDialog timeSelectDialog = new
TimeSelectDialog(frame, timeOfScreen1, btnScreen1.getX(), selectedName,
"1");
                    timeSelectDialog.setAlwaysOnTop(true);
                    timeSelectDialog.setVisible(true);
                    timeSelectDialog.setTimeSelectDialogListener(new
TimeSelectDialogListener() {
                        @Override
                        public void onCancelClicked() {
                            timeSelectDialog.dispose();
                        }

                        @Override
                        public void onTimeSelectClicked(String screenName,
String timeStr) {
                            listener.onTimeSelectClicked(screenName,
timeStr);
                            timeSelectDialog.dispose();
                        }
                    });
                }
            });
        }
        btnScreen1.setForeground(Color.WHITE);
        btnScreen1.setFont(new Font("Arial", Font.BOLD, 20));
        btnScreen1.setBounds(381, 468, 154, 48);

        JButton btnScreen2 = new JButton("Screen2");
        if(timeOfScreen2.isEmpty()){
            btnScreen2.setEnabled(false);
            btnScreen2.setUI(new BasicButtonUI(){
                @Override
                public void paint(Graphics g, JComponent c) {
                    g.setColor(new Color(169, 169, 169));
                    g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10,
10);
                    super.paint(g, c);
                }
            });
        }else{
            btnScreen2.setUI(new ClientNormalButtonUI());
```

```
            btnScreen2.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    TimeSelectDialog timeSelectDialog = new
TimeSelectDialog(frame, timeOfScreen2, btnScreen2.getX(), selectedName,
"2");
                    timeSelectDialog.setAlwaysOnTop(true);
                    timeSelectDialog.setVisible(true);
                    timeSelectDialog.setTimeSelectDialogListener(new
TimeSelectDialogListener() {
                        @Override
                        public void onCancelClicked() {
                            timeSelectDialog.dispose();
                        }

                        @Override
                        public void onTimeSelectClicked(String screenName,
String timeStr) {
                            listener.onTimeSelectClicked(screenName,
timeStr);
                            timeSelectDialog.dispose();
                        }
                    });
                }
            });
        }
        btnScreen2.setForeground(Color.WHITE);
        btnScreen2.setFont(new Font("Arial", Font.BOLD, 20));
        btnScreen2.setBounds(555, 468, 154, 48);

        JButton btnScreen3 = new JButton("Screen3");
        if(timeOfScreen3.isEmpty()){
            btnScreen3.setEnabled(false);
            btnScreen3.setUI(new BasicButtonUI(){
                @Override
                public void paint(Graphics g, JComponent c) {
                    g.setColor(new Color(169, 169, 169));
                    g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10,
10);
                    super.paint(g, c);
                }
            });
        }else{
            btnScreen3.setUI(new ClientNormalButtonUI());
```

```
            btnScreen3.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    TimeSelectDialog timeSelectDialog = new
TimeSelectDialog(frame, timeOfScreen3, btnScreen3.getX(), selectedName,
"3");
                    timeSelectDialog.setAlwaysOnTop(true);
                    timeSelectDialog.setVisible(true);
                    timeSelectDialog.setTimeSelectDialogListener(new
TimeSelectDialogListener() {
                        @Override
                        public void onCancelClicked() {
                            timeSelectDialog.dispose();
                        }

                        @Override
                        public void onTimeSelectClicked(String screenName,
String timeStr) {
                            listener.onTimeSelectClicked(screenName,
timeStr);
                            timeSelectDialog.dispose();
                        }
                    });
                }
            });
        }
        btnScreen3.setForeground(Color.WHITE);
        btnScreen3.setFont(new Font("Arial", Font.BOLD, 20));
        btnScreen3.setBounds(729, 468, 154, 48);

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBtnBackClicked();
            }
        });
        btnBack.setOpaque(true);
        btnBack.setBounds(50, 365, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());

        add(btnBack);
```

```
        add(btnScreen1);
        add(btnScreen2);
        add(btnScreen3);

    }

    /**
     * Override the method of void paintComponent(Graphics g).
     */
    @Override
    protected void paintComponent(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.drawImage(selectedMovieImg, 30, 10, 200, 296, null);
        graphics2d.drawLine(300, 5, 300, 500);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 18));
        int count = 0;
        graphics2d.drawString("TIMETABLE", 560, 60);

        graphics2d.setFont(new Font("Arial", Font.ITALIC, 15));
        if(!timeOfScreen1.isEmpty()){
            graphics2d.drawLine(420, 120 + 30 * (count), 820, 120 + 30 *
(count));
            count ++;
            graphics2d.drawString("SCREEN 1 : ", 500, 120 + 30*(count));
            for(int i = 0; i < timeOfScreen1.size(); i++ ){
                graphics2d.drawString(timeOfScreen1.get(i), 700, 120 + 30 *
(count));
                count++;
            }
        }
        if(!timeOfScreen2.isEmpty()){
            graphics2d.drawLine(420, 120 + 30 * (count), 820, 120 + 30 *
(count));
            count ++;
            graphics2d.drawString("SCREEN 2 : ", 500, 120 + 30 * (count));
            for(int i = 0; i < timeOfScreen2.size(); i++ ){
                graphics2d.drawString(timeOfScreen2.get(i), 700, 120 + 30 *
(count));
                count++;
            }
```

```
        }
        if(!timeOfScreen3.isEmpty()){
            graphics2d.drawLine(420, 120 + 30 * (count), 820, 120 + 30 *
(count));
            count++;
            graphics2d.drawString("SCREEN 3 : ", 500, 120 + 30 * (count));
            for(int i = 0; i < timeOfScreen3.size(); i++ ){
                graphics2d.drawString(timeOfScreen3.get(i), 700, 120 + 30 *
(count));
                count++;
            }
        }
        graphics2d.drawLine(420, 120 + 30 * (count), 820, 120 + 30 *
(count));

    }

}
```

```
package com.group99.gui;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.AbstractButton;
import javax.swing.JComponent;
import javax.swing.LookAndFeel;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of selected seat button UI.
 * @author group 99.
 *
 */
public class SeatSelectedButtonUI extends BasicButtonUI{

    private static final int SEAT_IMAGE_SIZE_WIDTH = 60;
```

```
    private static final int SEAT_IMAGE_SIZE_HEIGHT = 53;
    private BufferedImage image;
    /**
     * This is the constructor of SeatSelectedButtonUI.
     * @param number the column of seat id.
     * @throws IOException
     */
    public SeatSelectedButtonUI(String number) throws IOException {
        this.image = ImageIO.read(new FileInputStream("./../../resource/" +
"s" + number + ".png"));
    }

    @Override
    protected void paintText(Graphics g, AbstractButton b, Rectangle
textRect, String text) {
        super.paintText(g, b, textRect, text);
    }
    /**
     * Override the method of void installDefaults(AbstractButton b) to set
the LookAndFeel.
     */
    @Override
    protected void installDefaults(AbstractButton b) {
        super.installDefaults(b);
        LookAndFeel.installProperty(b, "opaque", Boolean.FALSE);
    }
    /**
     * Override the method of void paint(Graphics g, JComponent c) to set
the view when button isn't pressed.
     */
    @Override
    public void paint(Graphics g, JComponent c) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        graphics2d.drawImage(image, 0, 0, SEAT_IMAGE_SIZE_WIDTH,
SEAT_IMAGE_SIZE_HEIGHT, null);
        super.paint(graphics2d, c);
    }

}

package com.group99.gui;
```

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.AbstractButton;
import javax.swing.JComponent;
import javax.swing.LookAndFeel;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of selecting seat button UI.
 * @author group 99.
 *
 */
public class SeatSelectingButtonUI extends BasicButtonUI{

    private static final int SEAT_IMAGE_SIZE_WIDTH = 60;
    private static final int SEAT_IMAGE_SIZE_HEIGHT = 53;
    private BufferedImage image1;
    private BufferedImage image2;
    /**
     * This is the constructor of SeatSelectedButtonUI.
     * @param number the column of seat id.
     * @throws IOException
     */
    public SeatSelectingButtonUI(String number) throws IOException {
        this.image1 = ImageIO.read(new FileInputStream("./../../resource/"
+ number + ".png"));
        this.image2 = ImageIO.read(new FileInputStream("./../../resource/"
+ "s" + number + ".png"));
    }

    @Override
    protected void paintText(Graphics g, AbstractButton b, Rectangle
textRect, String text) {
        super.paintText(g, b, textRect, text);
    }
    /**
```

```
     * Override the method of void installDefaults(AbstractButton b) to set
the LookAndFeel.
     */
    @Override
    protected void installDefaults(AbstractButton b) {
        super.installDefaults(b);
        LookAndFeel.installProperty(b, "opaque", Boolean.FALSE);
    }
    /**
     * Override the method of void paint(Graphics g, JComponent c) to set
the view when button isn't pressed.
     */
    @Override
    public void paint(Graphics g, JComponent c) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        graphics2d.drawImage(image1, 0, 0, SEAT_IMAGE_SIZE_WIDTH,
SEAT_IMAGE_SIZE_HEIGHT, null);
        super.paint(graphics2d, c);
    }
    /**
     * Override the method of void paintButtonPressed(Graphics g,
AbstractButton b) to set the view when button is pressed.
     */
    @Override
    protected void paintButtonPressed(Graphics g, AbstractButton b) {
        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        graphics2d.drawImage(image2, 0, 0, SEAT_IMAGE_SIZE_WIDTH,
SEAT_IMAGE_SIZE_HEIGHT, null);
    }
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.List;
```

```
import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of panel of selecting ticket type.
 * @author group 99
 *
 */
public class SeatTypeSelectDialog extends JDialog{

    SeatSelectDialogListener listener;
    /**
     * This is the callback interface of SeatTypeSelectDialog.
     * @author group 99
     *
     */
    public interface SeatSelectDialogListener{
        public void onCancelClicked();
        public void onSeatSelectClicked(String ticketType);
    }
    /**
     * Set SeatTypeSelectDialog's listener.
     * @param listener A listener of SeatTypeSelectDialog.
     */
    public void setSeatSelectDialogListener(SeatSelectDialogListener
listener){
        this.listener = listener;
    }


    Container contentPane = getContentPane();
    /**
     * This is the constructor of ScreenSelectPanel.
     * @param frame The current frame.
     * @param seatX The x location of seat.
     * @param seatY The y location of seat.
     */
    public SeatTypeSelectDialog(JFrame frame,int seatX, int seatY) {
        setUndecorated(true);
```

```
        setBounds(frame.getX() + seatX - 20, frame.getY() + seatY - 115,
110, 160);
        contentPane.setLayout(null);
        BasicButtonUI buttonUI = new BasicButtonUI(){
            @Override
            public void paint(Graphics g, JComponent c) {
                g.setColor(new Color(255, 130, 71));
                g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
                super.paint(g, c);
            }

            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(192, 192, 192));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
            }
        };

        JButton childBtn = new JButton("Child");
        childBtn.setUI(buttonUI);
        childBtn.setForeground(Color.WHITE);
        childBtn.setBounds(5, 5, 100, 30);
        childBtn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onSeatSelectClicked("child");
            }

        });
        contentPane.add(childBtn);

        JButton adultBtn = new JButton("Adult");
        adultBtn.setUI(buttonUI);
        adultBtn.setForeground(Color.WHITE);
        adultBtn.setBounds(5, 40, 100, 30);
        adultBtn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onSeatSelectClicked("adult");
            }
```

```
        });
        contentPane.add(adultBtn);

        JButton seniorBtn = new JButton("Senior");
        seniorBtn.setUI(buttonUI);
        seniorBtn.setForeground(Color.WHITE);
        seniorBtn.setBounds(5, 75, 100, 30);
        seniorBtn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onSeatSelectClicked("senior");
            }

        });
        contentPane.add(seniorBtn);

        JButton studentBtn = new JButton("Student");
        studentBtn.setUI(buttonUI);
        studentBtn.setForeground(Color.WHITE);
        studentBtn.setBounds(5, 110, 100, 30);
        studentBtn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onSeatSelectClicked("student");
            }

        });
        contentPane.add(studentBtn);

        contentPane.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onCancelClicked();
            }

        });
        JLabel cancelLabel = new JLabel("cancel",JLabel.CENTER);
        cancelLabel.setFont(new Font("Arial", Font.BOLD, 12));
        cancelLabel.setBounds(5, 135, 100, 30);
        contentPane.add(cancelLabel);
    }
```

```
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Calendar;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import com.group99.dom.TicketDomParser;
import com.group99.javabean.Ticket;
/**
 * This is the boundary class of panel of statistics of ticket.
 * @author group 99
 *
 */
public class StatisticsPanel extends JPanel {

    BufferedImage image1, image2, image3, image4, image5;
    int totalSalesOfLA, totalSalesOfKO, totalSalesOfBE, totalSalesOfLO,
```

```
totalSalesOfMO;
    int totalSalesOfChild, totalSalesOfAdult, totalSalesOfSenior,
totalSalesOfStudent;
    float totalProfitOfLA, totalProfitOfKO, totalProfitOfBE,
totalProfitOfLO, totalProfitOfMO;
    float totalProfitOfChild, totalProfitOfAdult, totalProfitOfSenior,
totalProfitOfStudent;
    int totalSales;
    float totalProfit;

    private StatisticsPanelListener listener;
    /**
     * This is the callback interface of StatisticsPanel.
     * @author group 99
     *
     */
    public interface StatisticsPanelListener {
        public void onBackClicked();
    }
    /**
     * Set StatisticsPanel's listener.
     * @param listener A listener of StatisticsPanel.
     */
    public void setStatisticsPanelListener(StatisticsPanelListener
listener) {
        this.listener = listener;
    }
    /**
     * This is the constructor of StatisticsPanel.
     * @throws IOException
     */
    public StatisticsPanel() throws IOException {
        initData();
    }
    /**
     * Initialize the data of StatisticsPanel.
     * @throws IOException
     */
    private void initData() throws IOException {

        image1 = ImageIO.read(new FileInputStream("./../../resource/LA LA
LAND.jpg"));
        image2 = ImageIO.read(new FileInputStream("./../../resource/" +
"KONG-SKULL ISLAND" + ".jpg"));
```

```
        image3 = ImageIO.read(new FileInputStream("./../../resource/" +
"BEAUTY AND THE BEAST" + ".jpg"));
        image4 = ImageIO.read(new FileInputStream("./../../resource/" +
"LOGAN" + ".jpg"));
        image5 = ImageIO.read(new FileInputStream("./../../resource/" +
"MOONLIGHT" + ".jpg"));

        try {
            List<Ticket> tickets = TicketDomParser.getTickets();
            for (Ticket ticket : tickets) {

                totalSales++;
                totalProfit = totalProfit + ticket.getTicketPrice();

                if ("LA LA LAND".equals(ticket.getFilmName())) {
                    totalSalesOfLA++;
                    totalProfitOfLA = totalProfitOfLA +
ticket.getTicketPrice();

                }
                if ("KONG-SKULL ISLAND".equals(ticket.getFilmName())) {
                    totalSalesOfKO++;
                    totalProfitOfKO = totalProfitOfKO +
ticket.getTicketPrice();

                }
                if ("BEAUTY AND THE BEAST".equals(ticket.getFilmName())) {
                    totalSalesOfBE++;
                    totalProfitOfBE = totalProfitOfBE +
ticket.getTicketPrice();

                }
                if ("LOGAN".equals(ticket.getFilmName())) {
                    totalSalesOfLO++;
                    totalProfitOfLO = totalProfitOfLO +
ticket.getTicketPrice();

                }
                if ("MOONLIGHT".equals(ticket.getFilmName())) {
                    totalSalesOfMO++;
                    totalProfitOfMO = totalProfitOfMO +
ticket.getTicketPrice();

                }
```

```
                if ("child".equals(ticket.getTiketType())) {
                    totalSalesOfChild++;
                    totalProfitOfChild = totalProfitOfChild +
ticket.getTicketPrice();

                }
                if ("adult".equals(ticket.getTiketType())) {
                    totalSalesOfAdult++;
                    totalProfitOfAdult = totalProfitOfAdult +
ticket.getTicketPrice();

                }
                if ("senior".equals(ticket.getTiketType())) {
                    totalSalesOfSenior++;
                    totalProfitOfSenior = totalProfitOfSenior +
ticket.getTicketPrice();

                }
                if ("student".equals(ticket.getTiketType())) {
                    totalSalesOfStudent++;
                    totalProfitOfStudent = totalProfitOfStudent +
ticket.getTicketPrice();

                }
            }

        } catch (ParserConfigurationException | SAXException e) {
            e.printStackTrace();
        }

        setLayout(null);

        JButton btnBack = new JButton("Back");
        btnBack.setFont(new Font("Arial", Font.BOLD, 20));
        btnBack.setForeground(Color.WHITE);
        btnBack.setOpaque(true);
        btnBack.setBounds(70, 415, 154, 48);
        btnBack.setUI(new ClientNormalButtonUI());
        btnBack.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onBackClicked();
            }
```

```
        });
        add(btnBack);

        BufferedImage emailImage = ImageIO.read(new
FileInputStream("./../../resource/email.png"));
        JButton emailButton = new JButton();
        emailButton.setBounds(73, 475, 60, 60);
        emailButton.setOpaque(false);
        emailButton.setUI(new BasicButtonUI() {

            @Override
            public void paint(Graphics g, JComponent c) {
                Graphics2D graphics2d = (Graphics2D) g;
                graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                graphics2d.drawImage(emailImage, 0, 0, 60, 60, null);
            }

        });
        emailButton.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                JLabel sendSucessLabel = new JLabel("Send successfully !");
                sendSucessLabel.setFont(new Font("Arial", Font.PLAIN, 15));
                sendSucessLabel.setOpaque(false);
                sendSucessLabel.setForeground(Color.RED);
                sendSucessLabel.setBounds(85, 530, 200, 20);
                add(sendSucessLabel);
                repaint();

                try {
                    Calendar calendar = Calendar.getInstance();
                    FileWriter fw = new FileWriter("./statisticsOf" +
calendar.get(Calendar.YEAR) + "_" +  (calendar.get(Calendar.MONTH) + 1) +
"_" + calendar.get(Calendar.DAY_OF_MONTH) +".txt");
                    BufferedWriter bufw = new BufferedWriter(fw);
                    bufw.write("");
                    bufw.newLine();
                    bufw.write("Film Name              " + "   Total Sales
            " + "Total Profit");
                    bufw.newLine();
                    bufw.write("");
```

```
                    bufw.newLine();
                    bufw.write("LA LA LAND                " + totalSalesOfLA
+ "                $ " + totalProfitOfLA );
                    bufw.newLine();
                    bufw.write("KONG-SKULL ISLAND             " +
totalSalesOfKO + "                $ " + totalProfitOfKO );
                    bufw.newLine();
                    bufw.write("BEAUTY AND THE BEAST              " +
totalSalesOfBE + "                $ " + totalProfitOfBE );
                    bufw.newLine();
                    bufw.write("LOGAN                     " + totalSalesOfLO
+ "                $ " + totalProfitOfLO );
                    bufw.newLine();
                    bufw.write("MOONLIGHT                 " + totalSalesOfMO
+ "                $ " + totalProfitOfMO );
                    bufw.newLine();
                    bufw.newLine();
                    bufw.newLine();
                    bufw.write("");
                    bufw.newLine();
                    bufw.write("Ticket Type               " + "   Total
Sales              " + "Total Profit");
                    bufw.newLine();
                    bufw.write("");
                    bufw.newLine();
                    bufw.write("Adult                     " +
totalSalesOfAdult + "                $ " + totalProfitOfAdult );
                    bufw.newLine();
                    bufw.write("Child                     " +
totalSalesOfChild + "                $ " + totalProfitOfChild );
                    bufw.newLine();
                    bufw.write("Senior                    " +
totalSalesOfSenior + "                $ " + totalProfitOfSenior );
                    bufw.newLine();
                    bufw.write("Student                   " +
totalSalesOfStudent + "                   $ " + totalProfitOfStudent );
                    bufw.newLine();
                    bufw.newLine();
                    bufw.newLine();
                    bufw.write("");
                    bufw.newLine();
                    bufw.write("                          " + "   Total Sales
            " + "Total Profit");
                    bufw.newLine();
```

```
                    bufw.write("");
                    bufw.newLine();
                    bufw.write("                                ″ + totalSales + ″
                   $ ″ + totalProfit );
                    bufw.newLine();
                    bufw.flush();
                    bufw.close();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }

            }

        });
        add(emailButton);

    }
    /**
     * Override the method of void paint(Graphics g).
     */
    @Override
    public void paint(Graphics g) {

        Graphics2D graphics2d = (Graphics2D) g;
        graphics2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        graphics2d.setFont(new Font("Copperplate", Font.BOLD, 32));
        graphics2d.drawString("Statistics", 70, 50);

        graphics2d.drawImage(image4, 47, 130, 60, 89, null);
        graphics2d.drawImage(image1, 202, 130, 60, 89, null);
        graphics2d.drawImage(image3, 73, 170, 70, 104, null);
        graphics2d.drawImage(image5, 165, 170, 70, 104, null);
        graphics2d.drawImage(image2, 112, 210, 80, 119, null);
        graphics2d.drawLine(300, 5, 300, 530);

        graphics2d.setFont(new Font("Arial", Font.BOLD, 25));
        graphics2d.drawString("Email", 150, 513);

        graphics2d.drawLine(400, 40, 870, 40);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 16));
        graphics2d.drawString("Film Name", 455, 60);
        graphics2d.drawString("Total Sales", 610, 60);
```

```
        graphics2d.drawString("Total Profit", 740, 60);
        graphics2d.drawLine(400, 70, 870, 70);

        graphics2d.setFont(new Font("Arial", Font.PLAIN, 13));
        graphics2d.drawString("LA LA LAND", 458, 100);
        graphics2d.drawString(String.valueOf(totalSalesOfLA), 650, 100);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfLA), 760, 100);
        graphics2d.drawString("KONG-SKULL ISLAND", 425, 130);
        graphics2d.drawString(String.valueOf(totalSalesOfKO), 650, 130);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfKO), 760, 130);
        graphics2d.drawString("BEAUTY AND THE BEAST", 415, 160);
        graphics2d.drawString(String.valueOf(totalSalesOfBE), 650, 160);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfBE), 760, 160);
        graphics2d.drawString("LOGAN", 470, 190);
        graphics2d.drawString(String.valueOf(totalSalesOfLO), 650, 190);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfLO), 760, 190);
        graphics2d.drawString("MOONLIGHT", 455, 220);
        graphics2d.drawString(String.valueOf(totalSalesOfMO), 650, 220);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfMO), 760, 220);

        graphics2d.drawLine(400, 270, 870, 270);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 16));
        graphics2d.drawString("Ticket Type", 455, 290);
        graphics2d.drawString("Total Sales", 610, 290);
        graphics2d.drawString("Total Profit", 740, 290);
        graphics2d.drawLine(400, 300, 870, 300);

        graphics2d.setFont(new Font("Arial", Font.PLAIN, 15));
        graphics2d.drawString("Adult", 475, 330);
        graphics2d.drawString(String.valueOf(totalSalesOfAdult), 650, 330);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfAdult), 760, 330);
        graphics2d.drawString("Child", 475, 360);
        graphics2d.drawString(String.valueOf(totalSalesOfChild), 650, 360);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfChild), 760, 360);
        graphics2d.drawString("Senior", 472, 390);
        graphics2d.drawString(String.valueOf(totalSalesOfSenior), 650,
390);
```

```
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfSenior), 760, 390);
        graphics2d.drawString("Student", 469, 420);
        graphics2d.drawString(String.valueOf(totalSalesOfStudent), 650,
420);
        graphics2d.drawString("$ " + String.format("%.2f",
totalProfitOfStudent), 760, 420);

        graphics2d.drawLine(400, 460, 870, 460);
        graphics2d.setFont(new Font("Arial", Font.BOLD, 16));
        graphics2d.drawString("Total Sales", 610, 480);
        graphics2d.drawString("Total Profit", 740, 480);
        graphics2d.drawLine(400, 490, 870, 490);

        graphics2d.setFont(new Font("Arial", Font.PLAIN, 15));
        graphics2d.drawString(String.valueOf(totalSales), 650, 520);
        graphics2d.drawString("$ " + String.format("%.2f", totalProfit),
760, 520);

        super.paint(graphics2d);

    }

}

package com.group99.gui;

import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.List;

import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.plaf.basic.BasicButtonUI;
/**
```

```
 * This is the boundary class of dialog of checking student info.
 * @author group 99
 *
 */
public class StudentInfoCheckDialog extends JDialog{

    StudentInfoCheckDialogListener listener;
    /**
     * This is the callback interface of StudentInfoCheckDialog.
     * @author group 99
     *
     */
    public interface StudentInfoCheckDialogListener{
        public void onCancelClicked();
        public void onOkClicked(String studentName, String studentNum);
    }
    /**
     * Set StudentInfoCheckDialog's listener.
     * @param listener A listener of StudentInfoCheckDialog.
     */
    public void
setStudentInfoCheckDialogListener(StudentInfoCheckDialogListener listener){
        this.listener = listener;
    }


    Container contentPane = getContentPane();
    /**
     * This is the constructor of StudentInfoCheckDialog.
     * @param frame The current frame.
     * @param x The x location of current frame.
     * @param y The y location of current frame.
     */
    public StudentInfoCheckDialog(JFrame frame,int x, int y) {
        setUndecorated(true);
        setBounds(frame.getX() + x - 65, frame.getY() + y - 115, 200, 100);
        contentPane.setLayout(null);
        BasicButtonUI buttonUI = new BasicButtonUI(){
            @Override
            public void paint(Graphics g, JComponent c) {
                g.setColor(new Color(255, 130, 71));
                g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
                super.paint(g, c);
            }
```

```
            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(192, 192, 192));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
            }
        };

        JTextField nameTextField = new JTextField(10);
        nameTextField.setBounds(2, 5, 195, 25);
        nameTextField.setText("Your name...");
        contentPane.add(nameTextField);

        JTextField numTextField = new JTextField(10);
        numTextField.setBounds(2, 35, 195, 25);
        numTextField.setText("Your studentNum...");
        contentPane.add(numTextField);

        JButton studentBtn = new JButton("Ok");
        studentBtn.setUI(buttonUI);
        studentBtn.setForeground(Color.WHITE);
        studentBtn.setBounds(5, 65, 90, 25);
        studentBtn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onOkClicked(nameTextField.getText(),
numTextField.getText());
            }

        });
        contentPane.add(studentBtn);

        JButton cancelBtn = new JButton("Cancel");
        cancelBtn.setUI(buttonUI);
        cancelBtn.setForeground(Color.WHITE);
        cancelBtn.setBounds(105, 65, 90, 25);
        cancelBtn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onCancelClicked();
            }
```

```
        });
        contentPane.add(cancelBtn);
    }
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Label;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.List;

import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.plaf.basic.BasicButtonUI;
/**
 * This is the boundary class of dialog of error hint of student info.
 * @author group 99
 *
 */
public class StudentInfoCheckHintDialog extends JDialog{

    StudentInfoCheckHintDialogListener listener;
    /**
     * This is the callback interface of StudentInfoCheckHintDialog.
     * @author group 99
     *
     */
    public interface StudentInfoCheckHintDialogListener{
        public void onCancelClicked();
    }
    /**
     * Set StudentInfoCheckHintDialog's listener.
     * @param listener A listener of StudentInfoCheckHintDialog.
```

```
     */
    public void
setStudentInfoCheckHintDialogListener(StudentInfoCheckHintDialogListener
listener){
        this.listener = listener;
    }


    Container contentPane = getContentPane();
    /**
     * Initialize the data of StudentInfoCheckHintDialog.
     * @param frame The current frame.
     * @param x The x location of current frame.
     * @param y The y location of current frame.
     */
    public StudentInfoCheckHintDialog(JFrame frame,int x, int y) {
        setUndecorated(true);
        setBounds(frame.getX() + x - 65, frame.getY() + y - 115, 200, 100);
        contentPane.setLayout(null);
        BasicButtonUI buttonUI = new BasicButtonUI(){
            @Override
            public void paint(Graphics g, JComponent c) {
                g.setColor(new Color(255, 130, 71));
                g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
                super.paint(g, c);
            }

            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(192, 192, 192));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
            }
        };

        contentPane.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onCancelClicked();
            }

        });
        JLabel cancelLabel = new JLabel("Your info is
```

```
error!",JLabel.CENTER);
        cancelLabel.setFont(new Font("Arial", Font.BOLD, 15));
        cancelLabel.setBounds(30, 30, 142, 40);
        contentPane.add(cancelLabel);
    }
}

package com.group99.gui;

import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.plaf.basic.BasicButtonUI;

/**
 * This is the boundary class of dialog of selecting time.
 * @author group 99
 *
 */
public class TimeSelectDialog extends JDialog{

    TimeSelectDialogListener listener;
    Container contentPane = getContentPane();
    /**
     * This is the callback interface of TimeSelectDialog.
     * @author group 99
     *
     */
    public interface TimeSelectDialogListener{
```

```
        public void onCancelClicked();
        public void onTimeSelectClicked(String screenName, String timeStr);
    }
    /**
     * Set TimeSelectDialog's listener.
     * @param listener A listener of TimeSelectDialog.
     */
    public void setTimeSelectDialogListener(TimeSelectDialogListener
listener){
        this.listener = listener;
    }
    /**
     * This is the constructor of TimeSelectDialog.
     * @param frame The current frame.
     * @param timeOfScreen The selecting time.
     * @param screenX The current location x.
     * @param filmName The current film name.
     * @param screen The current screen.
     */
    public TimeSelectDialog(JFrame frame, List<String> timeOfScreen, int
screenX, String filmName, String screen) {
        setUndecorated(true);
        setBounds(frame.getX() + screenX + 5, frame.getY() + 320, 154,
180);
        contentPane.setLayout(null);
        int count = 0;
        BasicButtonUI buttonUI = new BasicButtonUI(){
            @Override
            public void paint(Graphics g, JComponent c) {
                g.setColor(new Color(255, 130, 71));
                g.fillRoundRect(0, 0, c.getWidth(), c.getHeight(), 10, 10);
                super.paint(g, c);
            }

            @Override
            protected void paintButtonPressed(Graphics g, AbstractButton b)
{
                g.setColor(new Color(192, 192, 192));
                g.fillRoundRect(0, 0, b.getWidth(), b.getHeight(), 10, 10);
            }
        };
        for(String timeStr : timeOfScreen){

            DateFormat dateFormat = new SimpleDateFormat("HH:mm");
```

```
            Date mDate = null;
            Date currentDate = null;
            try {
                mDate = dateFormat.parse(timeStr);
                currentDate = dateFormat.parse(dateFormat.format(new
Date()));
            } catch (ParseException e1) {
                e1.printStackTrace();
            }
            if(mDate.getTime() > currentDate.getTime()){
                JButton timeBtn = new JButton(timeStr);
                timeBtn.setUI(buttonUI);
                timeBtn.setForeground(Color.WHITE);
                timeBtn.setBounds(6, 10 + 50*count, 142, 41);

                timeBtn.addMouseListener(new MouseAdapter() {

                    @Override
                    public void mouseClicked(MouseEvent e) {
                        listener.onTimeSelectClicked(screen,timeStr);
                    }

                });

                contentPane.add(timeBtn);
                count++;
            }

        }

        contentPane.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onCancelClicked();
            }

        });
        JLabel cancelLabel = new JLabel("cancel",JLabel.CENTER);
        cancelLabel.setFont(new Font("Arial", Font.BOLD, 12));
        cancelLabel.setBounds(6, 10 + 45*count, 142, 41);
        contentPane.add(cancelLabel);
    }
}
```

```
package com.group99.gui;

import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 * This is the boundary class of welcome panel.
 * @author group 99
 *
 */
public class WelcomePanel extends JPanel{

    private WelcomePanelListener listener;
    JLabel label = null;
    JButton customerButton = null;
    JButton administratorButton = null;
    int count;
    /**
     * This is the callback interface of WelcomePanel.
     * @author group 99
     *
     */
    public interface WelcomePanelListener{
        public void onCustomerClicked();
        public void onAdministratorClicked();
    }
    /**
     * Set WelcomePanel's listener.
```

```
 * @param listener A listener of WelcomePanel.
 */
public void setWelcomePanelListener(WelcomePanelListener listener){
    this.listener = listener;
}

/**
 * This is the constructor of WelcomePanel.
 */
public WelcomePanel() {
    initData();
}
/**
 * Initialize the data of WelcomePanel.
 */
private void initData() {

    setLayout(null);

    customerButton = new JButton("Customer");
    customerButton.setUI(new ClientNormalButtonUI());
    customerButton.setForeground(Color.WHITE);
    customerButton.setFont(new Font("Arial", Font.BOLD, 20));

    customerButton.setVisible(true);
    customerButton.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            listener.onCustomerClicked();
        }
    });
    add(customerButton);

    administratorButton = new JButton("Administrator");
    administratorButton.setUI(new ClientNormalButtonUI());
    administratorButton.setForeground(Color.WHITE);
    administratorButton.setFont(new Font("Arial", Font.BOLD, 17));

    administratorButton.setVisible(true);
    administratorButton.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            listener.onAdministratorClicked();
        }
```

```
        });
        add(administratorButton);

        label = new JLabel();
        label.setFont(new Font("Copperplate", Font.PLAIN, 80));
        label.setText("99 Kiosk");
        label.setOpaque(false);
        configTimeArea();
        add(label);

    }

    /**
     * This method creates a timer task to update the time per 10 ms.
     */
    private void configTimeArea() {
        Timer tmr = new Timer();
        tmr.scheduleAtFixedRate(new JLabelTimerTask(), new Date(), 10);
    }

    /**
     * This is a class of timer task.
     * @author group 99
     *
     */
    protected class JLabelTimerTask extends TimerTask {

        @Override
        public void run() {

            if(count <= 90 && count >= 10){
                label.setBounds(310, 290 - count, 800, 80);
                customerButton.setBounds(200, 540 - count, 155, 50);
                administratorButton.setBounds(600, 540 - count, 155, 50);
            }else if(count > 90){
                return;
            }

            count ++;
        }
    }

}
```

```
package com.group99.dom;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.group99.javabean.Administrator;
/**
 * DOM parser of administrators.xml
 * @author group 99
 *
 */
public class AdministratorDomParser {
    /**
     * Get a List of Administrator.
     * @return List of object of Administrator.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<Administrator> getAdministrators()
            throws ParserConfigurationException, SAXException, IOException {

        File file = new File("./administrators.xml");
        InputStream inputStream = new FileInputStream(file);

        List<Administrator> administrators = new
ArrayList<Administrator>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
```

```
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element administratorElements = document.getDocumentElement();

        NodeList administratorNodes =
administratorElements.getElementsByTagName("administrator");

        for (int i = 0; i < administratorNodes.getLength(); i++) {
            Administrator administrator = new Administrator();

            Element administratorElement = (Element)
administratorNodes.item(i);

            NodeList childNodes = administratorElement.getChildNodes();

            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
                    if ((childNodes.item(j).getNodeName()).equals("id")) {

    administrator.setId(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("password")) {

    administrator.setPassword(childNodes.item(j).getTextContent());
                    }
                }
            }
            administrators.add(administrator);
        }
        return administrators;
    }
}

package com.group99.dom;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
```

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.group99.javabean.BankAccount;
/**
 * DOM parser of bank_accounts.xml
 * @author group 99
 *
 */
public class BankAccountDomParser {

    /**
     * To update bank account by providing account number.
     * @param updateAccNo Which account number you want to update.
     * @param updateAttribute The attribute you want to update.
     * @param updateValue The value you want to update.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     * @throws TransformerException
     */
    public static void updateAccounts(String updateAccNo, String
updateAttribute, String updateValue)
            throws ParserConfigurationException, SAXException, IOException,
TransformerException {

        File file = new File("bank_accounts.xml");
        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(file);
```

```
        document.getDocumentElement().normalize();

        NodeList accountNode = document.getElementsByTagName("account");
        Element element = null;

        for(int i=0; i<accountNode.getLength();i++){
            element = (Element) accountNode.item(i);

    if(element.getElementsByTagName("accountNum").item(0).getFirstChild().g
etTextContent().equals(updateAccNo)){
                 Node name =
element.getElementsByTagName(updateAttribute).item(0).getFirstChild();
                 name.setNodeValue(updateValue);

                document.getDocumentElement().normalize();
                TransformerFactory transformerFactory =
TransformerFactory.newInstance();
                Transformer transformer =
transformerFactory.newTransformer();
                DOMSource source = new DOMSource(document);
                StreamResult result = new StreamResult(new
File("bank_accounts.xml"));
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                transformer.transform(source, result);
                 break;
            }else{
                if(i == accountNode.getLength() - 1)
                    System.out.println("The account don't exist!");
            }
        }
    }
    /**
     *  Get a List of BankAccount.
     * @return List of object of BankAccount.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<BankAccount> getBankAccounts()
            throws ParserConfigurationException, SAXException, IOException {

        File file = new File("./bank_accounts.xml");
        InputStream inputStream = new FileInputStream(file);
```

```
        List<BankAccount> bankAccounts = new ArrayList<BankAccount>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element accountElements = document.getDocumentElement();

        NodeList accountNodes =
accountElements.getElementsByTagName("account");

        for (int i = 0; i < accountNodes.getLength(); i++) {
            BankAccount BankAccount = new BankAccount();

            Element accountElement = (Element) accountNodes.item(i);

            NodeList childNodes = accountElement.getChildNodes();

            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
                    if ((childNodes.item(j).getNodeName()).equals("name")) {

    BankAccount.setName(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("accountNum")) {

    BankAccount.setAccountNum(childNodes.item(j).getTextContent());
                    }else if
((childNodes.item(j).getNodeName()).equals("accountPassword")) {

    BankAccount.setAccountPassword(childNodes.item(j).getTextContent());
                    }else if
((childNodes.item(j).getNodeName()).equals("balance")) {

    BankAccount.setBalance(Float.parseFloat(childNodes.item(j).getTextConte
nt()));
                    }
                }
            }
            bankAccounts.add(BankAccount);
        }
        return bankAccounts;
    }
```

```
}

package com.group99.dom;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.group99.javabean.Film;
/**
 * DOM parser of films.xml
 * @author group 99
 *
 */
public class FilmDomParser {

    public static String FILM_NAME = "filmName";
    public static String FILM_DURATION = "filmDuration";
    public static String FILM_DIRECTOR = "filmDirector";
    public static String FILM_STARS = "filmStars";
    public static String FILM_PRICE = "filmPrice";
    public static String FILM_STORYLINE = "filmStoryline";

    /**
     * To update the films by providing the film name you want update.
```

```
     * @param updateFilmName The film name you want to update.
     * @param updateAttribute The attribute you want to update.
     * @param updateValue  The value you want to update.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     * @throws TransformerException
     */
    public static void updateFilms(String updateFilmName, String
updateAttribute, String updateValue)
            throws ParserConfigurationException, SAXException, IOException,
TransformerException {

        File file = new File("./films.xml");
        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(file);
        document.getDocumentElement().normalize();

        NodeList filmNode = document.getElementsByTagName("film");
        Element element = null;

        for(int i=0; i<filmNode.getLength();i++){
            element = (Element) filmNode.item(i);

    if(element.getElementsByTagName("filmName").item(0).getFirstChild().get
TextContent().equals(updateFilmName)){
                Node name =
element.getElementsByTagName(updateAttribute).item(0).getFirstChild();
                name.setNodeValue(updateValue);

                document.getDocumentElement().normalize();
                TransformerFactory transformerFactory =
TransformerFactory.newInstance();
                Transformer transformer =
transformerFactory.newTransformer();
                DOMSource source = new DOMSource(document);
                StreamResult result = new StreamResult(new
File("films.xml"));
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                transformer.transform(source, result);
                break;
```

```
            }else{
                if(i == filmNode.getLength() - 1)
                    System.out.println("The film don't exist!\tXML file
updated failed!");
            }
        }
    }

    /**
     * Get a List of Film.
     * @return List of object of Film.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<Film> getFilms()
            throws ParserConfigurationException, SAXException, IOException {

        File file = new File("./films.xml");
        InputStream inputStream = new FileInputStream(file);

        List<Film> films = new ArrayList<Film>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element filmsElements = document.getDocumentElement();

        NodeList filmNodes = filmsElements.getElementsByTagName("film");

        for (int i = 0; i < filmNodes.getLength(); i++) {
            Film film = new Film();

            Element filmElement = (Element) filmNodes.item(i);


    film.setFilmId(Integer.parseInt(filmElement.getAttribute("filmId")));

            NodeList childNodes = filmElement.getChildNodes();

            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
```

```
                    if
((childNodes.item(j).getNodeName()).equals("filmName")) {

    film.setFilmName(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("filmDuration")) {

    film.setFilmDuration(Integer.parseInt(childNodes.item(j).getTextContent
()));
                    } else if
((childNodes.item(j).getNodeName()).equals("filmDirector")) {

    film.setFilmDirector(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("filmStars")) {

    film.setFilmStars(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("filmStoryline")) {

    film.setFilmStoryline(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("filmPrice")) {

    film.setFilmPrice(Float.parseFloat(childNodes.item(j).getTextContent())
);
                    } else if
((childNodes.item(j).getNodeName()).equals("filmType")) {

    film.setFilmType(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("filmScore")) {

    film.setFilmScore(childNodes.item(j).getTextContent());
                    }
                }
            }
            films.add(film);
        }
        return films;
    }
}

package com.group99.dom;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.group99.javabean.Screen;
/**
 * DOM parser of screen*At*.xml.
 * @author group 99
 *
 */
public class ScreenSeatDomParser {
    /**
     * The update the seat by providing fileName and seat id.
     * @param fileName The file name you want to update.
     * @param updateSeatId The id of the seat you want to update.
     * @param updateValue The value you want to update(true of false).
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     * @throws TransformerException
     */
    public static void updateScreenSeat(String fileName, String
updateSeatId, String updateValue)
            throws ParserConfigurationException, SAXException, IOException,
```

```
TransformerException {

        File file = new File("./" + fileName);
        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(file);
        document.getDocumentElement().normalize();

        NodeList filmNode = document.getElementsByTagName("seat");
        Element element = null;

        for(int i=0; i<filmNode.getLength();i++){
            element = (Element) filmNode.item(i);

    if(element.getElementsByTagName("seatId").item(0).getFirstChild().getTe
xtContent().equals(updateSeatId)){
                Node name =
element.getElementsByTagName("seatIsEmpty").item(0).getFirstChild();
                name.setNodeValue(updateValue);

                document.getDocumentElement().normalize();
                TransformerFactory transformerFactory =
TransformerFactory.newInstance();
                Transformer transformer =
transformerFactory.newTransformer();
                DOMSource source = new DOMSource(document);
                StreamResult result = new StreamResult(new File(fileName));
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                transformer.transform(source, result);

                break;
            }else{
                if(i == filmNode.getLength() - 1)
                    System.out.println("The film don't exist!\tXML file
updated failed!");
            }
        }
    }


    /**
     * Get a List of Screen by providing file name.
```

```
     * @param fileName The name of a file you want to update.
     * @return A List of Screen.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<Screen> getScreen(String fileName)
            throws ParserConfigurationException, SAXException, IOException {

        File file = new File("./" + fileName);
        InputStream inputStream = new FileInputStream(file);

        List<Screen> screens = new ArrayList<Screen>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element screenElements = document.getDocumentElement();

        NodeList screenNodes = screenElements.getElementsByTagName("seat");

        for (int i = 0; i < screenNodes.getLength(); i++) {
            Screen screen = new Screen();

            Element screenElement = (Element) screenNodes.item(i);

            NodeList childNodes = screenElement.getChildNodes();

            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
                    if ((childNodes.item(j).getNodeName()).equals("seatId"))
{

    screen.setSeatId(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("seatMov")) {

    screen.setSeatMov(childNodes.item(j).getTextContent());
                    }else if
((childNodes.item(j).getNodeName()).equals("seatIsEmpty")) {

    screen.setSeatIsEmpty(childNodes.item(j).getTextContent());
```

```
                    }
                }
            }
            screens.add(screen);
        }
        return screens;
    }
}
```

```
package com.group99.dom;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.group99.javabean.Student;
/**
 * DOM parser of students.xml.
 * @author group 99
 *
 */
public class StudentDomParser {
    /**
     * Get a List of Student.
     * @return A List of Student.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<Student> getStudents()
            throws ParserConfigurationException, SAXException, IOException {
```

```
        File file = new File("./students.xml");
        InputStream inputStream = new FileInputStream(file);

        List<Student> Students = new ArrayList<Student>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element studentElements = document.getDocumentElement();

        NodeList studentNodes =
studentElements.getElementsByTagName("student");

        for (int i = 0; i < studentNodes.getLength(); i++) {
            Student Student = new Student();

            Element studentElement = (Element) studentNodes.item(i);

            NodeList childNodes = studentElement.getChildNodes();

            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
                    if ((childNodes.item(j).getNodeName()).equals("name")) {

    Student.setName(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("studentNum")) {

    Student.setStudentNum(childNodes.item(j).getTextContent());
                    }
                }
            }
            Students.add(Student);
        }
        return Students;
    }
}

package com.group99.dom;

import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;
import org.xml.sax.SAXException;

import com.group99.javabean.Ticket;
/**
 * DOM parser of tickets.xml.
 * @author group 99
 *
 */
public class TicketDomParser {
    /**
     * Creating a ticket by providing a Ticket.
     * @param ticket A object of Ticket.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     * @throws TransformerException
     */
    public static void createTicket(Ticket ticket)
            throws ParserConfigurationException, SAXException, IOException,
TransformerException {
        DocumentBuilderFactory dbfactory =
DocumentBuilderFactory.newInstance();
```

```
        DocumentBuilder dbbuilder = dbfactory.newDocumentBuilder();
        Document doc = dbbuilder.parse(new File("./tickets.xml"));

        Element ticketElement = doc.createElement("ticket");

        Element filmNameElement = doc.createElement("filmName");
        Element ticketIdElement = doc.createElement("ticketId");
        Element timeStrElement = doc.createElement("timeStr");
        Element screenNameElement = doc.createElement("screenName");
        Element ticketTypeElement = doc.createElement("ticketType");
        Element seatLocationElement = doc.createElement("seatLocation");
        Element ticketPriceElement = doc.createElement("ticketPrice");
        Element studentIdElement = doc.createElement("studentId");

        Text filmName =
doc.createTextNode(String.valueOf(ticket.getFilmName()));
        Text ticketId =
doc.createTextNode(String.valueOf(ticket.getTicketId()));
        Text timeStr =
doc.createTextNode(String.valueOf(ticket.getTimeStr()));
        Text screenName =
doc.createTextNode(String.valueOf(ticket.getScreenName()));
        Text ticketType =
doc.createTextNode(String.valueOf(ticket.getTiketType()));
        Text seatLocation =
doc.createTextNode(String.valueOf(ticket.getSeatLocation()));
        Text ticketPrice =
doc.createTextNode(String.valueOf(ticket.getTicketPrice()));
        Text studentId = doc.createTextNode(ticket.getStudentId());

        ticketElement.appendChild(filmNameElement).appendChild(filmName);
        ticketElement.appendChild(ticketIdElement).appendChild(ticketId);
        ticketElement.appendChild(timeStrElement).appendChild(timeStr);

    ticketElement.appendChild(screenNameElement).appendChild(screenName);

    ticketElement.appendChild(ticketTypeElement).appendChild(ticketType);

    ticketElement.appendChild(seatLocationElement).appendChild(seatLocation
);

    ticketElement.appendChild(ticketPriceElement).appendChild(ticketPrice);
        ticketElement.appendChild(studentIdElement).appendChild(studentId);
```

```
        doc.getDocumentElement().appendChild(ticketElement);

        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();

        DOMSource source = new DOMSource(doc);
        StreamResult streamResult = new StreamResult(new
File("./tickets.xml"));
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.transform(source, streamResult);
    }
    /**
     * Get a List of Ticket.
     * @return  A List of Ticket.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<Ticket> getTickets() throws
ParserConfigurationException, SAXException, IOException {

        File file = new File("./tickets.xml");
        InputStream inputStream = new FileInputStream(file);

        List<Ticket> tickets = new ArrayList<Ticket>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element ticketElenments = document.getDocumentElement();

        NodeList ticketNodes =
ticketElenments.getElementsByTagName("ticket");

        for (int i = 0; i < ticketNodes.getLength(); i++) {
            Ticket ticket = new Ticket();

            Element ticketElement = (Element) ticketNodes.item(i);

            NodeList childNodes = ticketElement.getChildNodes();
```

```
            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
                    if
((childNodes.item(j).getNodeName()).equals("filmName")) {

    ticket.setFilmName(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("ticketId")) {

    ticket.setTicketId(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("timeStr")) {

    ticket.setTimeStr(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("screenName")) {

    ticket.setScreenName(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("ticketType")) {

    ticket.setTiketType(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("seatLocation")) {

    ticket.setSeatLocation(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("ticketPrice")) {

    ticket.setTicketPrice(Float.parseFloat(childNodes.item(j).getTextConten
t()));
                    }else if
((childNodes.item(j).getNodeName()).equals("studentId")) {

    ticket.setStudentId(childNodes.item(j).getTextContent());
                    }
                }
            }
            tickets.add(ticket);
        }
        return tickets;
    }
}
```

```
package com.group99.dom;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.group99.javabean.TimeTable;
/**
 * DOM parser of timetables.xml.
 * @author group 99
 *
 */
public class TimeTableDomParser {
    /**
     * Get a List of TimeTable.
     * @return A List of TimeTable.
     * @throws ParserConfigurationException
     * @throws SAXException
     * @throws IOException
     */
    public static List<TimeTable> getTimeTables()
            throws ParserConfigurationException, SAXException, IOException {

        File file = new File("./timetables.xml");
        InputStream inputStream = new FileInputStream(file);

        List<TimeTable> timeTables = new ArrayList<TimeTable>();

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
```

```
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(inputStream);
        Element timeElements = document.getDocumentElement();

        NodeList timeNodes =
timeElements.getElementsByTagName("timetable");

        for (int i = 0; i < timeNodes.getLength(); i++) {
            TimeTable timetable = new TimeTable();

            Element timeElement = (Element) timeNodes.item(i);

            NodeList childNodes = timeElement.getChildNodes();

            for (int j = 0; j < childNodes.getLength(); j++) {
                if (childNodes.item(j).getNodeType() == Node.ELEMENT_NODE) {
                    if ((childNodes.item(j).getNodeName()).equals("time")) {

    timetable.setTime(childNodes.item(j).getTextContent());
                    } else if
((childNodes.item(j).getNodeName()).equals("movie")) {

    timetable.setMovie(childNodes.item(j).getTextContent());
                    }else if
((childNodes.item(j).getNodeName()).equals("screen")) {

    timetable.setScreen(childNodes.item(j).getTextContent());
                    }else if
((childNodes.item(j).getNodeName()).equals("id")) {

    timetable.setId(childNodes.item(j).getTextContent());
                    }
                }
            }
            timeTables.add(timetable);
        }
        return timeTables;
    }
}

package com.group99.javabean;
/**
 * The entity class of administrator.
 * @author group 99
```

```
 *
 */
public class Administrator {

    private String id;
    private String password;

    public Administrator() {
    }
    /**
     * This is the constructor of Administrator.
     * @param id ID of administrator.
     * @param password Password of administrator.
     */
    public Administrator(String id, String password) {
        this.id = id;
        this.password = password;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String toString() {
        return "Administrator [id=" + id + ", password=" + password + "]";
    }

}

package com.group99.javabean;
/**
```

```
 * The entity class of bank account.
 * @author group 99
 *
 */
public class BankAccount {

    private String name;
    private String accountNum;
    private String accountPassword;
    private float balance;
    /**
     * This is the constructor of BankAccount.
     * @param name The owner name of current bank account.
     * @param accountNum Account number.
     * @param accountPassword Account password.
     * @param balance Account balance.
     */
    public BankAccount(String name, String accountNum, String
accountPassword, float balance) {
        super();
        this.name = name;
        this.accountNum = accountNum;
        this.accountPassword = accountPassword;
        this.balance = balance;
    }

    public BankAccount() {

    }

    public float getBalance() {
        return balance;
    }

    public void setBalance(float balance) {
        this.balance = balance;
    }



    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getAccountNum() {
        return accountNum;
    }

    public void setAccountNum(String accountNum) {
        this.accountNum = accountNum;
    }

    public String getAccountPassword() {
        return accountPassword;
    }

    public void setAccountPassword(String accountPassword) {
        this.accountPassword = accountPassword;
    }

    @Override
    public String toString() {
        return "BankAccount [name=" + name + ", accountNum=" + accountNum +
", accountPassword=" + accountPassword
                + "]";
    }

}

package com.group99.javabean;
/**
 * The entity class of film.
 * @author group 99
 *
 */
public class Film {

    private int filmId;
    private String filmName;
    private int filmDuration;
    private String filmDirector;
    private String filmStars;
    private Float filmPrice;
    private String filmStoryline;
```

```
    private String filmType;
    private String filmScore;

    public Film(){

    }
    /**
     * This is the constructor of Film.
     * @param filmId The id of film.
     * @param filmName The name of film.
     * @param filmDuration The duration of film.
     * @param filmDirector The director of film.
     * @param filmStars The stars of film.
     * @param filmPrice The price of film.
     * @param filmStoryline The story line of film.
     * @param filmType The type of film.
     * @param filmScore The score of film.
     */
    public Film(int filmId, String filmName, int filmDuration, String
filmDirector, String filmStars, Float filmPrice,
            String filmStoryline, String filmType, String filmScore) {
        super();
        this.filmId = filmId;
        this.filmName = filmName;
        this.filmDuration = filmDuration;
        this.filmDirector = filmDirector;
        this.filmStars = filmStars;
        this.filmPrice = filmPrice;
        this.filmStoryline = filmStoryline;
        this.filmType = filmType;
        this.filmScore = filmScore;
    }

    public String getFilmType() {
        return filmType;
    }

    public void setFilmType(String filmType) {
        this.filmType = filmType;
    }

    public String getFilmScore() {
        return filmScore;
    }
```

```
public void setFilmScore(String filmScore) {
    this.filmScore = filmScore;
}

public int getFilmId() {
    return filmId;
}

public void setFilmId(int filmId) {
    this.filmId = filmId;
}

public String getFilmName() {
    return filmName;
}

public void setFilmName(String filmName) {
    this.filmName = filmName;
}

public int getFilmDuration() {
    return filmDuration;
}

public void setFilmDuration(int filmDuration) {
    this.filmDuration = filmDuration;
}

public String getFilmDirector() {
    return filmDirector;
}

public void setFilmDirector(String filmDirector) {
    this.filmDirector = filmDirector;
}

public String getFilmStars() {
    return filmStars;
}

public void setFilmStars(String filmStars) {
    this.filmStars = filmStars;
}
```

```
    public Float getFilmPrice() {
        return filmPrice;
    }

    public void setFilmPrice(Float filmPrice) {
        this.filmPrice = filmPrice;
    }

    public String getFilmStoryline() {
        return filmStoryline;
    }

    public void setFilmStoryline(String filmStoryline) {
        this.filmStoryline = filmStoryline;
    }


}

package com.group99.javabean;
/**
 * The entity class of screen.
 * @author group 99
 *
 */
public class Screen {

    private String seatId;
    private String seatMov;
    private String seatIsEmpty;

    public Screen() {
    }
    /**
     * This is the constructor of Screen.
     * @param seatId The seat ID.
     * @param seatMov The current movie name.
     * @param seatIsEmpty Whether the seat is empty.
     */
    public Screen(String seatId, String seatMov, String seatIsEmpty) {
        this.seatId = seatId;
        this.seatMov = seatMov;
        this.seatIsEmpty = seatIsEmpty;
```

```
    }

    public String getSeatId() {
        return seatId;
    }

    public void setSeatId(String seatId) {
        this.seatId = seatId;
    }

    public String getSeatMov() {
        return seatMov;
    }

    public void setSeatMov(String seatMov) {
        this.seatMov = seatMov;
    }

    public String getSeatIsEmpty() {
        return seatIsEmpty;
    }

    public void setSeatIsEmpty(String seatIsEmpty) {
        this.seatIsEmpty = seatIsEmpty;
    }

    @Override
    public String toString() {
        return "Screen [seatId=" + seatId + ", seatMov=" + seatMov + ",
seatIsEmpty=" + seatIsEmpty + "]";
    }

}

package com.group99.javabean;
/**
 * The entity class of student.
 * @author group 99
 *
 */
public class Student {

    private String name;
    private String studentNum;
```

```
    public Student() {
    }
    /**
     * This is the constructor of Student.
     * @param name The student name.
     * @param studentNum The student number.
     */
    public Student(String name, String studentNum) {
        this.name = name;
        this.studentNum = studentNum;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getStudentNum() {
        return studentNum;
    }

    public void setStudentNum(String studentNum) {
        this.studentNum = studentNum;
    }

    @Override
    public String toString() {
        return "Student [name=" + name + ", studentNum=" + studentNum +
"]";
    }


}

package com.group99.javabean;
/**
 * The entity class of ticket.
 * @author group 99
 *
 */
```

```
public class Ticket {

    private String filmName;
    private String ticketId;
    private String timeStr;
    private String screenName;
    private String tiketType;
    private String seatLocation;
    private float ticketPrice;
    private String studentId;

    public Ticket() {
    }
    /**
     * This is the constructor of Ticket.
     * @param filmName The name of film.
     * @param ticketId The ID of ticket.
     * @param timeStr The time of ticket.
     * @param screenName The screen of ticket.
     * @param tiketType The type of film.
     * @param seatLocation The location of seat.
     * @param ticketPrice The price of ticket.
     * @param studentId The student ID.
     */
    public Ticket(String filmName, String ticketId, String timeStr, String
screenName, String tiketType,
            String seatLocation, float ticketPrice, String studentId) {
        super();
        this.filmName = filmName;
        this.ticketId = ticketId;
        this.timeStr = timeStr;
        this.screenName = screenName;
        this.tiketType = tiketType;
        this.seatLocation = seatLocation;
        this.ticketPrice = ticketPrice;
        this.studentId = studentId;
    }

    public String getStudentId() {
        return studentId;
    }

    public void setStudentId(String studentId) {
        this.studentId = studentId;
```

```
}

public String getSeatLocation() {
    return seatLocation;
}

public void setSeatLocation(String seatLocation) {
    this.seatLocation = seatLocation;
}

public String getFilmName() {
    return filmName;
}

public void setFilmName(String filmName) {
    this.filmName = filmName;
}

public String getTicketId() {
    return ticketId;
}

public void setTicketId(String ticketId) {
    this.ticketId = ticketId;
}

public String getTimeStr() {
    return timeStr;
}

public void setTimeStr(String timeStr) {
    this.timeStr = timeStr;
}

public String getScreenName() {
    return screenName;
}

public void setScreenName(String screenName) {
    this.screenName = screenName;
}

public String getTiketType() {
    return tiketType;
```

```
    }

    public void setTiketType(String tiketType) {
        this.tiketType = tiketType;
    }

    public float getTicketPrice() {
        return ticketPrice;
    }

    public void setTicketPrice(float ticketPrice) {
        this.ticketPrice = ticketPrice;
    }

    @Override
    public String toString() {
        return "Ticket [filmName=" + filmName + ", ticketId=" + ticketId +
", timeStr=" + timeStr + ", screenName="
                + screenName + ", tiketType=" + tiketType + ",
seatLocation=" + seatLocation + ", ticketPrice="
                + ticketPrice + ", studentId=" + studentId + "]";
    }


}

package com.group99.javabean;
/**
 * The entity class of timetable.
 * @author group 99
 *
 */
public class TimeTable {

    private String id;
    private String screen;
    private String time;
    private String movie;

    public TimeTable(){

    }
    /**
     * This is the constructor of TimeTable.
```

```
     * @param id ID of timetable.
     * @param screen Current screen.
     * @param time Current time.
     * @param movie current movie name.
     */
    public TimeTable(String id, String screen, String time, String movie) {
        super();
        this.id = id;
        this.screen = screen;
        this.time = time;
        this.movie = movie;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getScreen() {
        return screen;
    }

    public void setScreen(String screen) {
        this.screen = screen;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getMovie() {
        return movie;
    }

    public void setMovie(String movie) {
        this.movie = movie;
    }
```

```
    @Override
    public String toString() {
        return "TimeTable [id=" + id + ", screen=" + screen + ", time=" +
time + ", movie=" + movie + "]";
    }

}
```