**[2018-2 Domain Analysis and SW Design]**

**Project Elaboration Phase Iteration 1 Report**

**Ajou Central Library Management System**

**Team. 치즈수플레**

**Jiseon Lee,** 201621491, [acr8286@gmail.com](mailto:acr8286@gmail.com)

**Seongmin Ahn**, 201521514, [ja036@ajou.ac.kr](mailto:ja036@ajou.ac.kr)

**AnKyu Lee**, 201421155, [lang0909@ajou.ac.kr](mailto:lang0909@ajou.ac.kr)

**Juhee Maeng**, 201521141, [mjh0208@ajou.ac.kr](mailto:mjh0208@ajou.ac.kr)

**Yonggyu Kim**, 201321187, [dydrb2535@ajou.ac.kr](mailto:dydrb2535@ajou.ac.kr)

**Table of Contents**

1. **Vision···································································································4**
   1. Introduction**····················································································4**
   2. Business Opportunity**······································································4**
   3. Problem Statement**··········································································4**
   4. Product Position Statement**······························································4**
   5. User Summary**················································································5**
   6. User Level Goal**·············································································5**
   7. Product Overview & Product Perspective**·········································5**
   8. Summary of System Feature**····························································5**
2. **Requirements·······················································································6**
   1. Use Case Diagram**··········································································6**
   2. Use Case Text**·················································································6**
      1. Functional Requirements**···························································6**
      2. Use Case Model**········································································6**
      3. Non-Functional Requirements**····················································9**
3. **Analysis Modeling··············································································10**
   1. Domain Model Diagram**································································10**
   2. System Sequence Diagram**····························································11**
      1. Use Case 1. EnterLibrary**·························································11**
      2. Use Case 2. ExitLibrary**···························································12**
      3. Use Case3. ManagePenalty**······················································14**
      4. Use Case4. ReserveSeat**······················································14**
      5. Use Case5. ExtendSeat**······················································14**
      6. Use Case6. ExtendSeat **······················································14**
      7. Use Case3. UpdatedInfo**·····················································14**
4. **Design Modeling···············································································15**
   1. <Use Case 1. EnterLibrary> Realization**·········································15**
   2. <Use Case 2. ExitLibrary> Realization**···········································16**
   3. <Use Case 3. ManagePenalty > Realization**···································16**
   4. <Use Case 4. ReserveSeat > Realization**·····························16**
   5. <Use Case 5. ExtendSeat > Realization**···································16**
   6. <Use Case 6. ExtendSeat > Realization**···································16**
5. **Architecture**
   1. **Introduction································································**
   2. **Architectural Factors································································**
   3. **Logical View································································**
   4. **Process View································································**
   5. **Deployment View···································································**
   6. **UseCase View·····································································**
   7. **Data View·········································································**
6. **Conclusion·····················································································**
7. **Reference··························································································18**
8. **Appendix··························································································18**
9. **Revision History················································································19**

**Table of Figures**

< Figure1. Context Diagram >**······································································5**

< Figure2. Use Case Diagram >**····································································6**

< Figure3. Domain Model Diagram >**·························································11**

< Figure 4. Use Case 1. SSD >**···································································12**

< Figure 5. Use Case 2. SSD >**···································································13**

< Figure 6. Use Case 3. SSD >**···································································14**

< Figure 7. Use Case 1. DSD >**··································································16**

< Figure 8. Use Case 1. DCD >**··································································17**

< Figure 9. Use Case 2. DSD >**··································································17**

< Figure 10. Use Case 2. DCD >**································································18**

**1. Vision**

1.1. Introduction

The existing Ajou Central Library access system is required to take a student ID card when entering. But it does not require a student ID card when leaving because the entrance is opened automatically. Administrators cannot compare the faces and photos of students entering the library for 24 hours a day. Therefore, there is a problem that different people can come in using the same student ID card. There is also another problem inside the library. The reading rooms are available on each floor of the library, and students can be issued a seat from the first floor and use it for a certain period of time. Likewise, it is possible to return the seat through terminals in first floor. The second problem is that students leave the reading room for a long time or leave without return the seat.

We will solve the problem that is duplication ID and not return seat. To distinguish duplication ID, we should compare it to information that user is in the library. To distinguish not return seat, we should install out gate machine and check user out time. And we will provide LendSeat service. It is biggest feature for users who will be exit of the library for a long time. They did not have to leave their bags.

It will be able to provide more comfortable and more space for Ajou University students who use the reading room. In addition, the management will be easier to manage. This complemented system is mainly for user’s satisfaction. These features will give users to use the better Reading Room.

1.2. Business Opportunity

The problem with using current reading room is that users who do not use seat reserve their seats and leave reading room. This problem becomes severe during the test period. At this time, the university community will be aware of complaints about vacant seat with baggage.

In addition, Enter to the incorrect way is a matter of worrying as a library manager. He should prevent user from borrowing a ID card from improper use of the library.

This system helps administrator to resolve problems with complaints and incorrect enter.

1.3. Problem Statement

The existing AJou Central Library access system is required to take a student ID card when entering. But it does not require a student ID card because the entrance is opened automatically when leaving. In connection with student access to the library, there is a problem that it is not known whether or not the student is using the library. Therefore, it is possible to use duplicate student IDs, so that many people can use the library by borrowing the student ID card of another student.

In the case of the reading room, the basic usable time is 4 hours, and if it is insufficient, the time can be further extended. The more students who leave the seat for a long time or do not return the seat, the more students cannot use the room even if there are enough seats available.

1.4. Product Position Statement

The library administrators mainly use this system in managing the entrance of student and the reservation of seats in reading room. He’s goal is to improve user satisfaction by solving user complaints.

Currently, the library enter / exit system does not address issues of duplicate enter and non-returned seat. If a person enters the library and the other get ID card over the fence, the card can open enter machine. Because the exit system can leave without ID card, user who reserved a seat leaves, machine cannot identify it.

Since there isn’t LendSeat system, users book their seats by putting their bag and leave reading room. In this case, the users who really want to use the seat will not have enough space to use it.

This System prevents duplication User ID, and induces effective reading room reservation by penalty and LendSeat.

1.5. User Summary

**User** - people who use library

**Administrator** - person who manage library

1.6. User Level Goal

**User** - Library access, seat reservation, seat return, seat extension

**Administrator** - Administrator authentication, registration of ID card, management of duplicate access ID, deletion of entrance record, opening/closing of reading room

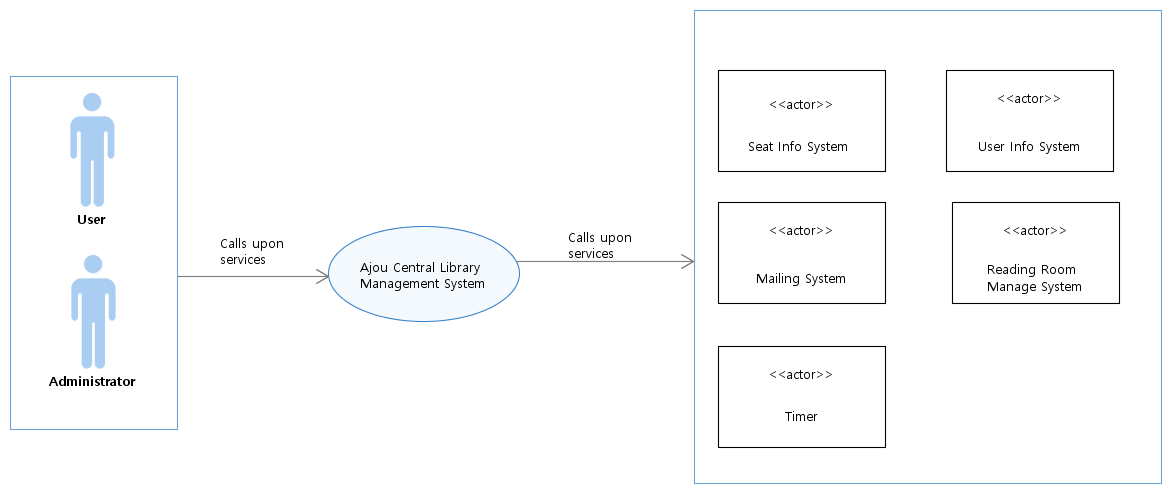
**In/Out system** - Record library access information, check user information

**Reservation system** - Offers seat data, Offers reservation/departure function

**ReadingRoomManageSystem** - Reserve, Return and Lend a seat.

1.7. Product overview & Product Perspective

The Seat reservation system will usually reside in libraries. they will be in close to the library network to extend reservation time, either inside or close outside. It will provide services to users, and collaborate with other systems, as indicated in Figure.



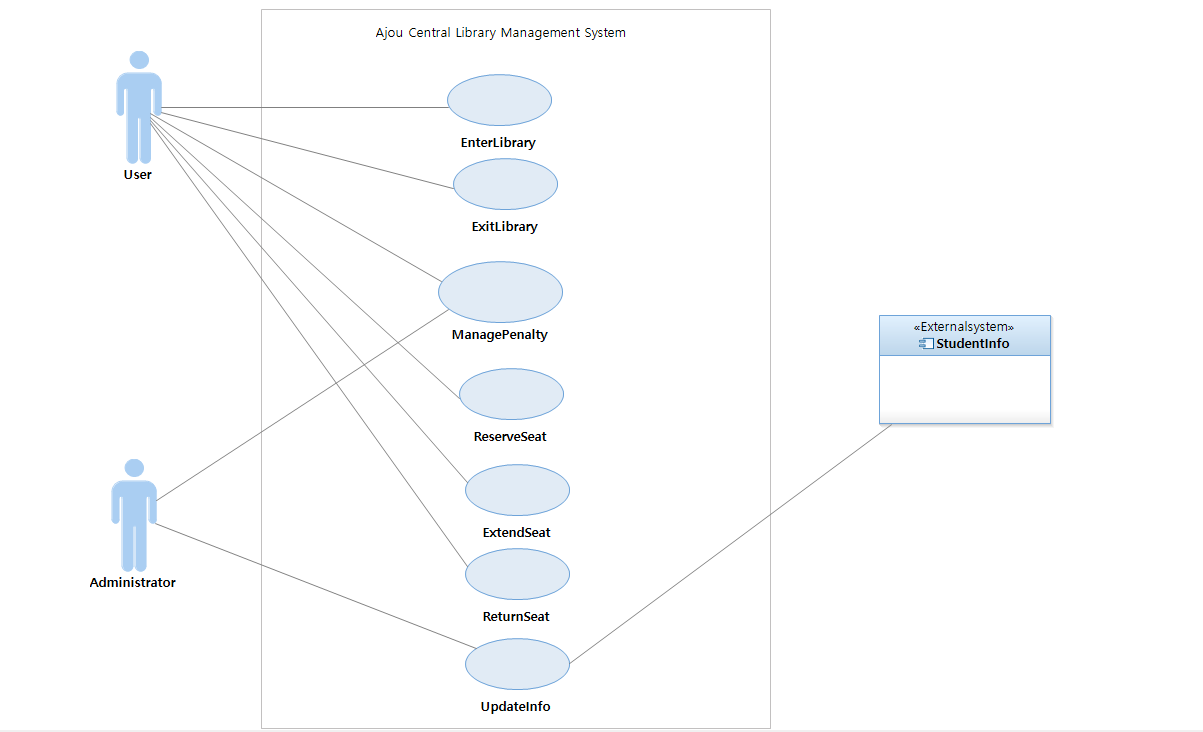
< Figure1. Context Diagram >

1.8. Summary of System Feature

* registration authorization
* identify the location of the student that reserved seat, either inside library or outside library
* automatic assignment of penalty
* prevention of duplicate access for student or unauthorized person

**2. Requirements**

**2.1. Use Case Diagram**



< Figure2. Use Case Diagram >

**2.2. Use Case Text**

**2.2.1. Functional Requirements**

* **Use case 1. EnterLibrary:** User puts his/her ID card on the Library Entry Machine.The the system checks the user information is registered. The system also checks the user is in library, checks the user does not have the penalty, and finds the user’s seat number currently reserved. Finally the system opens the Library Entry Machine.
* **Use case 2. ExitLibrary:** User puts his/her ID card on the Library Exit Machine. Then the system checks the user’s ID in the library. Also, the system finds the user’s seat number currently reserved, and checks the difference between the current time and the user’s seat expiration time is over 1 hour. The system modifies user’s seat expiration time to the ‘current time + 1 hour’ and removes information that user is inside the library. Finally the system opens the Library Entry Machine.
* **Use case 3. ManangePenalty:** User wants to watch his/her wrong penalty record. User sends a protest message to administrator. Then the administrator checks user’s penalty record. Administrator sends a detailed penalty reason to user. User submits an explanation why user was forced to receive a penalty. Administrator reads the explanation and determines whether it is valid. Administrator removes penalty of the user. Administrator informs user that penalty has been removed. User can now enter the library.
* **Use case 4. ReserveSeat:** User puts his/her ID card on the Reading Room Machine and system shows the current seat table(valid seats). User selects the seat he/she wants. System puts the his/her ID number and seat start time to the seat information. System sets up the user’s end time to ‘current time + 4 hours’ and starts the timer to countdown for 4 hours. Finally the system sends message that the reservation is completed.
* **Use case 5. ExtendSeat:** User puts his/her ID card on the Reading Room Machine. Then the system shows the option; ‘extend’ and ‘return’. User selects the option ‘extend’ on the machine. The system checks the user’s seat reservation time is extendable and extends the user’s seat end time as ‘end time + 4 hours’. Finally the system sends the changed end time to the user.
* **Use case 6. ReturnSeat:** User puts his/her ID card on the Reading Room Machine. Then the system shows the option ‘return’, ‘extend’. User selects the option ‘return’ on the machine.The system deletes the user’s ID number, seat start time, seat expiration time from the seat information. Finally the system sends the messages that the return seat is successed.
* **Use case 7. UpdateInfo:** Administrator clicks the button of ‘update’ of system. Then system requests the current student information to the college. After that, system updates its information by comparing its own user information with the student information received from college. Finally it shows the alert window for administrator to notify that the update is finished.

**2.2.2. Use Case Model**

* **Use case 1. EnterLibrary**
* Use Case Name: EnterLibrary
* Scope: Library Entry machine
* Level: user-goal
* Primary Actor: User
* Stakeholders and Interests:
* Administrator: Wants to allow the identified users only.
* User: Wants to enter library.
* Preconditions: The user does not enter the library.
* Postconditions: The only verified user enters the library.
* Main Success Scenario:

1. User puts his/her ID card on the Library Entry Machine.
2. The system checks the user information is registered.
3. The system checks the user is in library.
4. The system checks the user does not have the penalty.
5. The system finds the user’s seat number currently reserved.
6. The system opens the Library Entry Machine.

* Extensions(Alternative Flows):

2a. User is not registered.

1. The user requests the administrator to register.
2. The administrator registers the user information in the library.

3a. The user is in the library(duplicate entry).

1. The user receives a penalty.
2. A penalty is recorded in the user information.
3. The Library Entry Machine is not open.

4a. The user already has the penalty.

1. The Library Entry Machine is not open.

5a. There is user’s seat number.

1. The return time will be plus 4 the reservation time.

* Special Requirements:
* Mobile ID card also plays the same role with student ID card so it can be used instead of that card.
* The procedure of registering ID on data must be done through the administrator.
* The administrator can open the door arbitrarily.
* User authentication and door opening times should be within 1 seconds.
* Only one users at a time should place on the sensor their id card.
* The user must place on the sensor their id card for at least 0.5 seconds.
* The door is open for up to 3 seconds.
* The user put their id card on the sensor within 3 cm.
* Technology and Data variations List
* Most of the visitors of the library is student.
* Most of the non-registered students are freshmans, graduates or students on a leave of absence.
* Faculty members can also become users.
* Check the user’s information with the sensor.
* Frequency of Occurrence:
* This system should always be up and running because the library is open 24 hours a day, 7 days a week.
* **Use case 2. ExitLibrary**
* Use Case Name: ExitLibrary
* Scope: Library Exit Machine
* Level: user-goal
* Primary Actor: User
* Stakeholders and Interests
* User : Wants to leave the library.
* Administrator : Wants to let user returns his/her seat in the reading room before leave and no Library Exit Machine error.
* Preconditions: The user is inside the library.
* Postconditions: The user leaved the library.
* Main Success Scenario:

1. User puts his/her ID card on the Library Exit Machine.
2. The system checks the user’s ID in the library.
3. The system finds the user’s seat number currently reserved.
4. The system checks the difference between the current time and the user’s seat expiration time is over 1 hour.
5. The system modifies user’s seat expiration time to the ‘current time + 1 hour’.
6. The system removes information that user is inside the library.
7. The system opens the Library Entry Machine.

* Extension:.

2a. The user ID does not exist in the library.

1. System sends a warning message to user and administrator.

3a. There is no user’s seat number currently reserved.

1. System removes information that user is inside the library.
2. System opens the Entry Machine.

4a. User’s seat expiration time is not over 1 hour.

1. System removes information that user is inside the library.

2. System opens the Entry Machine.

* Special Requirements:
* Mobile ID card also plays the same role with the user ID card, so both of them can be used in the Exit Machine.
* The administrator can open the door of the Exit Machine arbitrarily.
* Only one user should place his/her ID card on the sensor of the Exit Machine at a time.
* The user must place his/her ID card on the sensor for at least 0.5 seconds.
* The user put their id card on the sensor within 1 cm.
* The checking procedure and the door opening times should be within 1 seconds.
* The door is open for up to 3 seconds.
* Technology and Data variations List:
* All of the users who use Exit Machine should be inside the library who is already registered in the library and used Entry Machine.
* The sensor of the Exit Machine can read the user information from the ID card.
* Seat information from the reading room is about each state of the seats whether they are reserved or not and whose the present owner is.
* Frequency of Occurrence:
* This system should always be up and running because the library is open 24 hours a day, 7 days a week.
* **Use case 3. ManagePenalty**
* Use Case Name: ManagePenalty
* Scope: System
* Level: user-goal
* Primary Actor: User
* Stakeholders and Interests:
* User: Wants to complain about his/her penalty.
* Administrator: Wants to modify the user’s wrong penalty information.
* Preconditions: User is identified and authenticated.
* Postconditions: User’s penalty information is changed. Library entry of user becomes possible.
* Main Success Scenario:

1. User checks his/her wrong penalty record.
2. User sends a protest message to system.
3. System sends a detailed penalty reason to user.
4. User submits an explanation to system why user was forced to receive a penalty.
5. System sends an explanation to administrator
6. Administrator reads the explanation and determines whether it is valid.
7. Administrator revises penalty of the user.
8. Administrator informs user that penalty has been revised
9. User can now enter the library.

* Extensions(Alternative Flows):

7a. Administrator confirm penalty due to the system error

1. Administrator removes penalty of the user.

2. Administrator informs user that penalty has been removed

3. User can now enter the library.

7b. Administrator judges that user’s explanation is not valid

1. Administrator informs user that the explanation is not valid

2. User cannot enter the library.

* Special Requirements:
* Touch screen UI on a large flat panel monitor. Text must be visible from 0.5 meter.
* Mobile ID card also plays the same role with the user ID card, so both of them can be used in ManageSeat machine
* User can attach an explanation on Touch screen UI.
* Administrator can access the system and clear user’s penalty.
* Rules for reasonable reason are established.
* Accuracy of the system is 99%.
* If administrator is in the spot, administrator can immediately deal with the user’s protest.
* Technology and Data variations List

1a. User identifier entered by Unique number.

4a. There are a variety of explanation description

* Frequency of Occurrence: This system should always be up and running because the reading room in the library is open 24 hours a day, 7 days a week.
* **Use case 4. ReserveSeat**
* Use Case Name: ReserveSeat
* Scope: Reading Room Machine
* Level: user-goal
* Primary Actor: User
* Stakeholders and Interests:
* User: Wants to reserve his/her seat in the reading room.
* Administrator: Wants no error when the user uses the seat.
* Preconditions: The user does not have his/her seat.
* Postconditions: The user made a reservation for the seat.
* Main Success Scenario:

1. User puts his/her ID card on the Reading Room Machine.
2. System shows the current seat table(valid seats).
3. User selects the seat he/she wants.
4. System puts the his/her ID number and seat start time to the seat information.
5. System sets up the user’s end time to ‘current time + 4 hours’ and starts the timer to countdown for 4 hours.
6. The system sends message that the reservation is completed.

* Extensions(Alternative Flows):

3a. The user selects already reserved seat.

1. The system shows error message ‘It’s already reserved seat’.
2. The system shows the current seat information again for the user to select the other seat.

5a. If it’s on a timer,

1. The user’s seat reservation is held until the timer is done.

5b. If the timer is done,

1. System checks the ‘seat expiration time - seat start time’ is same with 4 hours.

1a. When the ‘seat expiration time - seat start time’ is under the 4 hours,

1. The system adds penalty to the user.
2. The system deletes the user’s ID number, seat start time, seat expiration time from the seat information.
3. The system deletes the user’s ID number, seat start time, seat expiration time from the seat information.

* Special Requirements:
* Mobile ID card also plays the same role with student ID card so it can be used instead of that card.
* The rule to receive the penalty is that when the countdown expires when the expiration time - the reservation time is less than 4 hours.
* User authentication should be within 1 seconds.
* The user must place on the sensor their ID card for at least 0.5 seconds.
* The user put their ID card on the sensor within 3 cm.
* Touch screen UI of the ManageSeat machine on a large flat panel monitor.
* Accuracy of the system is 99%.
* If administrator is in the spot, administrator can immediately deal with the user’s protest.
* Technology and Data variations List
* Sensor of the Seat Machine can read the user information from the ID card.
* User’s seat information is stored Seat Database.
* Frequency of Occurrence: Reading room is used 24 hours a day.
* **Use case 5. ExtendSeat**
* Use Case Name: ExtendSeat
* Scope: Reading Room Machine
* Level: user-goal
* Primary Actor: User
* Stakeholders and Interests:
* User: Wants to extend the return time of his/her seat.
* Administrator: Wants no error when the user extends his/her seat.
* Preconditions: The user has his/her seat in the reading room.
* Postconditions: The user’s seat expiration time is extended.
* Main Success Scenario:

1. User puts his/her ID card on the Reading Room Machine.
2. The system shows the option; ‘extend’ and ‘return’.
3. User selects the option ‘extend’ on the machine.
4. The system checks the user’s seat reservation time is extendable.
5. The system extends the user’s seat end time as ‘end time + 4 hours’.
6. The system sends the changed end time to the user.

* Extensions(Alternative Flows):

4a. It is not an extensible time.

* The system sends message that it is not possible to extend the seat.
* Special Requirements:
* Mobile ID card also plays the same role with student ID card so it can be used instead of that card.
* The seat extendable time is from 2 hours before the expiration time.
* User authentication should be within 1 seconds.
* The user must place on the sensor their id card for at least 0.5 seconds.
* The user put their id card on the sensor within 3 cm.
* Touch screen UI on a large flat panel monitor.
* Accuracy of the system is 99%.
* If administrator is in the spot, administrator can immediately deal with the user’s protest.
* Technology and Data variations List
* Sensor of the Seat Machine can read the user information from the ID card.
* User’s seat information is stored Seat Database.
* Frequency of Occurrence: Reading room is used 24 hours a day.
* **Use case 6. ReturnSeat**
* Use Case Name: ReturnSeat
* Scope: Reading Room Machine
* Level: user-goal
* Primary Actor: User
* Stakeholders and Interests:
* User : Wants to return his/her seat in the reading room.
* Administrator : Wants no error when the user return his/her seat.
* Preconditions: The user has his/her seat.
* Postconditions: The user does not have his/her seat.
* Main Success Scenario:

1. User puts his/her ID card on the Reading Room Machine.
2. The system shows the option ‘return’, ‘extend’.
3. User selects the option ‘return’ on the machine.
4. The system deletes the user’s ID number, seat start time, seat expiration time from the seat information.
5. The system sends the messages that the return seat is successed.

* Extensions(Alternative Flows):
* N/A
* Special Requirements:
* Mobile ID card also plays the same role with student ID card so it can be used instead of that card.
* User authentication should be within 1 seconds.
* The user must place on the sensor their id card for at least 0.5 seconds.
* The user put their id card on the sensor within 3 cm.
* Touch screen UI on a large flat panel monitor.
* Accuracy of the system is 99%.
* If administrator is in the spot, administrator can immediately deal with the user’s protest.
* Technology and Data variations List
* Sensor of the Seat Machine can read the user information from the ID card.
* User’s seat information is stored Seat Database.
* Frequency of Occurrence: Reading room is used 24 hours a day.
* **Use case 7. UpdateInfo**
* Use Case Name: UpdateInfo
* Scope: System
* Level: user-goal
* Primary Actor: Administrator
* Stakeholders and Interests:
* Administrator: Wants to update the student information in system.
* User: Wants simple registration using library.
* Preconditions: The user information of system doesn’t have current student information of college.
* Postconditions: The user information of system has current student information of college.
* Main Success Scenario:

1. Administrator pushes the button of ‘update’ of system.
2. System requests the current student information to the college.
3. College sends it student information to the system.
4. System compares its own user information with the student information received from college and add the content of student information to the user information of system.
5. System notifies to the administrator that the update is finished.

* Extensions(Alternative Flows):

3a. College does not respond and request failed:

1. System notifies to the administrator that the request is failed.

4a. The content of student information already exists in the user information of system:

1. Do nothing. Just pass and go to the next content to compare.

* Special Requirements:
* Existing in user information of system means that it is possible to use library.
* Updating information is same job with registering the whole student in the information to the library.
* Technology and Data variations List
* The student information from college contains only about students.
* The user information of system contains students, facility members and library staffs who are the all of the user of library.
* Frequency of Occurrence:
* It occurs at least once every 6 months which means once per semester.
* It is used right before a semester because there is big change in the student information of college before the semester begins.

**2.2.3. Non-Functional Requirements**

**Introduction**

This is for documenting all of non-functional requirements of the use cases to supplement the use case models.

**Functionality**

* Logging and Error Handling: Log all of errors in the storage. Only the administrator can handle and access to those errors.

**Human Factors**

* To communicate or interact about the managing penalty, the user and administrator may uses touch screen UI on a flat panel monitor. So text must be visible from 0.5 meter.
* Use case 1 and 2 occur at the entrance of library. For the comfortable library and effective usage of reading room, the speed and error-free processing are very important.
* When the user gets penalty, the entry machine should give signal to the user to alert his/her penalty. Its shape should be sound not graphic. It’s because the penalty is given when user enters the library, that means, user interface is a entry machine, not a screen.

**Reliability**

* Recoverability: If there is error in the entry/exit machine, the administrator takes over the authority to control the entrance of the library from the machines. If the system fails to access the external system, it notifies immediately to the administrator through the graphic as a alert window.

**Performance**

* As mentioned at the human factors, the speed is important. So our goal is taking less than 1 second as an average time between placing ID card and opening the door.

**Supportability**

* Adaptability: When new user wants to use the library, he/she only needs registration of ID card to be allowed in the system, because the registration makes the user information accept the new one.

**Interfaces**

* Hardware: Entry Machine and Exit Machine are needed for control the entrance of the library and check the penalty. Reading Room Machine is also needed to reserve, extend, and return the seat in the reading room for users.
* Software: Only student information is from college. Library possesses its own user information and seat information library.

**Application-specific Domain (Business) Rules**

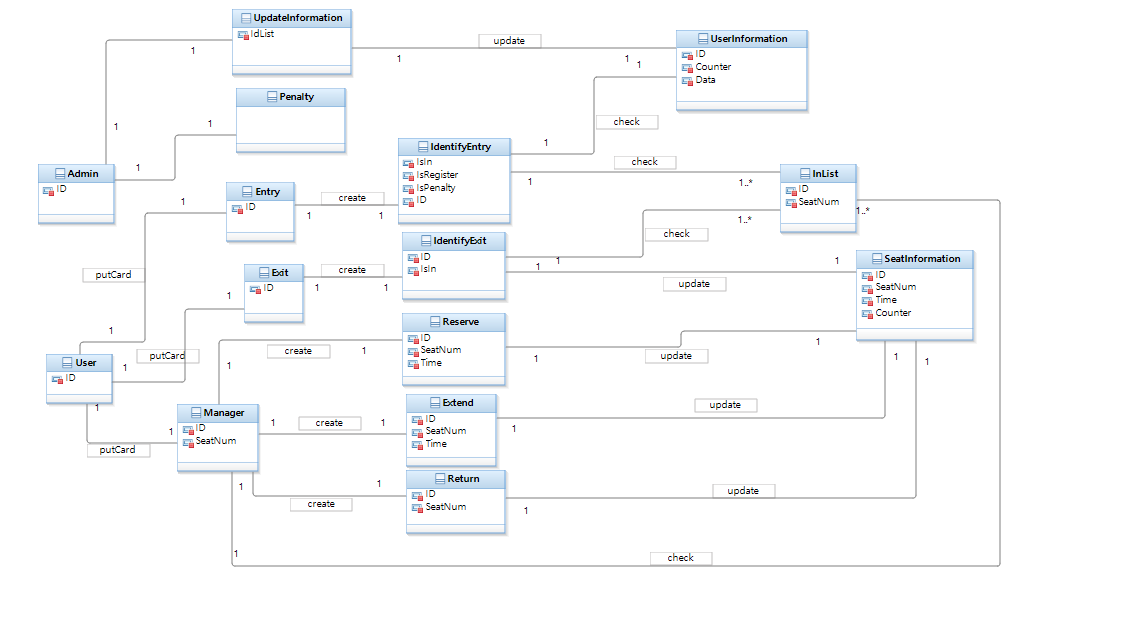
* Rule 1. Penalty rules: Different weights are applied depending on the number of penalties. If user has 1 penalty, he/she cannot use library service for one week. If user has 2 penalties, he/she cannot use library service for one month. If user has 3 penalties, he/she cannot use library service for whole semester. This rule has low changeability because the library adopts only this one rule as a penalty rules.

**3. Analysis Modeling**

**3.1. Domain Model Diagram**

We analyzed these 7 use cases in this Final document; use case 1 named EnterLibrary, use case 2 named ExitLibrary, use case 3 named ManagePenalty, use case 4 named ReserveSeat, use case 5 named ExtendSeat, use case 6 named ReturnSeat and use case 7 named UpdatedInfo.

Use case 1 has conceptual classes of User, Entry, IndentifyEntry, UserInfo, and InList. Use case 2 has conceptual classes of User, Exit, IdentifyExit, SeatInfo and InList. Use case 3 has conceptual classes of Admin, User, Penalty, IdentifyPenalty and UserInfo. Use Case 4 has conceptual classes of User, Manager, Reserve, InList and SeatInfo. Use Case 5 has conceptual classes of User, Manager, Extend, InList and SeatInfo. Use Case 6 has conceptual classes of User, Manager, Return, InList and SeatInfo. Use Case 7 has conceptual classes of Admin and Update. Thus we set the scope of the domain model as shown below.

****

< Figure3. Domain Model Diagram >

**3.2. System Sequence Diagram**

The system sequence diagram is a visual summary of the individual use cases.

In use case 1, the user puts ID card on the Library Entry Machine, the system. Inside the system, it checks 3 kinds of the user's information from the ID and then gives the value of True of False to open the Entry machine for the request process.

In use case 2, the user puts ID card on the Library Exit Machine, the system. After the system checks about the user's seat information from the ID, it gives the value of True or False to open the Exit Machine.

In use case 3, the user requests information related to the penalty of the user. The user checks the penalty information and then, sends a message to the system to inform the Administrator. The administrator who receives the message modifies the penalty information of the user according to the situation and informs the user of the user's penalty information.

In use case 4, the user put ID card on the Reading Room Machine, the system. Then the system sends the current seat table of the reading room to the user to let user choose the valid seat. After the user chooses the specific seat and sends the seat number to the system, then the system reserves that seat for the ID and gives value of true meaning the job is completed.

In use case 5, the user puts ID card on the Reading Room Machine, the system. The system sends message to the user to select which job you want between 'extend' and 'exit'. After the user chooses the 'extend', the system extend the end time in the user's seat information and sends that upgraded end time to the user.

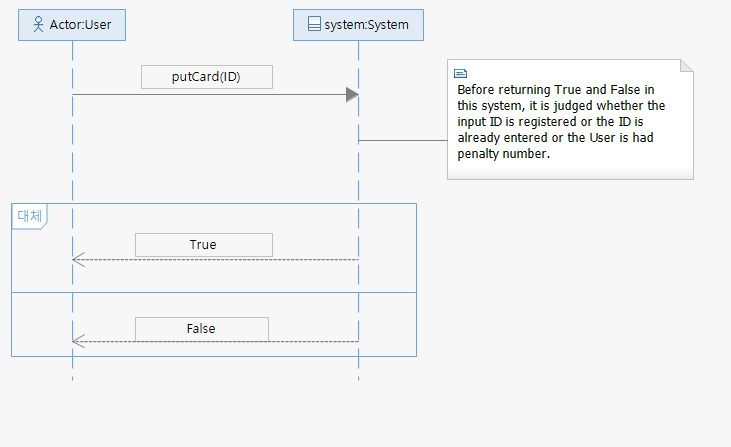
In use case 6, the user puts ID card on the Reading Room Machine, the system. The system sends message to the user to select which job you want between 'extend' and 'exit'. After the user selects the 'return', the system return the seat of the user and send message to the user meaning the job is successfully done.

In use case 7, the administrator orders the system to update the user information in the library. Then the system requests the current student information to the college and updates the library's user information. After that, it sends the value of true meaning that the job is completed.

**3.2.1. Use case 1. EnterLibrary**

**System Sequence Diagram**

The user puts ID card on the Library Entry Machine, the system. Inside the system, it checks 3 kinds of the user's information from the ID; whether the user is registered, doesn’t exit inside library(duplication), and has penalty. Then the system gives the value of True or False to open the Entry machine for the request process.

****

< Figure 4. Use Case 1. SSD >

**System Operation List**

* PutCard(ID): User puts his/her ID card on the library entry machine, the system, giving ID number to the system. Then, the system responses as True or False depending on the ID.

**Operation Contracts**

Operation: PutCard(ID)

Cross References: Use Case 1. EnterLibrary

Preconditions: The ID number doesn’t exist in the InList.

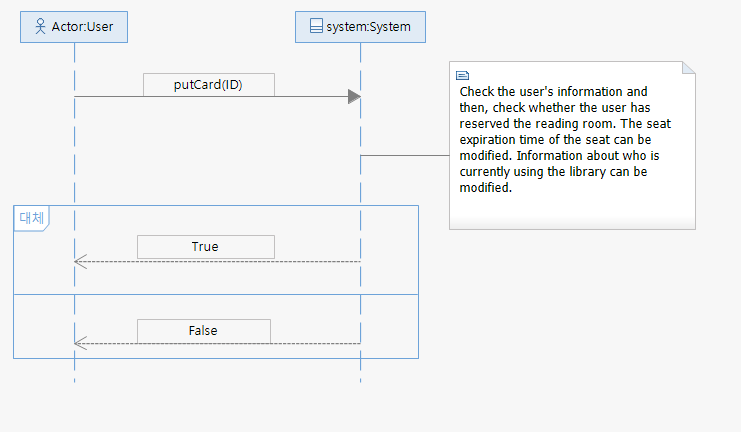
Postconditions:

* An IdentifyEntry instance was created.
* An ID attribute in the IdentifyEntry instance were initialized by given ID.

**3.2.2. Use case 2. ExitLibrary**

**System Sequence Diagram**

The user puts ID card on the Library Exit Machine, the system. After the system checks about the user's seat information from the ID, depending on the result, it modifies the end time of seat reservation. Then it gives the value of True or False to open the Exit Machine.

****

< Figure 5. Use Case 2. SSD >

**System Operation List**

* PutCard(ID): User puts his/her ID card on the Library Exit Machine, the system, giving ID number to the system. Then it responses as True or False depending on the ID.

**Operation Contracts**

Operation: PutCard(ID)

Cross References: Use Case 2. ExitLibrary

Preconditions: The ID number exists in the InList.

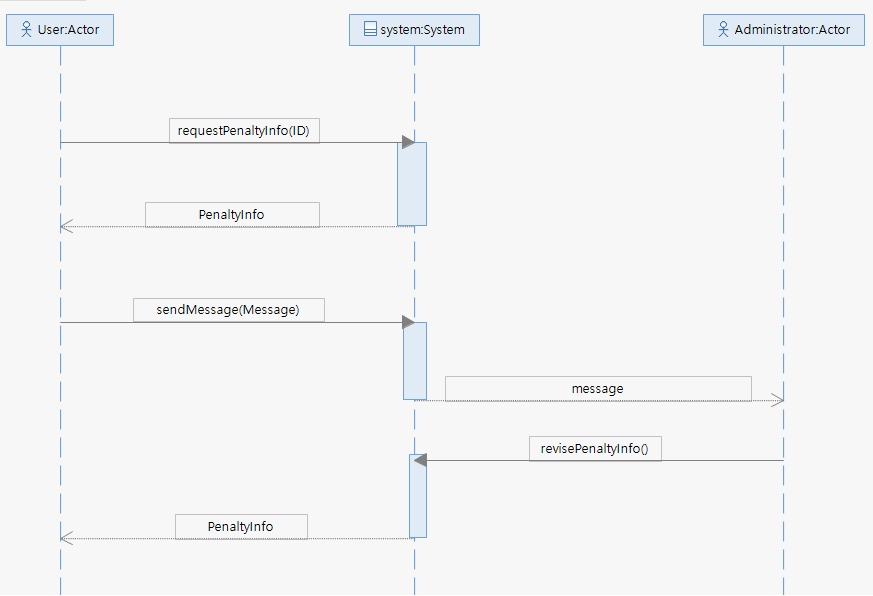
Postconditions:

* An IdentifyExit instance was created.
* An ID attribute in IdentifyExit instance was initialized by given ID.
* If the differences between current time and the end time of user’s seat was over 1 hour, the end time in the seat information was modified to ‘current time + 1 hour’.
* The value of ID in InList was deleted.

**3.2.3. Use case 3. ManagePenalty**

**System Sequence Diagram**

The user requests information related to the penalty of the user. The user checks the penalty information and then, sends a message to the system to inform the Administrator. The administrator who receives the message modifies the penalty information of the user according to the situation and informs the user of the user's penalty information.

****

< Figure 6. Use Case 3. SSD >

**System Operation List**

* RequestInfo: User checks his/her penalty through system and system gives answer about the penalty information.
* SendProtestMessage: To show complaint about his/her penalty information, user sends protest message to system, and then system deliver detail penalty to the user as a response.
* SubmitExplanation: User explains about the context received and upload (send mail) the explanation on the system, and then it is sended to administrator
* RevisePenalty: Administrator judges the message whether to remove the penalty and if he/she decides to remove, he/she requests to remove penalty through system. System removes the penalty and inform this completion to the user.

**Operation Contracts**

Operation: RequestInfo

Cross References: Use Case 3. ManagePenalty

Preconditions: Users can not enter the library

Postconditions:

* User’s request was stored in the system

Operation: SendProtestMessage

Cross References: Use Case 3. ManagePenalty

Preconditions: There is no mail in the system’s mailbox for the user.

Postconditions:

* User’s mail was stored in the system

Operation: SubmitExplanation

Cross References: Use Case 3. ManagePenalty

Preconditions: There is no explanation mail in the system

Postconditions:

* User’s explanation was stored in the system
* User’s explanation was stored in the administrator’s mailbox

Operation: RevisePenalty

Cross References: Use Case 3. ManagePenalty

Preconditions:

* There is no change of the penalty

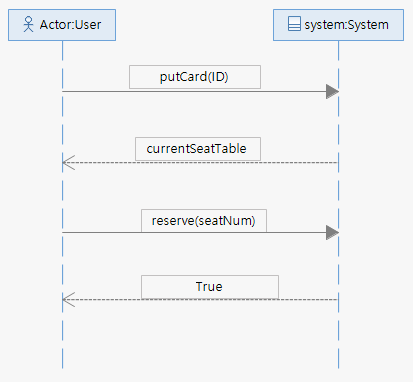
Postconditions:

* The penalty was revised
* User could enter the library

**3.2.4. Use case 4. ReserveSeat**

**System Sequence Diagram**

The user put ID card on the Reading Room Machine, the system. Then the system checks the ID does not have his/her own seat, so sends the current seat table of the reading room to the user to inform the valid seats. After the user chooses the specific seat and sends the seat number to the system, then the system reserves that seat for the ID and gives value of true meaning the job is completed.



< Figure 7. Use Case 4. SSD >

**System Operation List**

* putCard(ID): User puts his/her ID card on the Reading Room Machine, the system, giving ID number to the system. Then it gives current seat table to show the valid seats to the user.
* reserve(seatNum): User gives seat number to the system to reserve the seat. System uploads the seat number to the seat information and send the value of true to the users meaning that the reservation is done.

**Operation Contracts**

Operation: putCard(ID)

Cross References: Use Case 4. ReserveSeat

Preconditions: System, the Manager instance, doesn’t get the user ID number, so the ID number is 0.

Postconditions:

* An ID attribute of Manager instance was initialized by given ID.
* An Reserve instance was created. After that, the ID number changed to 0.
* An ID attribute in Reserve instance was initialized by given ID.
* An seatTable attribute was initialized from SeatInfo instance.

Operation: reserve(seatNum)

Cross References: Use Case 4. ReserveSeat

Preconditions: Seat number of user does not exist in the seat information.

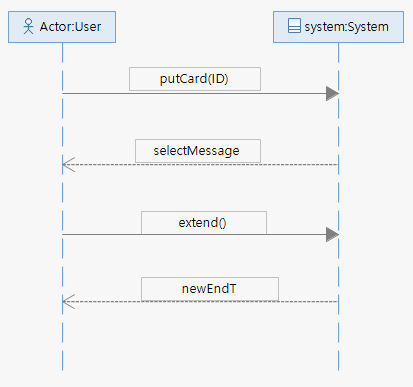
Postconditions:

* An seatNum attribute of Reserve instance was initialized by given seatNum.
* An seatNum attribute of InList instance was initialized from Reserve instance.
* ID, seatNum, StartT, EndT attribute of SeatInfo instance was initialized from Reserve instance.
* User got true value telling that the reservation is done.

**3.2.5. Use case 5. ExtendSeat**

**System Sequence Diagram**

The user puts ID card on the Reading Room Machine, the system. The system sends message to the user to select which job you want between 'extend' and 'exit'. After the user chooses the 'extend', the system extends the end time in the user's seat information and sends that changed end time to the user.

****

< Figure 8. Use Case 5. SSD >

**System Operation List**

* putCard(ID): User puts his/her ID card on the Reading Room Machine, the system, giving ID number to the system. Then it sends the message to user let him/her choose what to do between ‘extend’ and ‘return’.
* extend(): User chooses the ‘extend’, so orders extend to system. The the system extend the end time of the seat the user reserved and send the newEndT to user as a response.

**Operation Contracts**

Operation: putCard(ID)

Cross References: Use Case 5. ExtendSeat

Preconditions: User doesn’t get any message from the system and the System, the Manager instance, doesn’t get the user ID number, so the ID number is 0.

Postconditions:

* An ID attribute of Manager instance was initialized by given ID.
* An seatNum attribute of Manager instance was initialized from InList.
* The user got message of selection.

Operation: extend()

Cross References: Use Case 5. ExtendSeat

Preconditions: User’s seat exists and the end time of reservation doesn’t changed. (The difference from current time is over 2 hours)

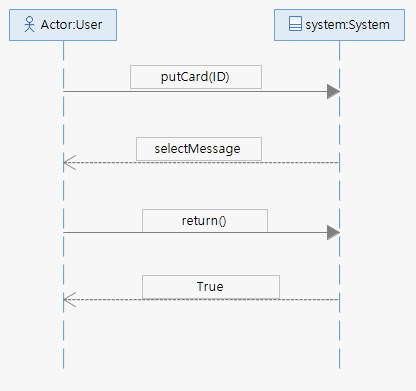
Postconditions:

* An Extend instance was created and the ID attribute of the Manager instance changed to 0.
* ID and seatNum attribute of Extend instance was initialized from Manager instance.
* An endT attribute of Extend instance was initialized from SeatInfo instance.
* An endT in seatTable attribute of SeatInfo instance was updated with ‘endT + 4 hours’.
* An newEndT attribute of Extend instance was initialized from SeatInfo instance.
* The user got the newEndT from system.

**3.2.6. Use case 6. ReturnSeat**

**System Sequence Diagram**

The user puts ID card on the Reading Room Machine, the system. The system sends message to the user to select which job you want between 'extend' and 'exit'. After the user selects the 'return', the system return the seat of the user and send message to the user meaning the job is successfully done.



< Figure 9. Use Case 6. SSD >

**System Operation List**

* putCard(ID): User puts his/her ID card on the Reading Room Machine, the system, giving ID number to the system. Then it sends the message to user let him/her choose what to do between ‘extend’ and ‘return’.
* return(): User chooses the ‘return’, so orders return to system. The the system return the user’s reserved seat by deleting the information and send the success message to the user as a response.

**Operation Contracts**

Operation: putCard(ID)

Cross References: Use Case 6. ReturnSeat

Preconditions: User doesn’t get any message from the system and the System, the Manager instance, doesn’t get the user ID number, so the ID number is 0.

Postconditions:

* An ID attribute of Manager instance was initialized by given ID.
* An seatNum attribute of Manager instance was initialized from InList.
* The user got message of selection.

Operation: return()

Cross References: Use Case 6. ReturnSeat

Preconditions: Seat number of user exists in the seat information.

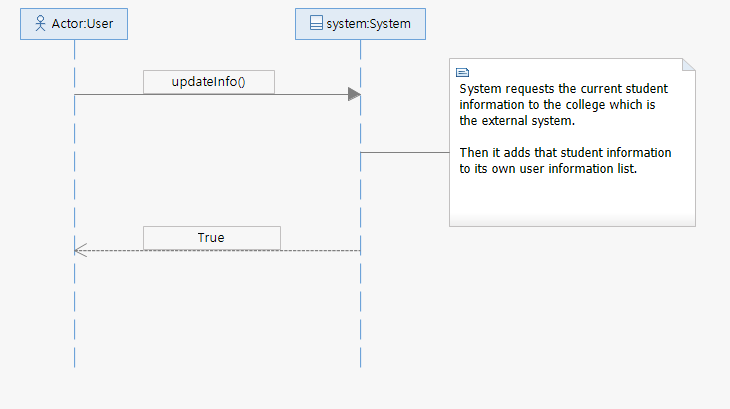
Postconditions:

* An Return instance was created and the ID attribute of the Manager instance changed to 0.
* ID and seatNum attribute of Return instance was initialized from Manager instance.
* An seatNum in inList attribute of InList instance was initialized to 0.
* An ID, StartT and EndT in seatTable attribute of Extend instance was initialized to 0.
* The user got changed success message from system.

**3.2.7. Use case 7. UpdateInfo**

**System Sequence Diagram**

The administrator orders the system to update the user information in the library. Then the system requests the current student information to the college and updates the library's user information. After that, it sends the value of true meaning that the job is completed.



< Figure 10. Use Case 7. SSD >

**System Operation List**

* UpdateInfo(): Administrator request an update of the user information to the system. Then the system requests current student information to the college which is an external system. When the job is finished, the system send a ‘true’ message to the administrator.

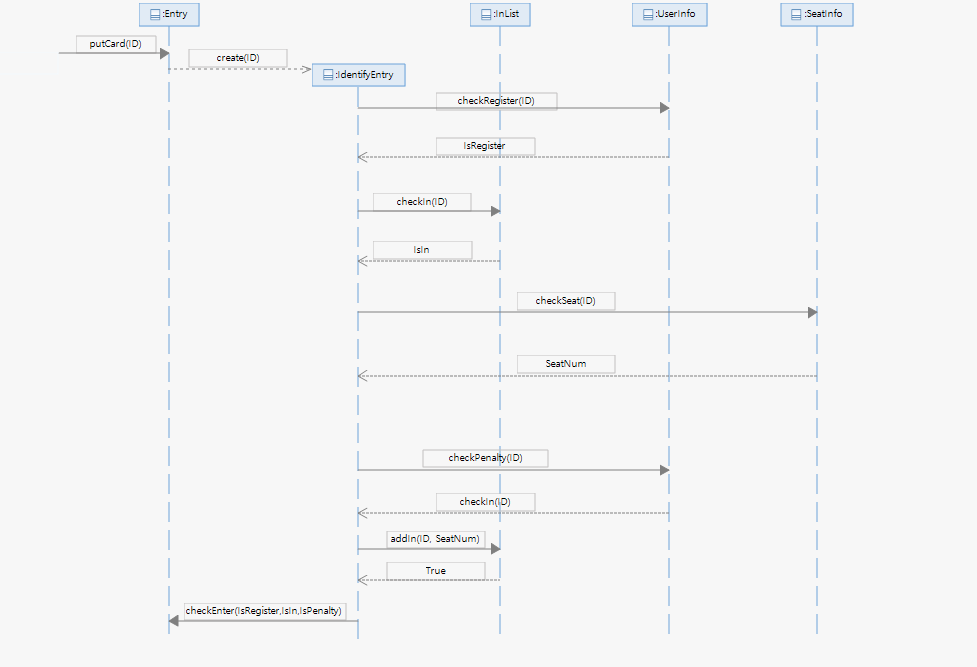
**Operation Contracts**

**4. Design Modeling**

In the Requirement sector, we mentioned 4 use cases; use case 1 named EnterLibrary, use case 2 named ExitLibrary, use case 3 named Manage Penalty and use case 4 named LendSeat. But in this elaboration iteration 1, we analyzed 3 use cases in the Use-Case Model and Analysis Model. In this sector, we set the scope of the design model as 2 use cases; use case 1 and use case 2. The design models are shown below.

**4.1. <Use case 1. EnterLibrary> Realization**

**4.1.1) Design Sequence Diagram: putCard(ID)**

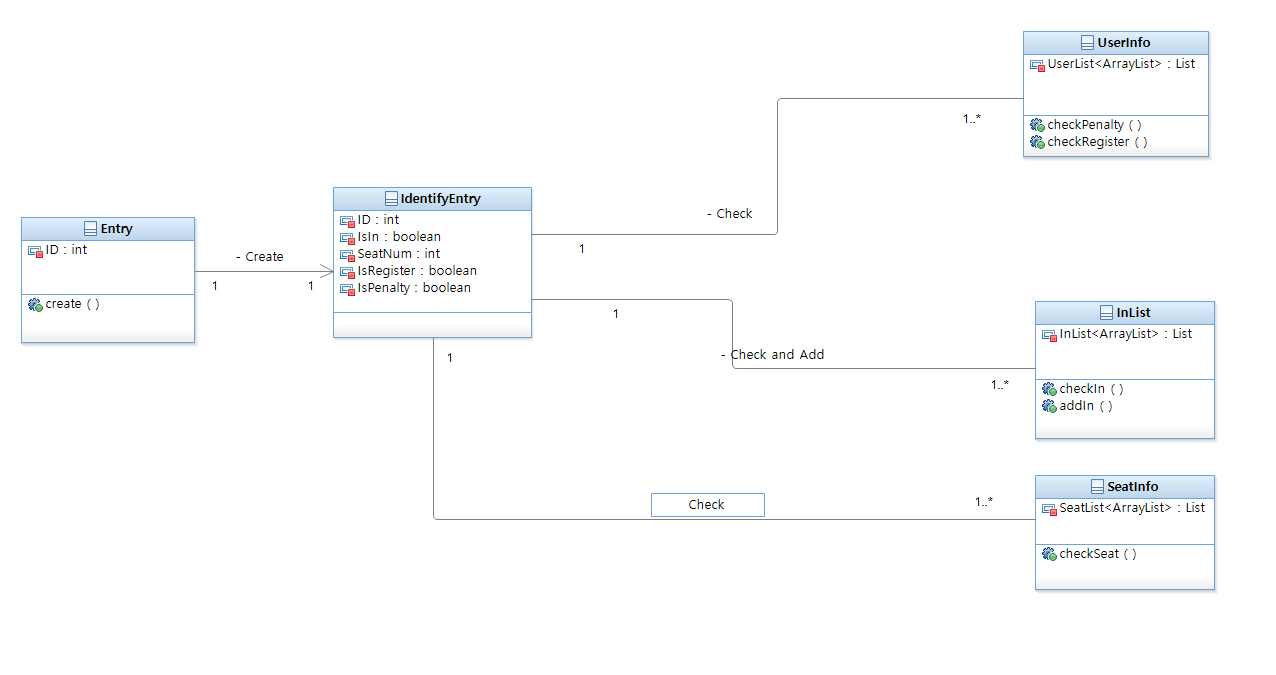
****

< Figure 11. Use Case 1. DSD >

**4.1.2) GRASP patterns**

|  |  |
| --- | --- |
| Pattern / Principle | Description |
| Creator | Class Entry creates an instance of class IdentifyEntry. |
| Controller | The IdentifyEntry is the first object beyond the UI layer that is responsible for receiving or handling a system operation message. And it has an attribute that indicates whether the user is a registered user or already enter user or received penalty user. |
| High Cohesion | The IdentifyEntry manages this information all at once and sends the results of the information to the Entry. |
| Low Coupling | None |

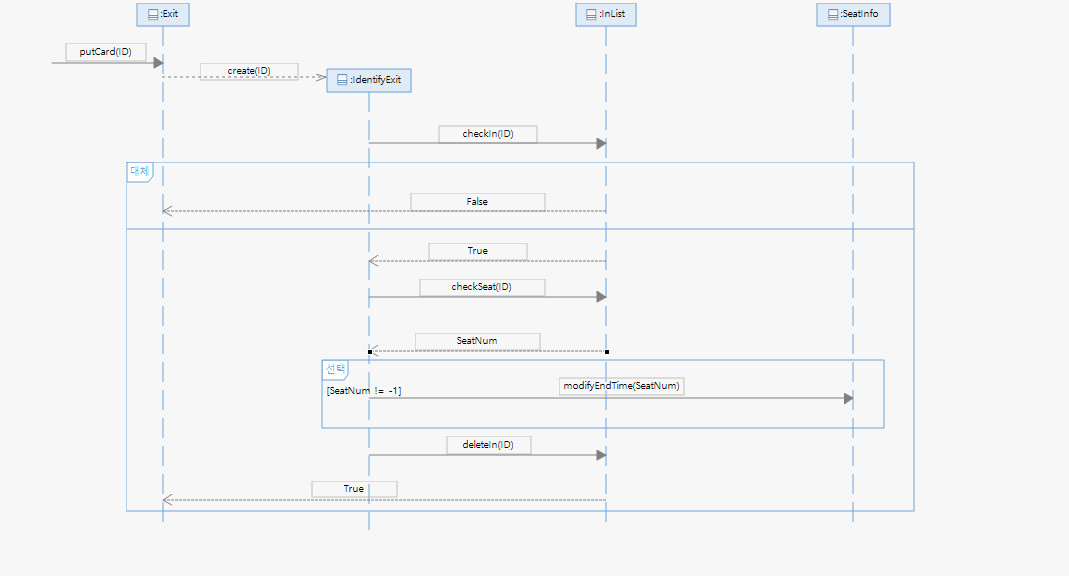
**Combined Design Class Diagram**



< Figure 12. Use Case 1. DCD >

**4.2. <Use case 2. ExitLibrary> Realization**

**Design Sequence Diagram: putCard(ID)**

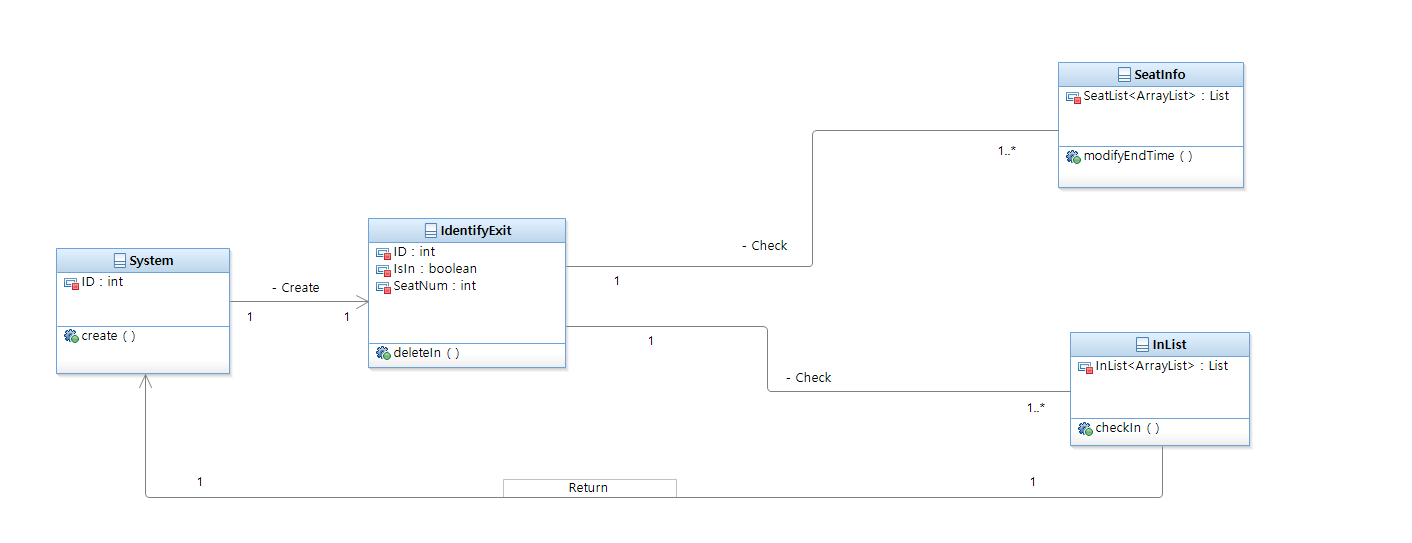


< Figure 13. Use Case 2. DSD >

**GRASP patterns**

|  |  |
| --- | --- |
| Pattern / Principle | Description |
| Creator | Class Exit creates an instance of class IdentifyExit. |
| Controller | The IdentifyExit is the first object beyond the UI layer that is responsible for receiving or handling a system operation message.  And it has an attribute that indicates whether the user is the user that was in library or reserved the seat. |
| High Cohesion | The IdentifyExit manages this information all at once and sends the results of the information to the Exit. |
| Low Coupling | None |

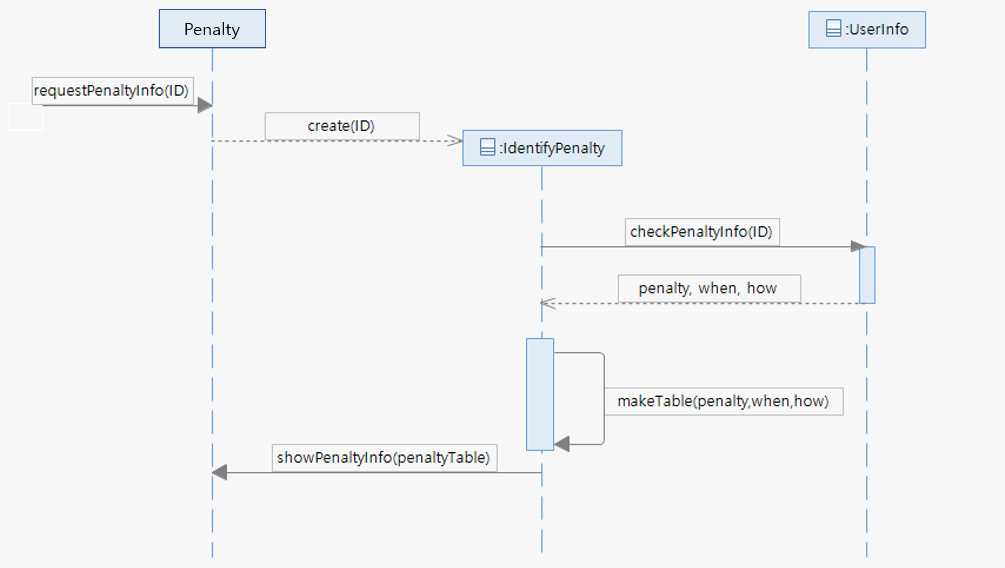
**Combined Design Class Diagram**

****

< Figure 14. Use Case 2. DCD >

**4.3. <Use case 3. ManagePenalty> Realization**

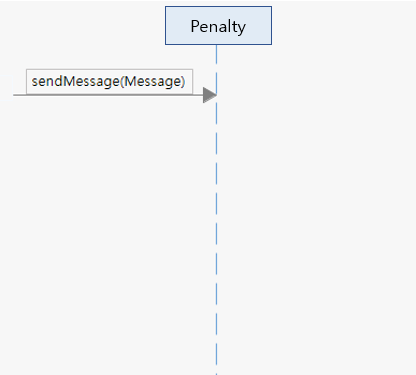
**Design Sequence Diagram: requestPenaltyInfo(ID)**

****

< Figure 15. Use Case 3. DSD >

|  |  |
| --- | --- |
| Pattern / Principle | Description |
| Creator | Class Penalty creates an instance of class IdentifyPenalty. |
| Controller | The IdentifyPenalty is the first object beyond the UI layer that is responsible for receiving or handing a system operation messages. And it has attributes that indicates penalty information. |
| High Cohesion | The IdenfifyPenalty manages this information all at once and sends the results of the information to the Penalty class. |
| Low Coupling | None |

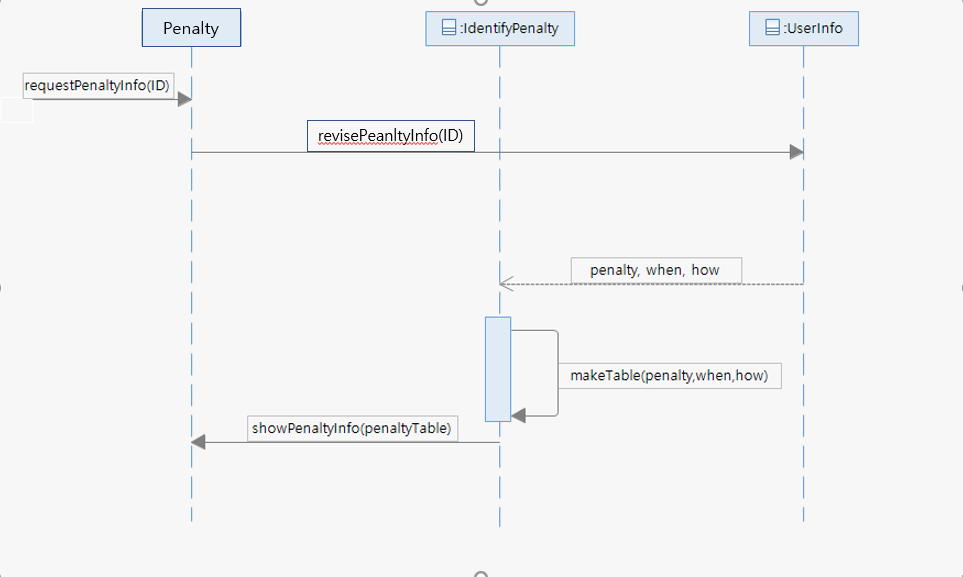
**Design Sequence Diagram: sendMessage(Message)**

****

< Figure 16. Use Case 3. DSD >

|  |  |
| --- | --- |
| Penalty / Principle | Description |
| Creator | None |
| Controller | None |
| High Cohesion | None |
| Low Coupling | None |

**Design Sequence Diagram: revisePenaltyInfo(ID)**

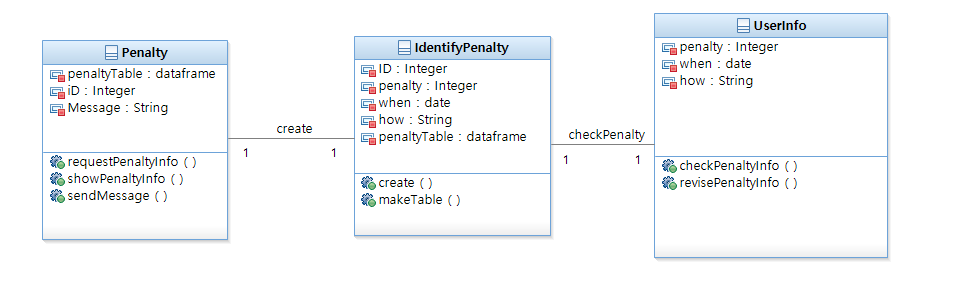
****

< Figure 17. Use Case 3. DSD >

**GRASP patterns**

|  |  |
| --- | --- |
| Penalty / Principle | Description |
| Creator | Class Penalty creates an instance of class IdentifyPenalty. |
| Controller | None |
| High Cohesion | The IdentifyPenalty receives informations for penalty Table and then sends the results of the information to the Penalty class. |
| Low Coupling | None |

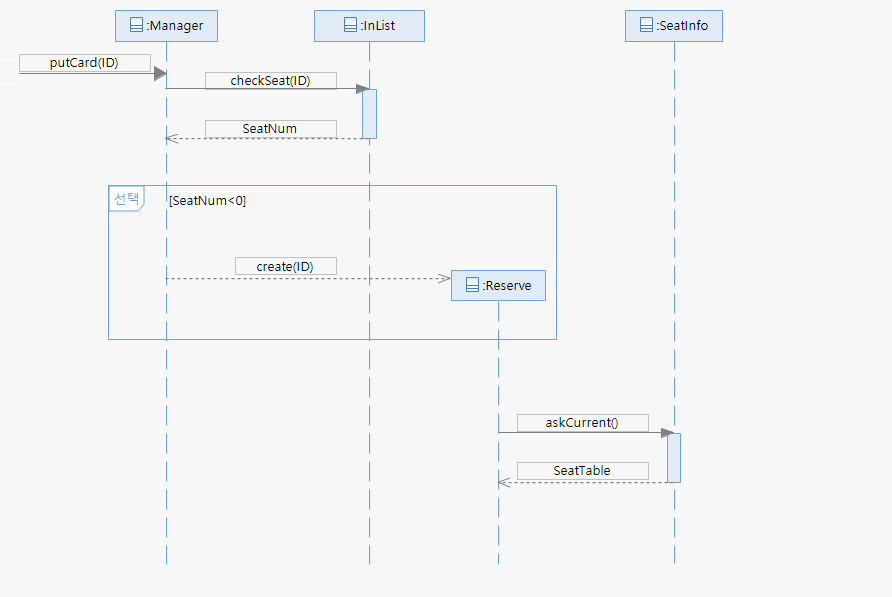
**Combined Design Class Diagram**



< Figure 18. Use Case 3. DCD >

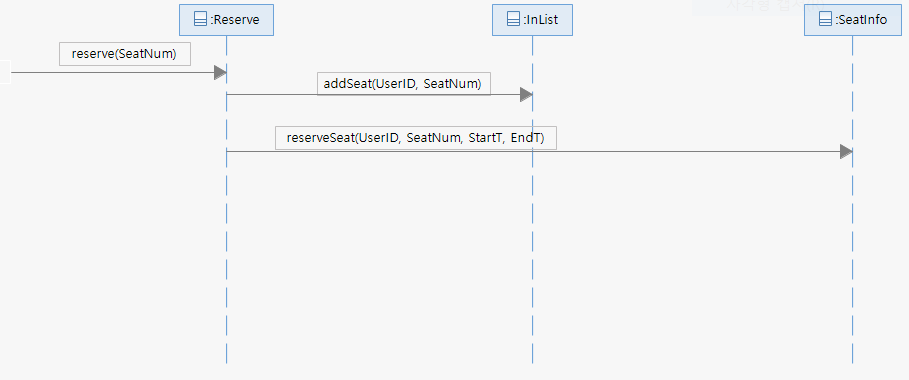
**4.4. <Use case 4.ReserveSeat> Realization**

**Design Sequence Diagram: PutCard(ID)**

****

< Figure 19. Use Case 4. DSD >

|  |  |
| --- | --- |
| Penalty / Principle | Description |
| Creator | Class Manager creates an instance of class Reserve. |
| Controller | None |
| High Cohesion | None |
| Low Coupling | None |

****

< Figure 20. Use Case 4. PutCard(User\_ID) DSD >

|  |  |
| --- | --- |
| Penalty / Principle | Description |
| Creator | None |
| Controller |  |
| High Cohesion |  |

**GRASP patterns**

Name : Creator

Problem : Who should be responsible for creating a new instance u.

Solution : Assign System the responsibility to create an instance of UserDescription.

* System has the initializing data for User\_ID of UserDesciription.

Name : Controller

Problem : What first object beyond the UI layer receives and coordinates ("controls") a

system operation?

Solution : The UserDescription is the first object beyond the UI layer that is responsible for receiving or handling a system operation message. And it has an attribute that indicates whether the user is a registered user or already enter user or received penalty user.

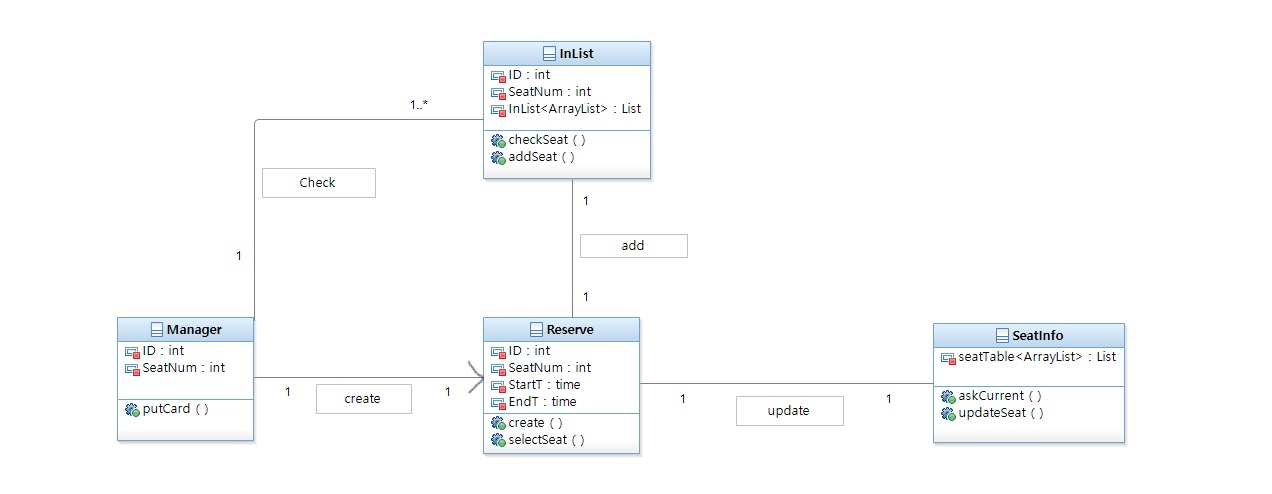
Name : High Cohesion

Problem : How to keep objects focused, understandable, and manageable, and as a side

effect, support Low Coupling?

Solution : The UserDescription manages this information all at once and sends the results of the information to the system.

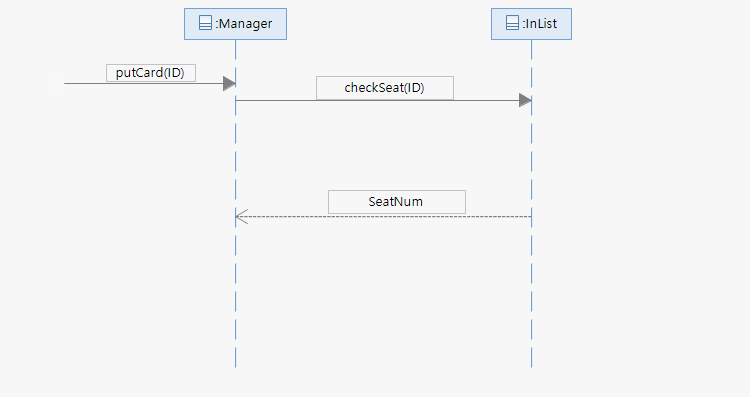
**Combined Design Class Diagram**

****

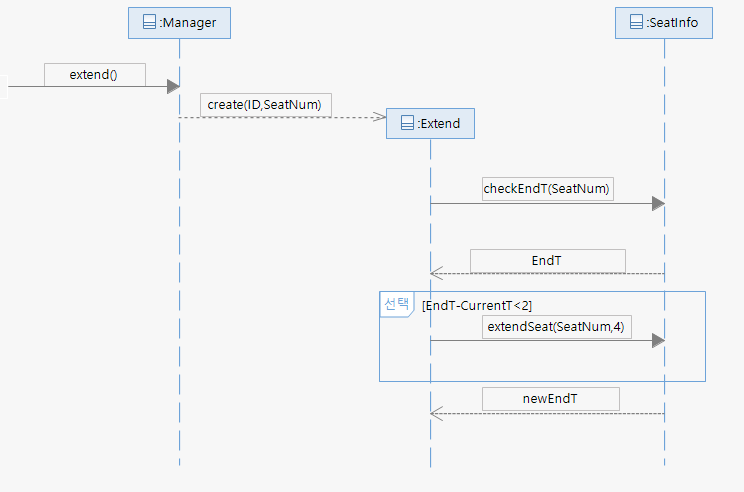
< Figure 21. Use Case 4. DCD >

**4.5. <Use case 5. ExtendSeat> Realization**

**Design Sequence Diagram**



< Figure 22. Use Case 5. DSD >



< Figure 23. Use Case 5. DSD >

**GRASP patterns**

Name: Controller

Problem: Who is responsible for ManageSeat system operation?

Solution:

* When a Union receives information from an IDcard, it requests information from each Object.
* First, ask UserInfoDescription if User is correct
* Then ask SeatInfoDescription if User has a seat.
* If you have a seat, change the reservation time for ReadingRoomDescription to one hour from now.

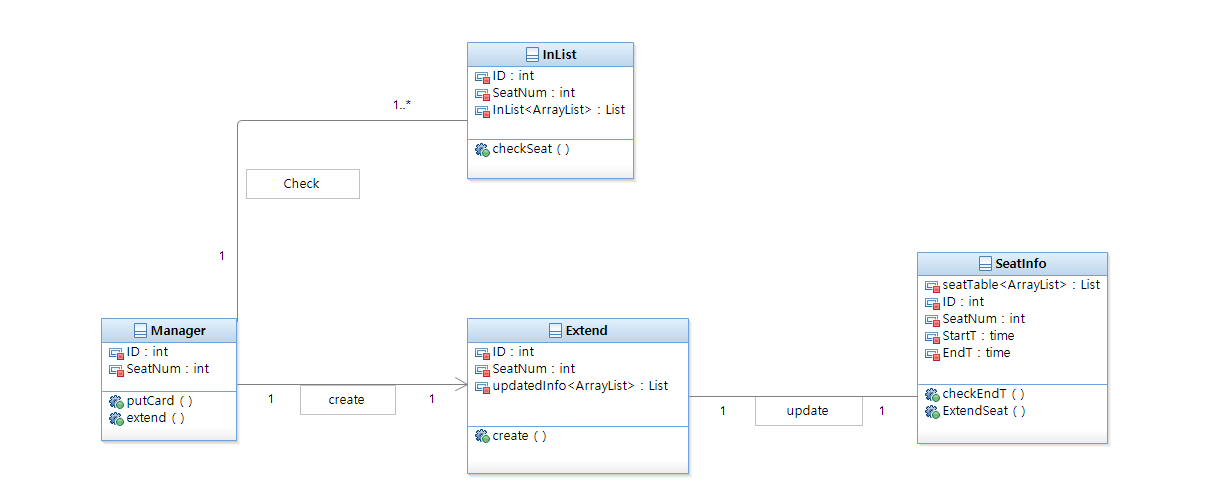
Name: Low Coupliing

Problem: How to reduce the impact of seat information

Solution:

* The user may or may not have returned the seat.
* So, finding a place to return it and changing the holding time to return it should be separated. By doing, our system can reduce the tasks of ReadingRoomSystem.

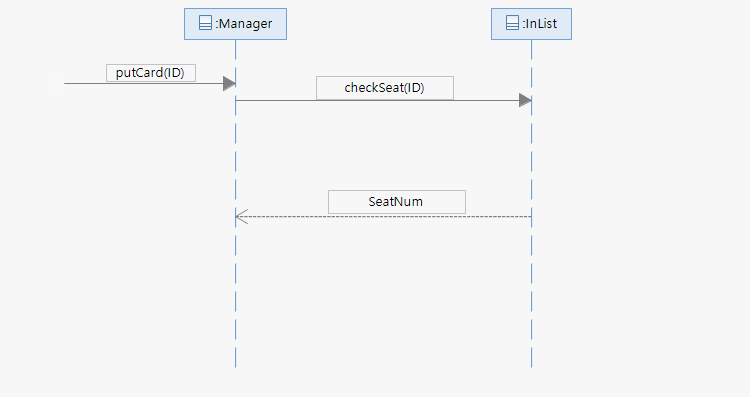
**Combined Design Class Diagram**



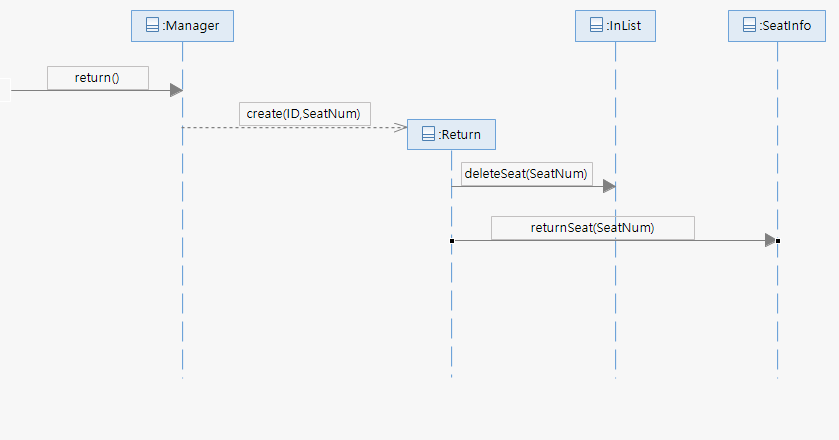
< Figure 23. Use Case 5. DCD >

**4.6. <Use case 6. ReturnSeat> Realization**

**Design Sequence Diagram**

****

< Figure 24. Use Case 6. DSD >



< Figure 25. Use Case 6. DSD >

**GRASP patterns**

Name: Controller

Problem: Who is responsible for ManageSeat system operation?

Solution:

* When a Union receives information from an IDcard, it requests information from each Object.
* First, ask UserInfoDescription if User is correct
* Then ask SeatInfoDescription if User has a seat.
* If you have a seat, change the reservation time for ReadingRoomDescription to one hour from now.

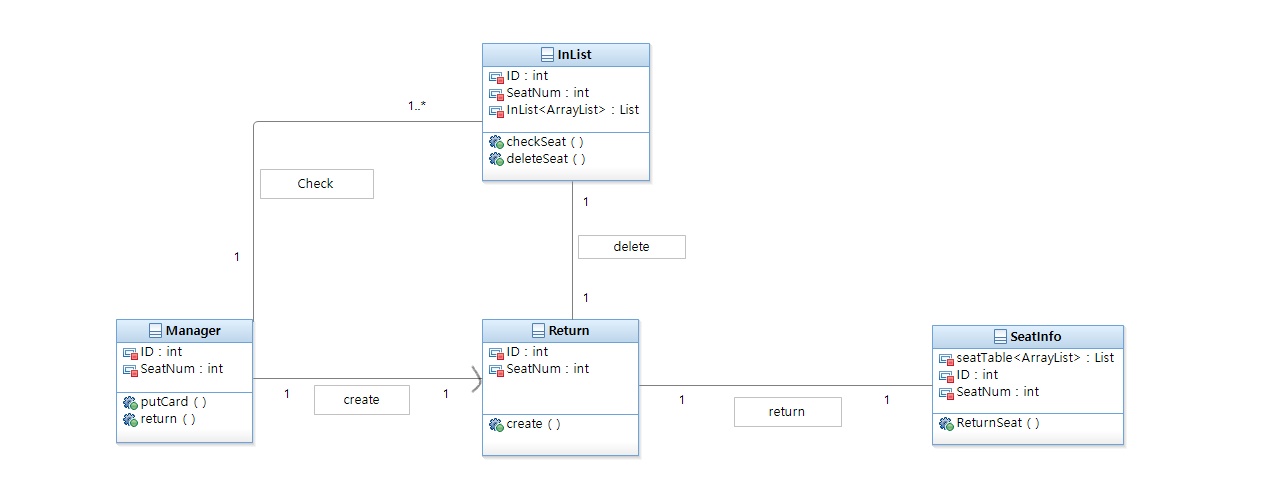
Name: Low Coupliing

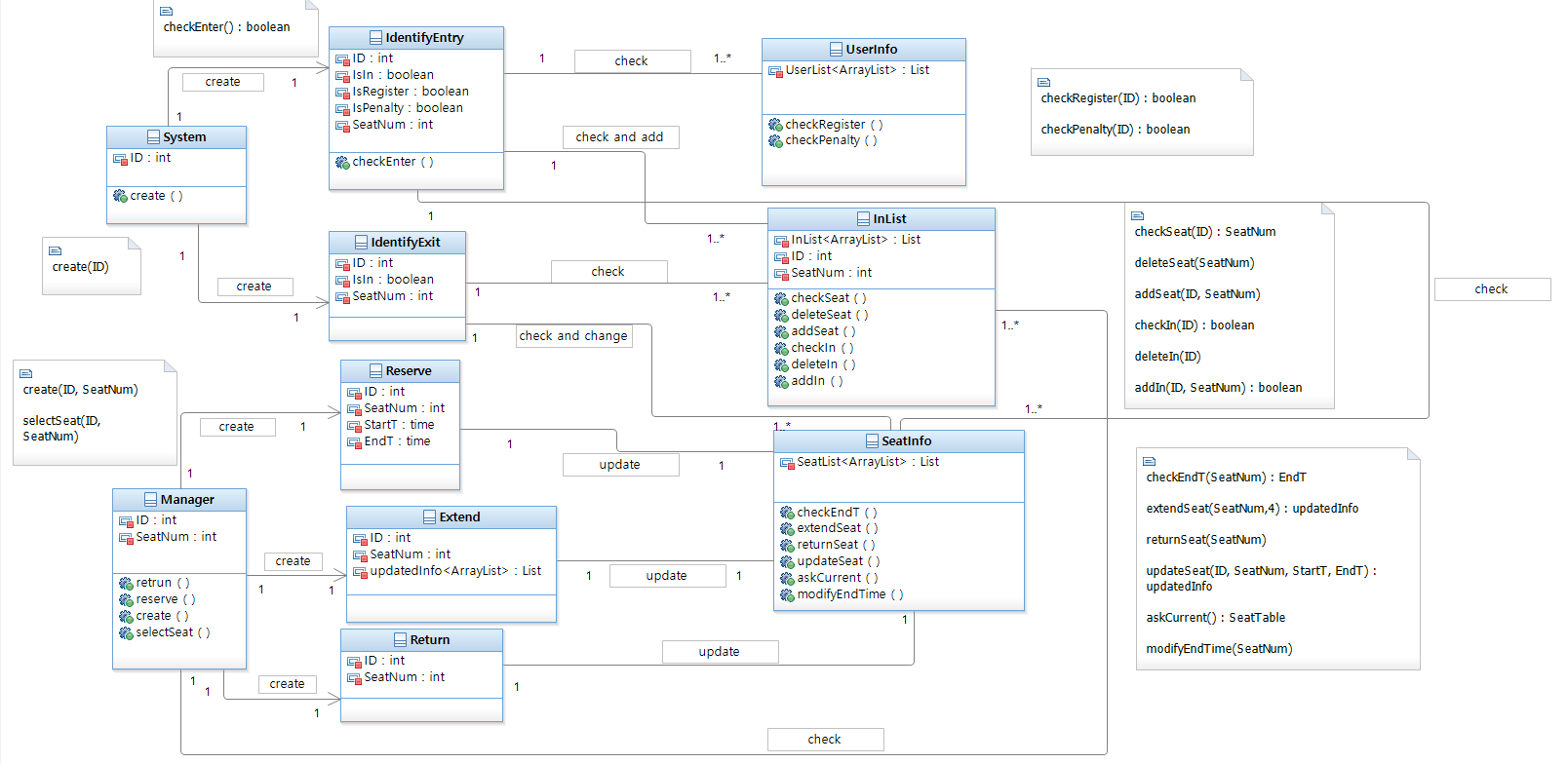
Problem: How to reduce the impact of seat information

Solution:

* The user may or may not have returned the seat.
* So, finding a place to return it and changing the holding time to return it should be separated. By doing, our system can reduce the tasks of ReadingRoomSystem.

**Combined Design Class Diagram**

****

****

**5. Architecture**

**5.1. Introduction**

* **logical view : To provide a basis for understanding the structure and organization of**

**the design of the system**

* **data view : First, we have to draw the id data in the entry service and check the data**
* **flow through UserInfo and InList. Secondly, we got the id data in the exit service, and the data flow through InList and SeatInfo. Third, we got the id data during the extension, reservation, and return of the seat, and the data flow through InList and SeatInfo**
* **process view : First, we are drawing a different process to access SeatInfo in the entry**
* **process.**
* **Second, we were in the process of accessing InList and SeatInfo during the**
* **exit process**
* **Third, we had to change the penalty in UserInfo during the automatic return**
* **of seat.**
* **deployment view : We were drawn because we updated the user's information every 6 months**
* **from external data.**
* **usecase view : The most important part of our usecase was the return and extension**
* **of the seat reservation. I thought that the reason would be the fast circulation**
* **of the seat due to our penalty granting and automatic return during the five**
* **usecases.**

**5.2. Architectural Factors**

**Introduction**

This is for documenting all of non-functional requirements of the use cases to supplement the use case models.

**Functionality**

* Logging and Error Handling: Log all of errors in the storage. Only the administrator can handle and access to those errors.

**Human Factors**

* To communicate or interact about the managing penalty, the user and administrator may uses touch screen UI on a flat panel monitor. So text must be visible from 0.5 meter.
* Use case 1 and 2 occur at the entrance of library. For the comfortable library and effective usage of reading room, the speed and error-free processing are very important.
* When the user gets penalty, the entry machine should give signal to the user to alert his/her penalty. Its shape should be sound not graphic. It’s because the penalty is given when user enters the library, that means, user interface is a entry machine, not a screen.

**Reliability**

* Recoverability: If there is error in the entry/exit machine, the administrator takes over the authority to control the entrance of the library from the machines. If the system fails to access the external system, it notifies immediately to the administrator through the graphic as a alert window.

**Performance**

* As mentioned at the human factors, the speed is important. So our goal is taking less than 1 second as an average time between placing ID card and opening the door.

**Supportability**

* Adaptability: When new user wants to use the library, he/she only needs registration of ID card to be allowed in the system, because the registration makes the user information accept the new one.

**Interfaces**

* Hardware: Entry Machine and Exit Machine is needed for control the entrance of the library and check the penalty.

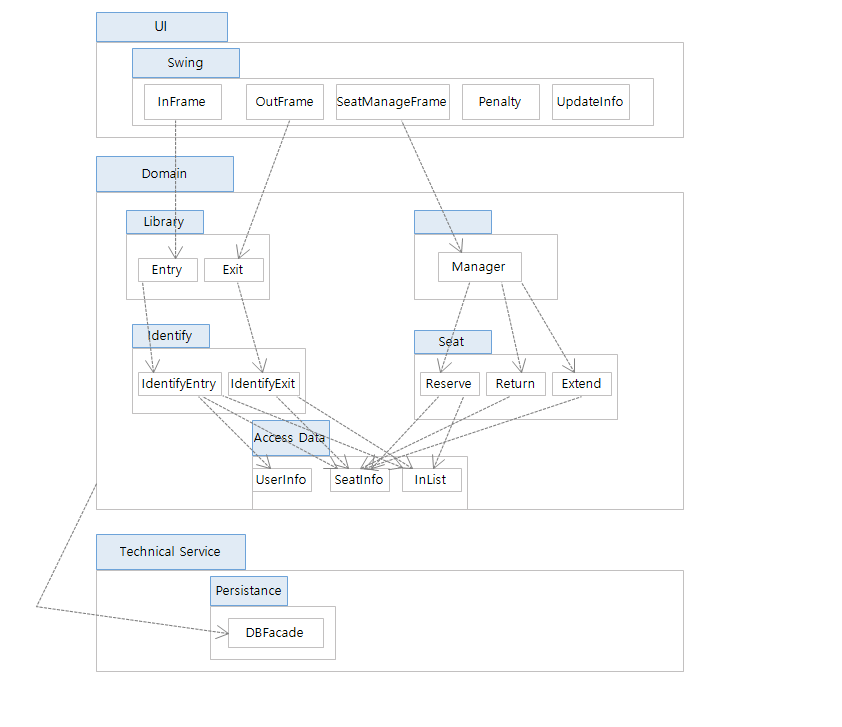
**Non-functionality**

The penalty rules are for efficient circulation of seats.

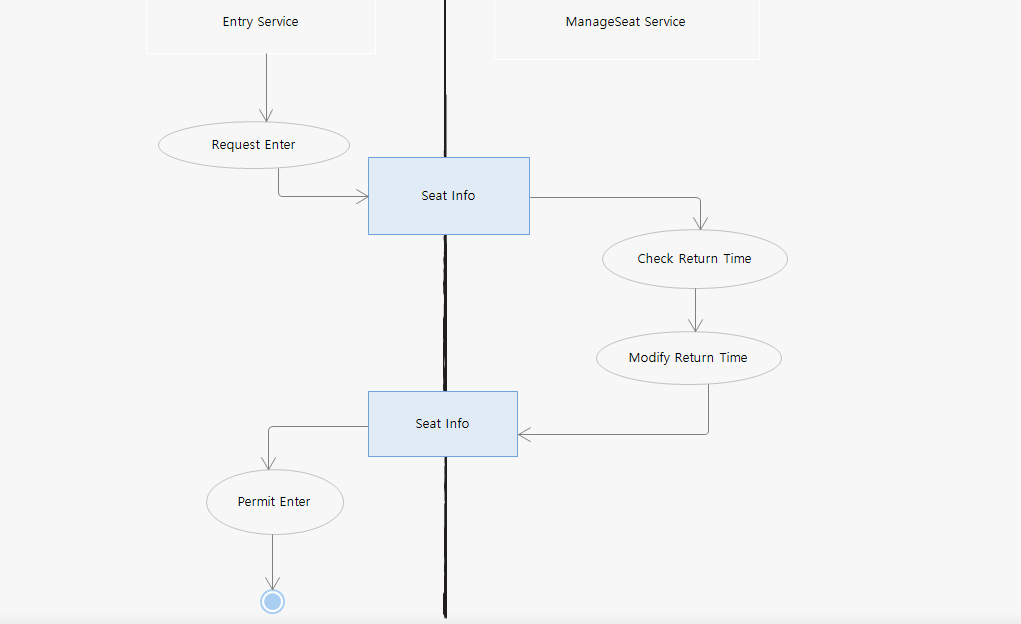
1. Duplicate access restriction rule : We have an InList, which is a list of people in the library. By using the InList, it is possible to prevent duplicate access. And penalties are imposed.
2. Seat automatic return rule : If user reserve a place in the library and then go out in the middle, user must come within one hour. If user arrive within an hour, the time will add up to four hours at the first reserved time. However, if user does not come in, seat will be returned and penalties are imposed.

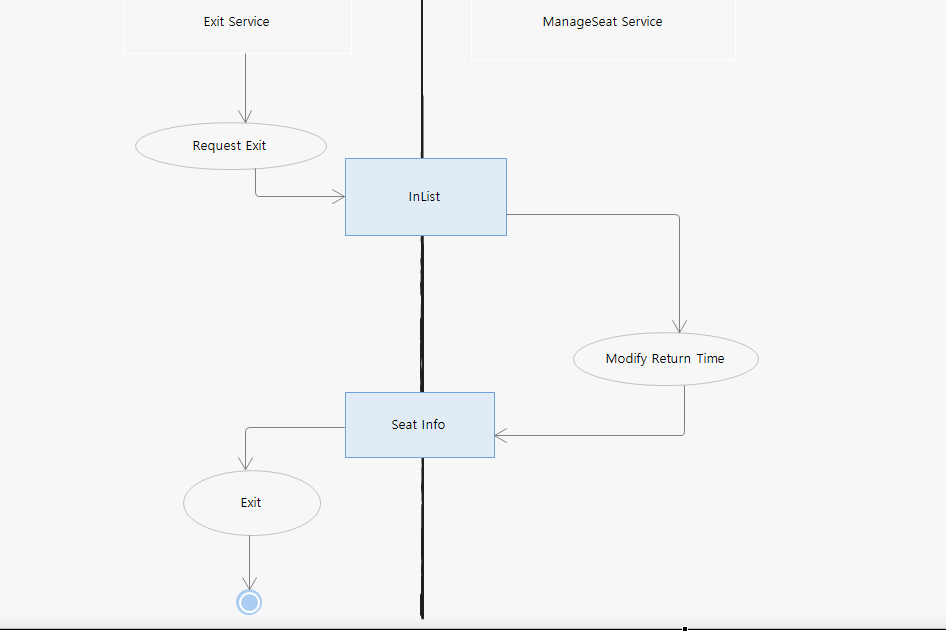
By using the rules described above, people who have accumulated penalties higher than a certain score can disable access to the library and lead to an efficient circulation of seats.

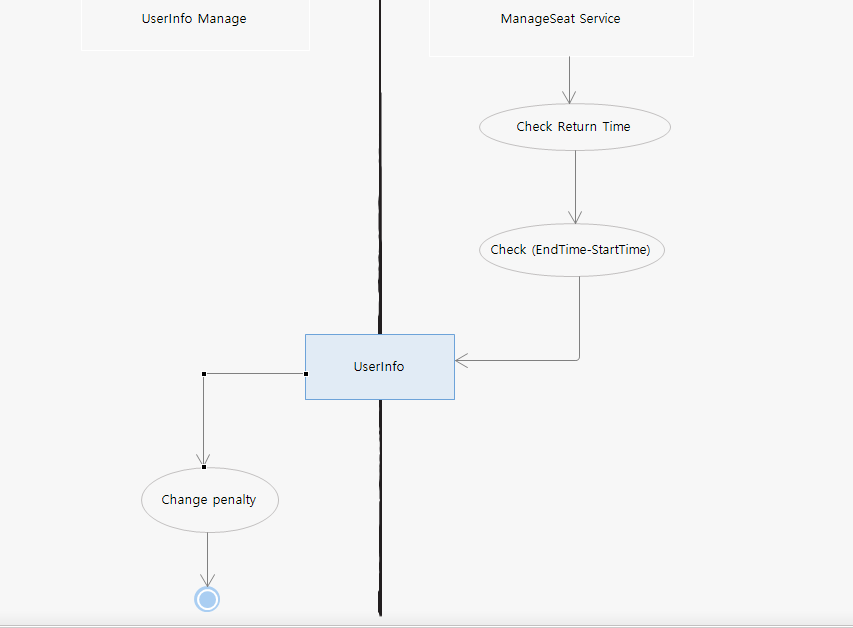
**5.3. Logical View**



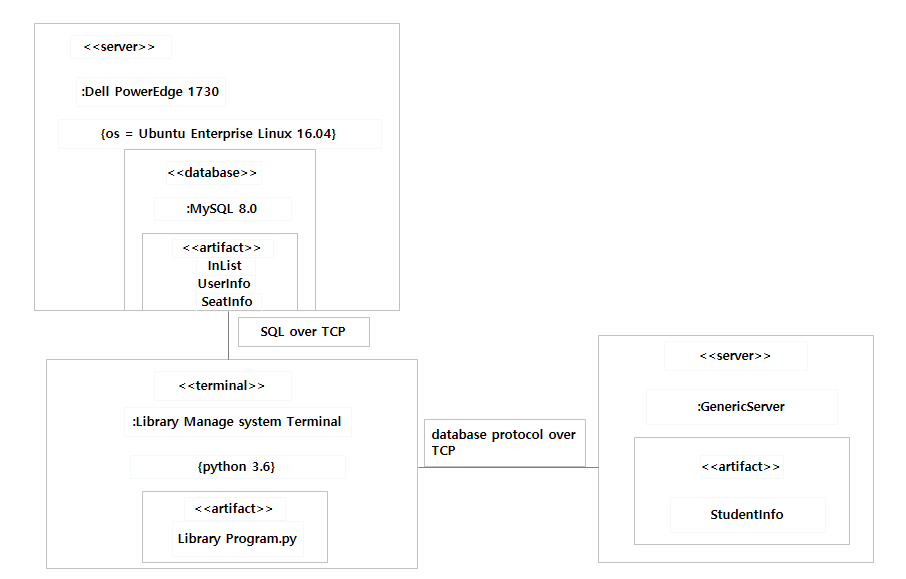
**5.4. Process View**

****

****

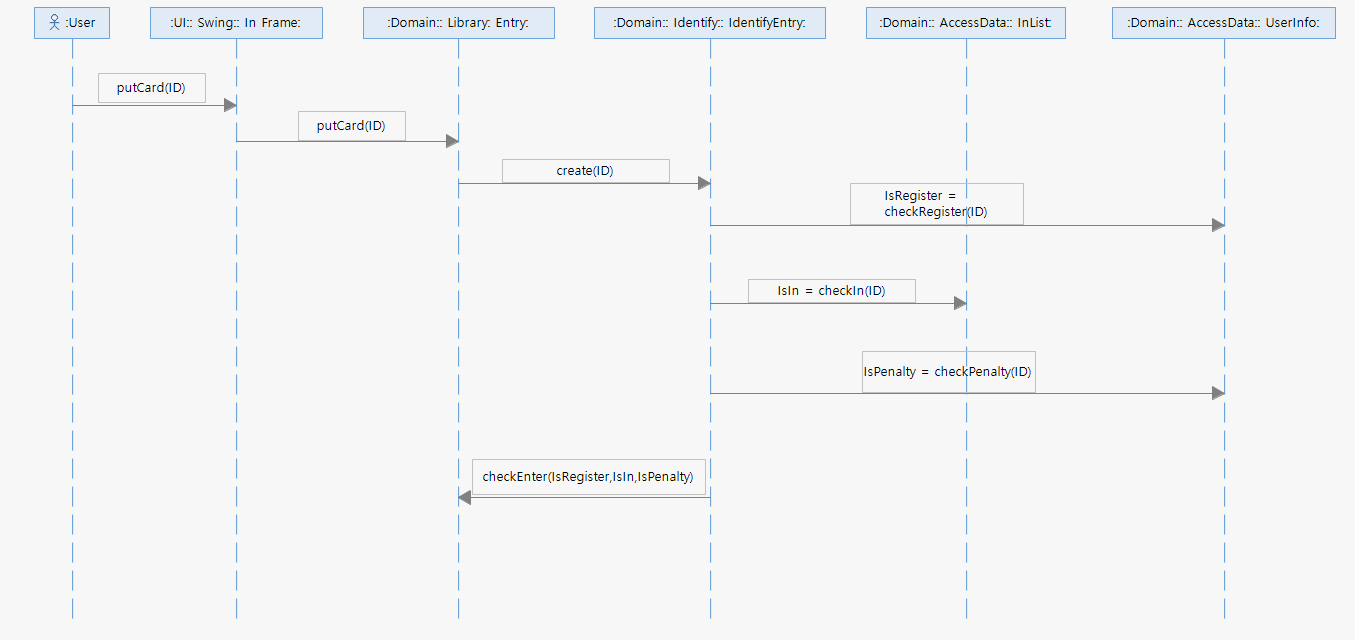
****

**5.5. Deployment View**

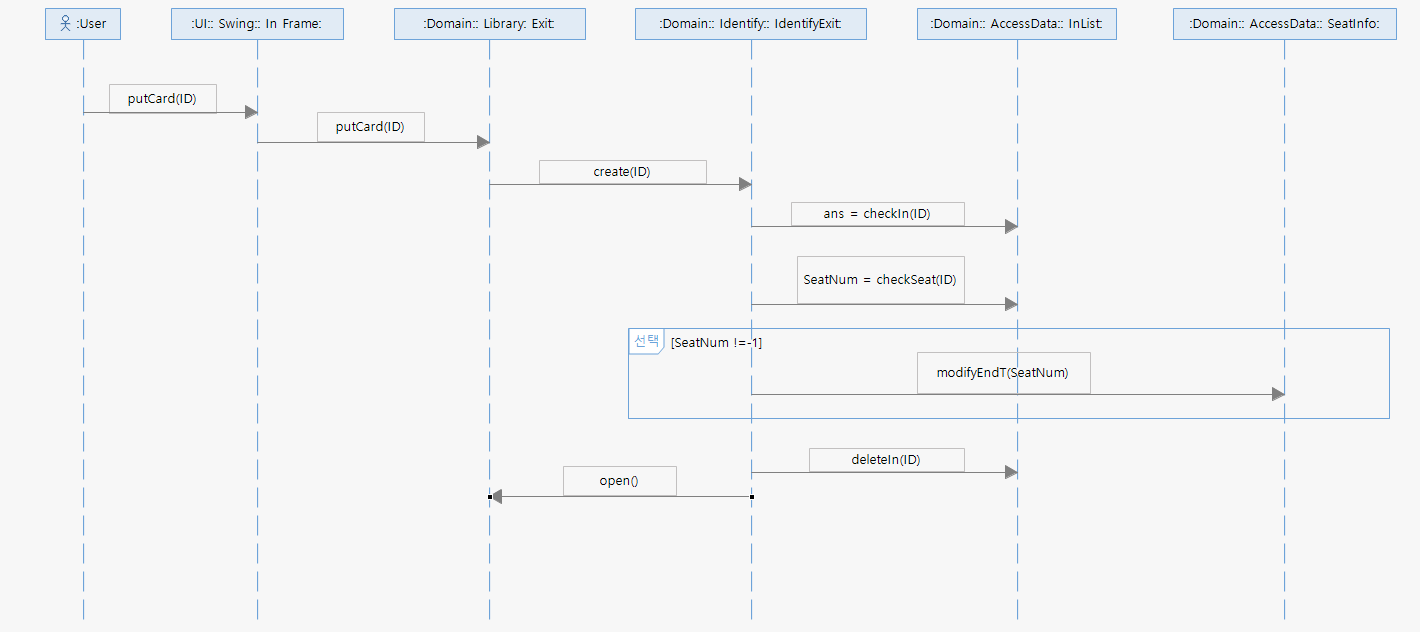
****

**5.6. Use Case View**

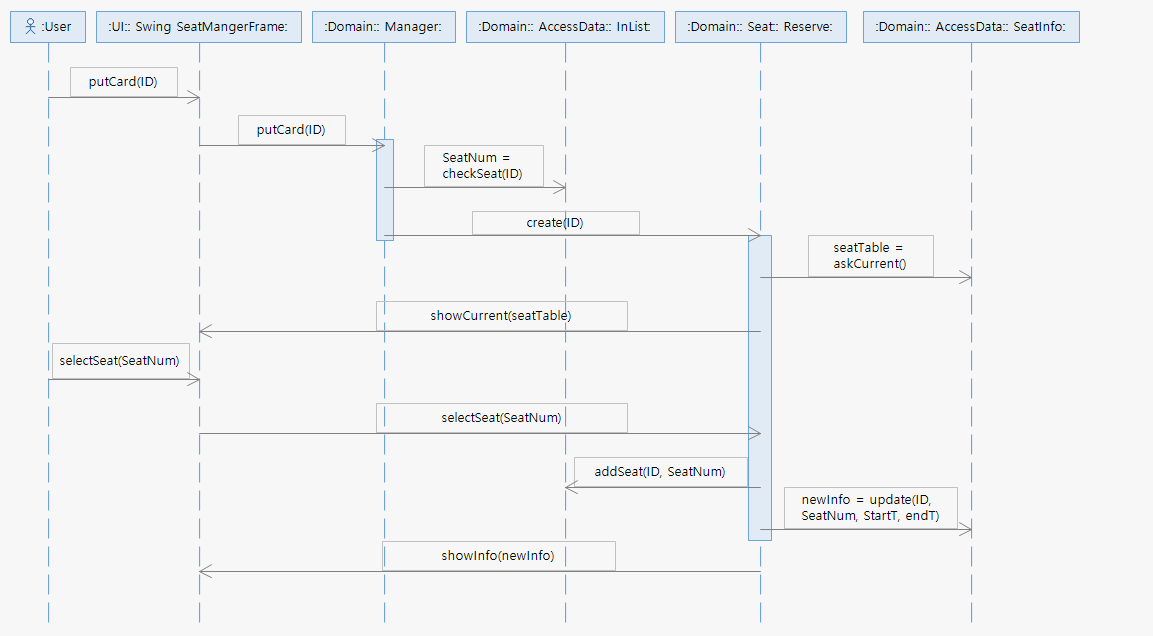
**5.6.1. Use case 1. EnterLibrary View**

****

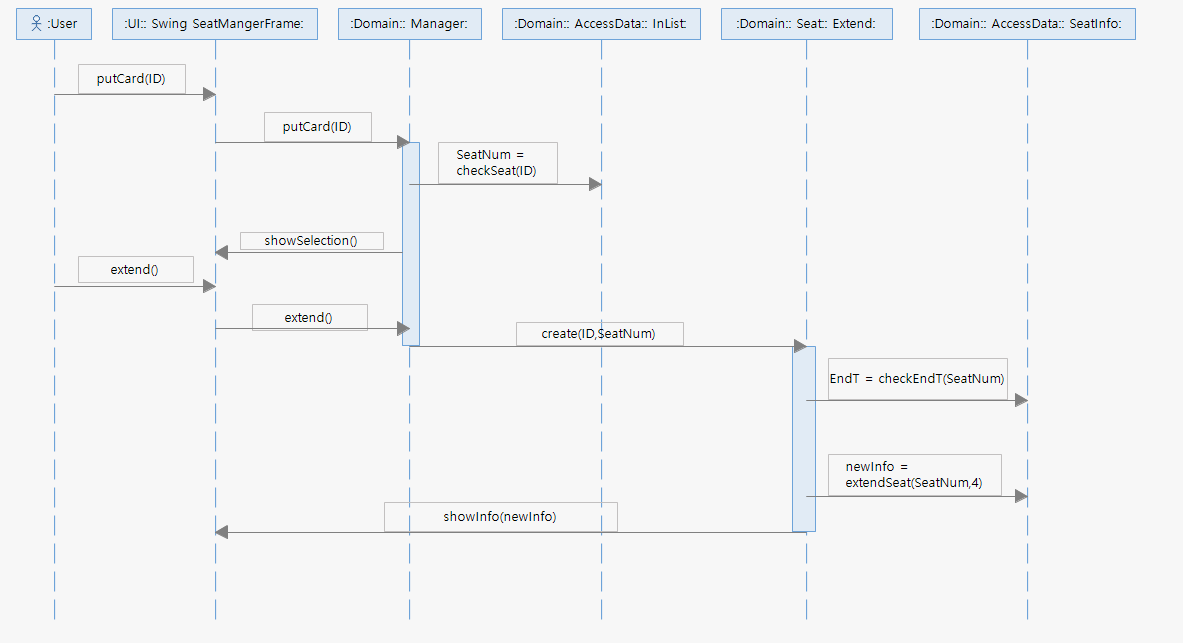
**5.6.2. Use case 2. ExitLibrary View**

****

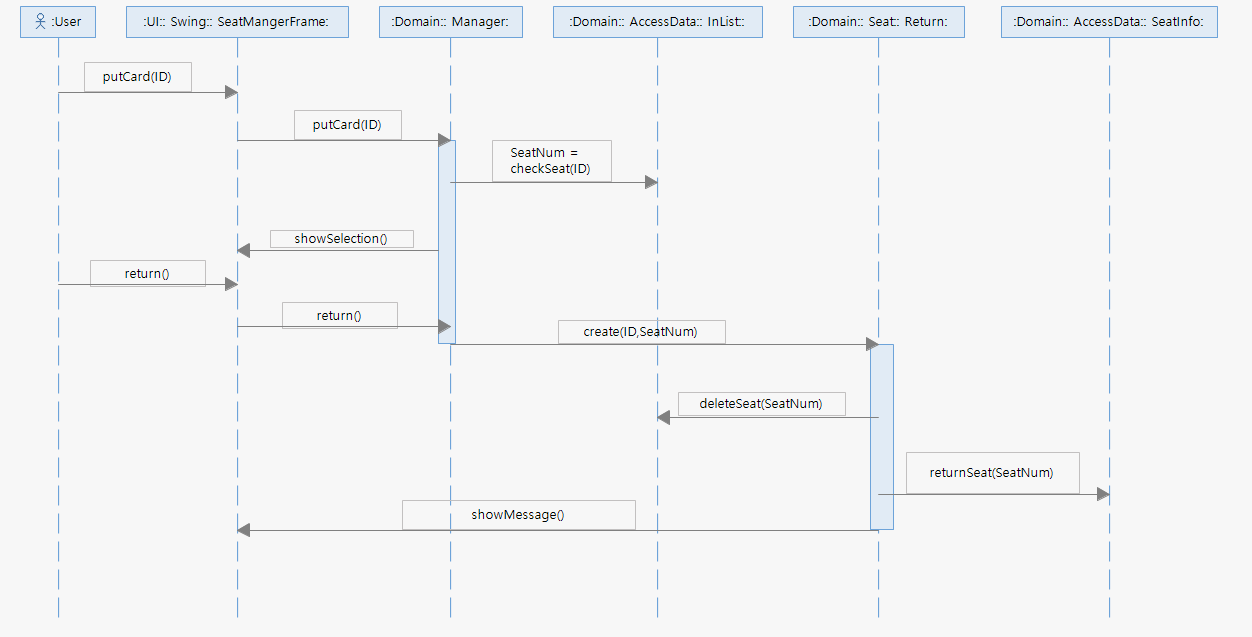
**5.6.3. Use case 4. ReserveSeat View**

****

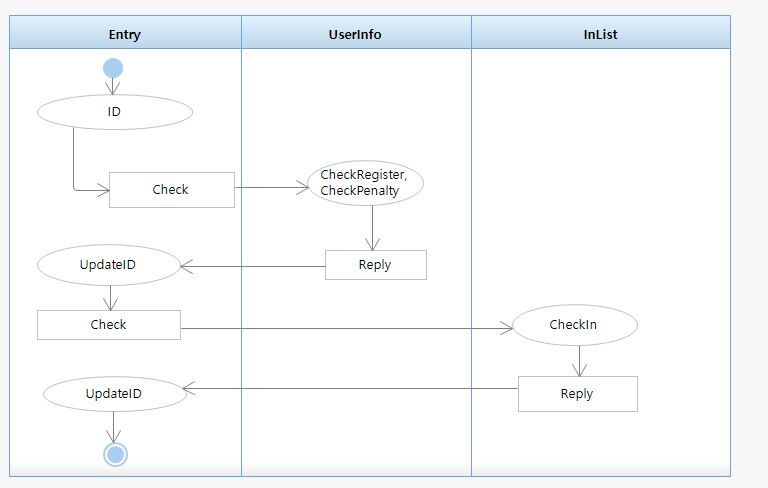
**5.6.4. Use case 5. ExtendSeat View**

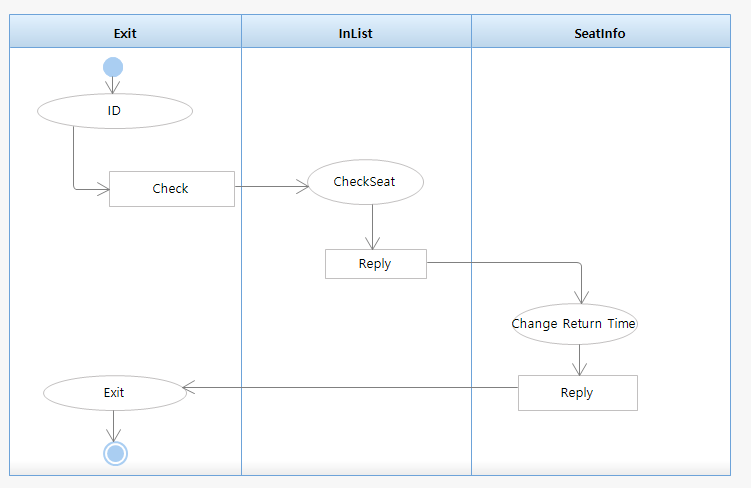
****

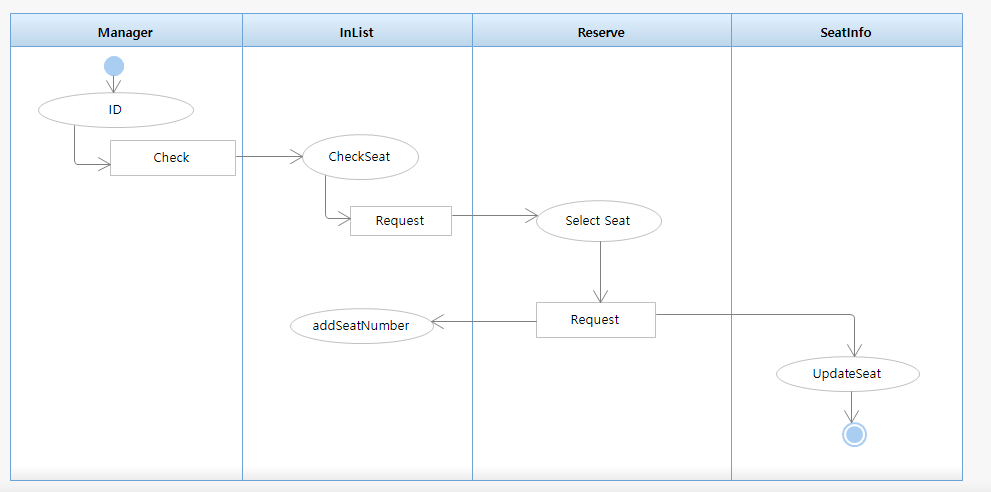
**5.6.5. Use case 6. ReturnSeat View**

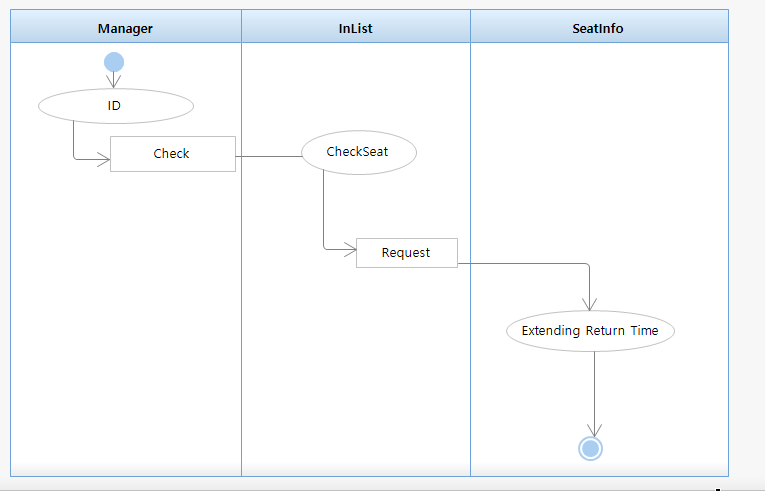
****

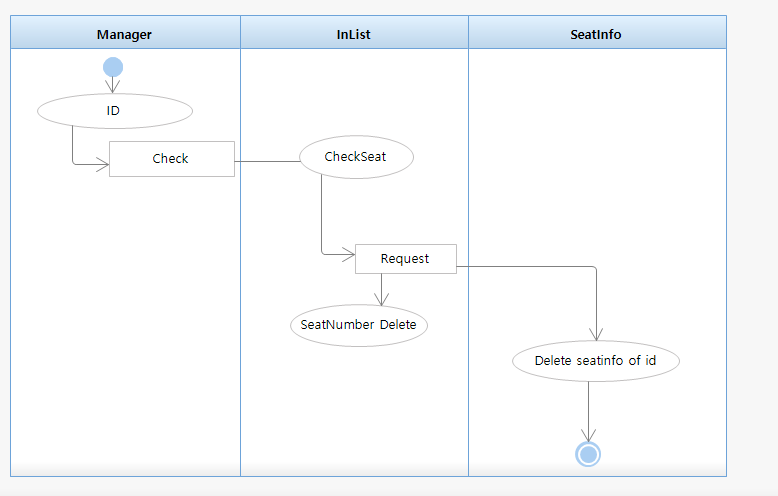
**5.7. Data View**

****

****

****

****

****

**6. Conclusion**

We are currently experiencing problems with the inefficient operation of our school library reading room system and trying to improve the problems. Despite the fact that all the seats reserved for the trial period are not being used, there is no seat available for reservation, so you must wait in line.

We thought that there were two major causes of this problem.

1. By using Mobile user ID Card and user ID Card, Access of outsiders
2. Non-returned seats due to users not returning, even though they do not use the reading room seats.

Designed to solve these problems.

Our school library reading room system is only accessible if we have a user ID card and there is no problem with reservation. To solve this problem, InList was created separately to manage the people inside the library and prevent duplication of access to prevent outsider access. By adding a database of users in the library in the existing system, we effectively prevented outsider access. Additionally, penalties were imposed on users who attempted duplicate entry.

And now, our school library reading room system does not manage the reading room user separately. We made it possible to manage by adding a database of reading room users. So that the users of the reading room can see whether they are inside or outside the library. If user in outside does not come in within one hour, seat is automatically returned and charged an additional penalty. By automatically returning and imposing a penalty and adding a database, the seat was effectively prevented from being vacated for a long period of time.

The design of the part we wanted to implement was designed with the exception of Use Case 7. UpdateInfo. The Demo program is divided into Use Case 1. EnterLibrary and Use Case 2. ExitLibrary.

The program was implemented using Python and we replaced it with receiving the ID of the student through the UI because the card sensor is not yet available.

When entering the library, first check that the student's ID is correct. Also check if you are already in the library. This imposes penalty by checking the duplication of the ID. If there is no problem, after adding the student to InList, the door will open.

When you leave the library, you will also be asked for your student ID. If the ID is not correct, you will be notified to contact the administrator. If the student has not yet returned the seat, set the seat to return after one hour. If there is no problem, after removing the student in InList, the door will open.

We only write a small program with a personal project. So I did not feel the need for design. However, if the project is large, it is more efficient to create the program by dividing the team members.

When team members work together, the most important thing is to clearly define the roles of each use case. Also, the names of variables and methods should be unified. All of these tasks are identical to the design process.

If you design your design clearly, when you implement your code, the tasks of each Class will not overlap. Also, when correcting errors, you can fix them more efficiently.

We have made additional use of Naming, making sure that the role of Use Cases is correct, and that a slight improvement is required in the design part of Use Case 3. ManagePenalty. Penalty is a new addition to ours, and there are still a lot of places to go.

Before developing the software, we learned that I need to understand the requirements of the User. If we have learned how to efficiently capture requirements and have learned the requirements, we have learned how to analyze how to design software efficiently with these requirements. We learned about this development process by figuring out the need and not just developing the software.

**7. Reference**

* Lecture Note
* Craig Larman-Applying UML and Patterns\_ An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)-Prentice Hall (2004)
* <http://blog.naver.com/PostView.nhn?blogId=bestymw&logNo=130042282790&parentCategoryNo=&categoryNo=6&viewDate=&isShowPopularPosts=true&from=search>
* <http://39ya.tistory.com/163>
* <https://blog.naver.com/ljh0326s/221001892737>
* <http://csis.pace.edu/~marchese/CS389/L9/Sequence%20Diagram%20Tutorial.pdf>
* <https://cafe.naver.com/architect1/191>

**8. Appendix**

**8.1. Glossary**

* Administrator : Manager managing Ajou University library system
* User : Use Ajou University library people.
* Penalty : the act that users break the rule of library
* User ID card : pass card for Ajou University Library.
* Mobile ID card: Same work with student ID card, but on the phone.
* EntryMachine : Machine for managing library enter.
* ExitMachine : Machine for managing library exit.
* Reading room machine(= terminal) : A place where students can study. students have to reserve the seat for usage
* SeatInfoSystem : Can bring up seat information and show seat information.
* UserInformation : Can bring up user information.
* Reservation : Reserve reading room seat.
* Extension : Extend reading room seat.
* Return : Return reading room seat.
* SeatHoldingTime : Time to keep seat.

**8.2. Source Code**

**7. Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** |
| Inception draft | Oct 2, 2018 | First draft. To be refined primarily during elaboration. | 치즈수플레 |
| Elaboration draft | Nov 2, 2018 | Second draft. | 치즈수플레 |
| Final draft | Dec 7, 2018 | Final draft. | 치즈수플레 |