

**CS3343 Software Engineering Practice**

**Analysis and Design Report (Group 5)**

**Project Title:** Sport Facility and Equipment Booking System

|  |  |
| --- | --- |
| **Name** | **Student Number** |
| NG Ka Hung | 58533046 |
| LIN Weiyuan | 58537684 |
| LAU Wing Him | 58537137 |
| IP Wing Kai | 58120400 |
| LEE Kin Wai | 57925345 |
| Chan Sing To | 58533974 |

**Table of Content**

[**1. Design Constraints 3**](#_sj95d5wrm7l0)

[**2. Use case 4**](#_67eh7a15lngf)

[**3. Class diagram 11**](#_2pd1k3q125eg)

[3.1. Overall Class Diagram 11](#_bvu540vifnjv)

[3.2. Equipment, Facility 12](#_f1qlh3vsnbaw)

[3.3. Book Record 12](#_ho0vv2mkn3tl)

[3.4. Functions classes 13](#_h6nu6o5vl5or)

[**4. Design Pattern 18**](#_i755fcbexykf)

[4.1. Singleton Pattern 18](#_h065fegtuktg)

[4.2. State Pattern 18](#_be4bawu3414)

[**5. Design Principle 19**](#_ulk00pzc7fzl)

[5.1. Single Responsibility Principle 19](#_6j9taux45tlu)

[5.2. Open Closed Principle 19](#_q38s9zv5u1hi)

[5.3. Dependency Inversion Principle 20](#_x20aqkpe2xv4)

[**6. Sequence diagram 21**](#_abx08hp8gkp3)

[6.1. Book facility 21](#_cmmkkhblagd3)

[6.2. Book equipment 23](#_3ytk3ksjr37u)

[6.3. Handling Current Booking 24](#_2d5qdjekxccx)

[Update booking facility 24](#_e0276euzob2i)

[Update booking date time 25](#_jufa90e502kr)

[Cancel Booking 27](#_ysae36ywvxay)

[6.4. Handling confirmed booking 28](#_m370fn3iz0ee)

[Update Booking facility 28](#_riql7jbdml5u)

[Update booking date time 30](#_v0je28qsu8zz)

[Cancel Booking 32](#_evvvr5c1bfpi)

# 

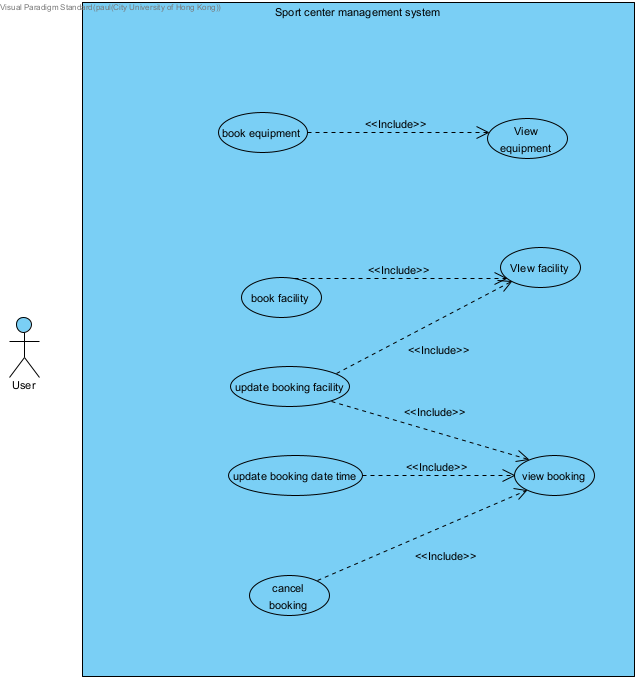
# 1. Design Constraints

The main constraints and boundaries of the project are as follow:

* **User Interface will be limited to a console-based (text) environment.**
  + Due to the project being a "SportApp java" application, the user interface will be text-based. This limits the use of graphical calendars, drag-and-drop scheduling, or integrated visual feedback.
* **Data persistence will be managed without a formal relational database.**
  + Owing to project scope, the system will rely on in-memory structures (like ArrayList and HashMap, as noted in FacilityBookManager) to manage bookings and user data during runtime. Confirmed records may be stored in simple files, but the system will not implement a full-scale SQL or NoSQL database.
* **The payment module will be a simulation rather than a functional transaction system.**
  + To keep the project scope manageable, the payment function (e.g., "Octopus") will only simulate a successful transaction (e.g., "Please finish payment via Octopus app (hold 5s then go next)"). It will not integrate with any external payment gateways but will update the booking status from **Pending Payment** to **Completed Payment** upon a simulated success.
* **Complex business logic for penalties, refunds, and memberships is simplified or excluded.**
  + To focus on the core booking lifecycle, complex business rules are omitted. For example, the system enforces a "no refund" policy for completed bookings, and concepts like penalties for late returns or membership-based discounts ("TOOMYYMembership benefits") are considered outside the primary scope.
* **User "shopping cart" (Pending Booking) data is non-persistent between sessions.**
  + The "View Pending Booking" function acts as a temporary shopping cart. If the user selects "E Exit" from the Home Page before completing payment, any items in this pending state will be lost. Only bookings that have gone through the payment simulation are considered permanent records.

# 

# 2. Use case



|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | UC1 | |
| **Use Case Name:** | Update Booking Date Time | |
| **Actor(s):** | User | |
| **Description:** | User wishes to update the date and time of an existing booking. The system allows the user to select a booking and choose a new available date and time slot. | |
| **Trigger:** | This use case is initiated when the user selects the option to update booking date and time. | |
| **Normal Flows:** | **Actor Action** | **System Response** |
| Step 1: User selects a booking to update  Step 3: User inputs a new date for the booking  Step 5: User selects an available time slot from the displayed options | Step 2: System validates the selected booking and checks if it can be updated (cannot update today's bookings)  Step 4: System retrieves available time slots for the selected date  Step 6: System attempts to update the booking with the new date and time  Step 7: System confirms whether the update was successful or failed |
| **Alternative flow** |  | |
| **Post-conditions:** | The selected booking has been updated with the new date and time, or the system displays an appropriate error message if the update failed. | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | UC2 | |
| **Use Case Name:** | Cancel booking | |
| **Actor(s):** | User | |
| **Description:** | User wishes to cancel an existing booking. The system prompts for confirmation before proceeding with the cancellation. | |
| **Trigger:** | This use case is initiated when the user selects the option to cancel a booking. | |
| **Normal Flows:** | **Actor Action** | **System Response** |
| Step 1: User selects a booking to cancel  Step 3: User confirms cancellation by entering 'Y' | Step 2: System displays the selected booking and prompts for confirmation  Step 4: System processes the cancellation request  Step 5: System confirms successful cancellation |
| **Alternative flow** | Step3: User disagree to cancel the booking by entering ‘N’ | |
| **Post-conditions:** | The selected booking has been cancelled and removed from the system, or the cancellation was aborted by the user. | |

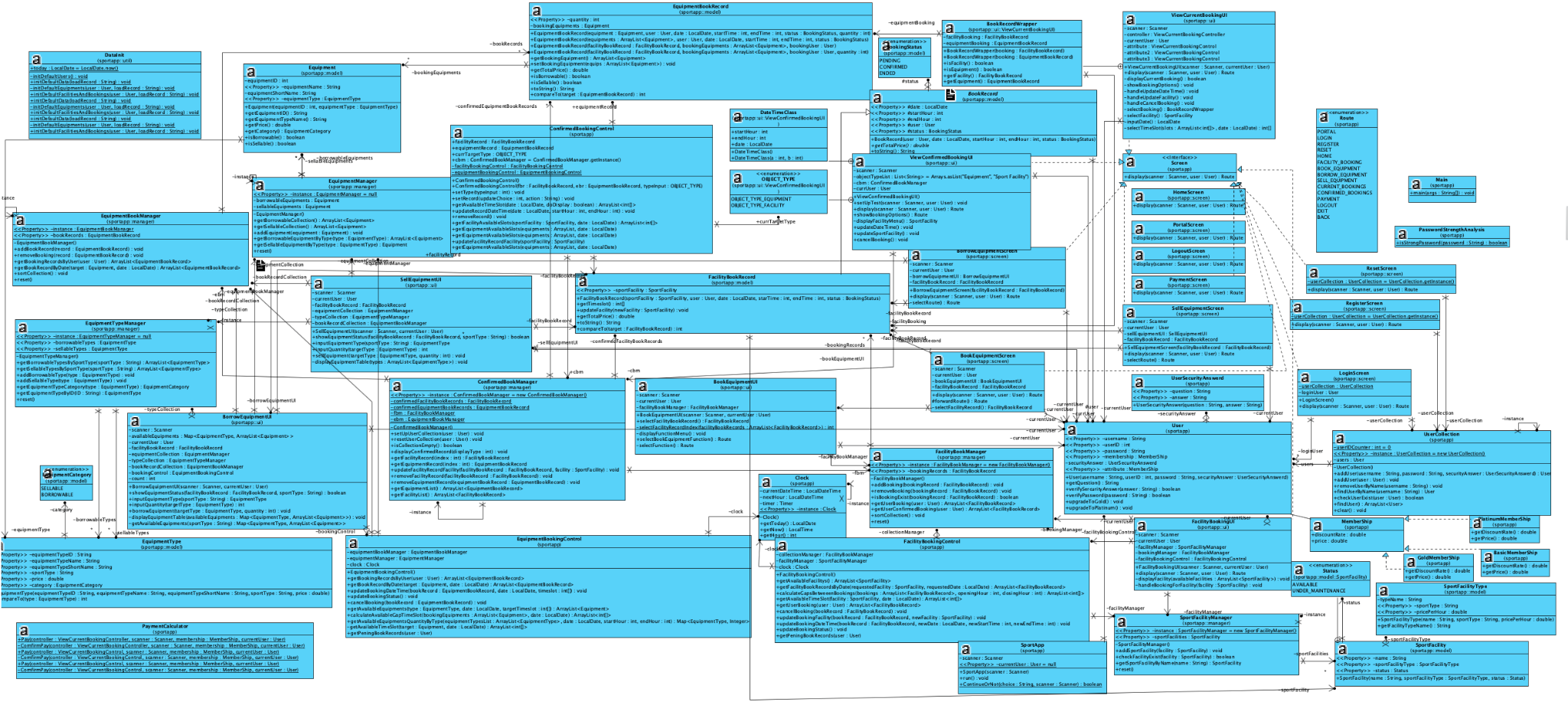
|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | UC3 | |
| **Use Case Name:** | Update Booking Facility | |
| **Actor(s):** | User | |
| **Description:** | User wishes to change the facility for an existing facility booking. The system allows the user to select a new facility and available time slot. | |
| **Trigger:** | This use case is initiated when the user selects the option to update booking facility. | |
| **Normal Flows:** | **Actor Action** | **System Response** |
| Step 1: User selects a booking to update  Step 3: User selects a new facility  Step 4:  Step 5: User inputs a new date  Step 7: User selects an available time slot | Step 2: System validates that the selected booking is a facility booking  Step 4: System prompts user to select a new date  Step 5:  Step 6: System retrieves available time slots for the selected facility and date  Step 8: System attempts to update the booking with the new facility, date, and time  Step 9: System confirms whether the update was successful or failed |
| **Alternative flow** |  | |
| **Post-conditions:** | The selected facility booking has been updated with the new facility, date, and time, or the system displays an appropriate error message. | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | UC4 | |
| **Use Case Name:** | Book Sport Facility | |
| **Actor(s):** | User | |
| **Description:** | User wishes to book a sport facility by selecting a date and available time slot. The system guides the user through the booking process with validation checks. | |
| **Trigger:** | This use case is initiated when the user selects the option to book facility. | |
| **Normal Flows:** | **Actor Action** | **System Response** |
| Step 1: User selects a specific sport facility to book  Step 3: User inputs booking date in dd/MM/yyyy format  Step 5: User inputs start hour and end hour for the booking | Step 2: System prompts user to enter booking date in dd/MM/yyyy format  Step 4: System validates the time input against business rules  Step 6: System creates a new booking with PENDING status  Step 7: System displays booking confirmation details |
| **Alternative flow** |  | |
| **Post-conditions:** | A new facility booking has been created with PENDING status, and confirmation details are displayed to the user. | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | UC5 | |
| **Use Case Name:** | Book Sport Equipment | |
| **Actor(s):** | User | |
| **Description:** | User wishes to book a sport Equipment by selecting a date and available time slot. The system guides the user through the booking process with validation checks. | |
| **Trigger:** | This use case is initiated when the user selects the option to book Equipment. | |
| **Normal Flows:** | **Actor Action** | **System Response** |
| Step 2: User selects a specific Equipment type to book  Step 4: User inputs booking date in dd/MM/yyyy format  Step 6: User inputs start hour and end hour for the booking | Step 3: System prompts user to enter booking date in dd/MM/yyyy format  Step 5: System validates the time input against business rules  Step 8: System creates a new booking with PENDING status  Step 9: System displays booking confirmation details |
| **Alternative flow** |  | |
| **Post-conditions:** | A new Equipment booking has been created with PENDING status, and confirmation details are displayed to the user. | |

# 3. Class diagram

## 3.1. Overall Class Diagram

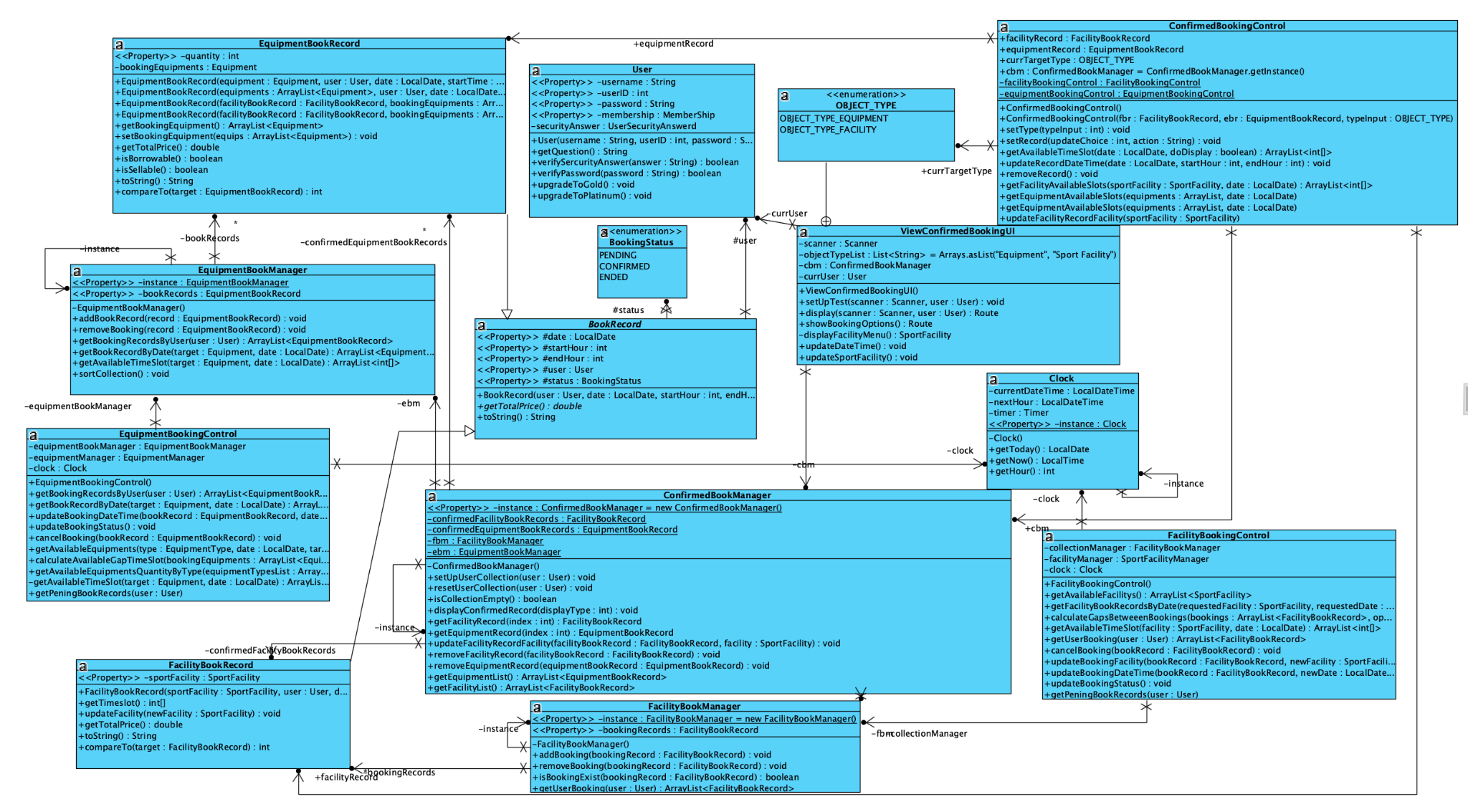
As seen in the graph, this is the overall class diagram. Since the graph is so complicated and there are too many classes to see in the graph. The overall class diagram is broken down by different functions and features in the following class diagrams.

## 

## 3.2. Equipment, Facility

These are the basic elements that the system utilizes and manipulates with. The information contained in Equipment and Facility cannot be altered by the user during the run of the program. As seen above, the equipment contains other class object like equipment type and facility contains facility type.

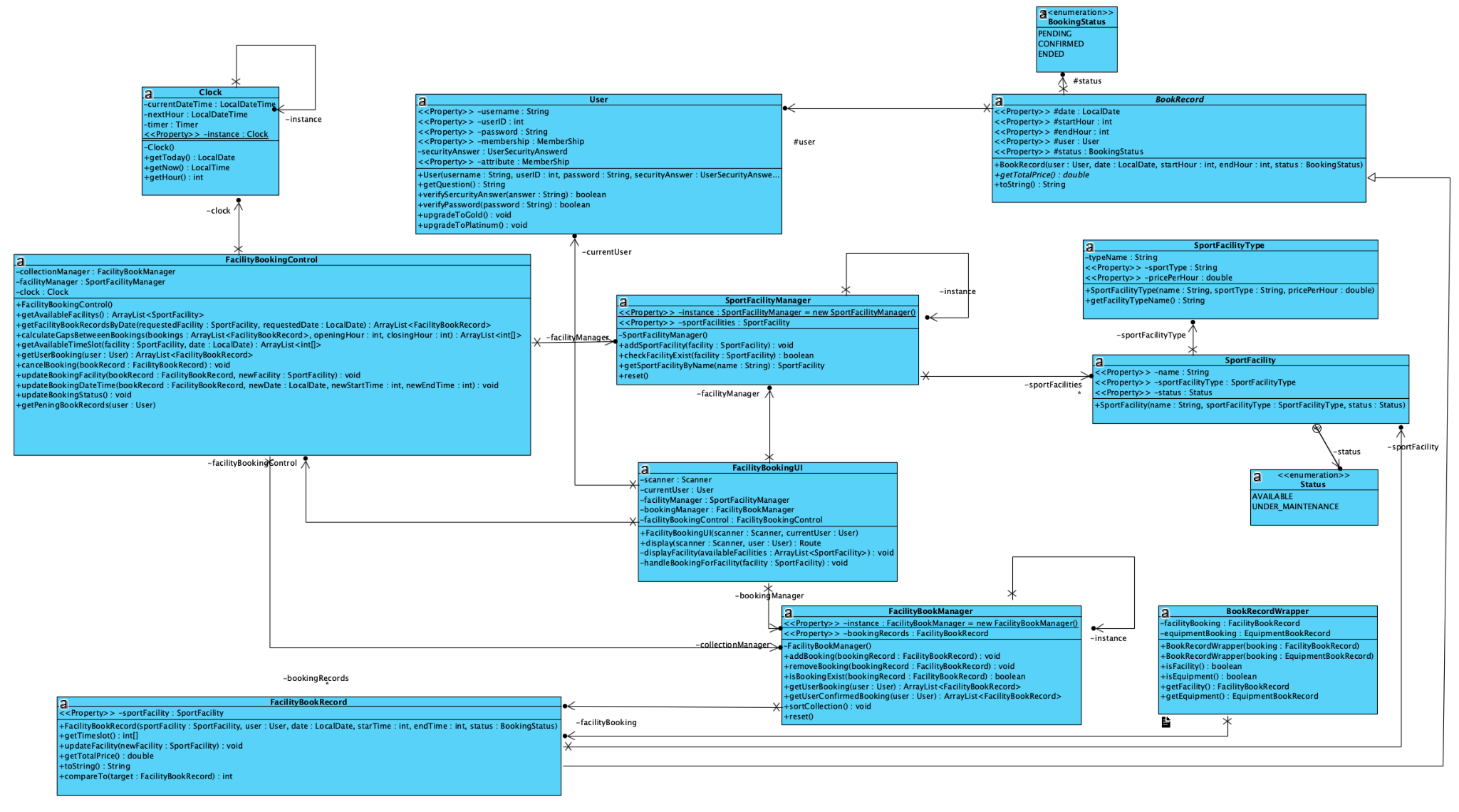
## 3.3. Book Record



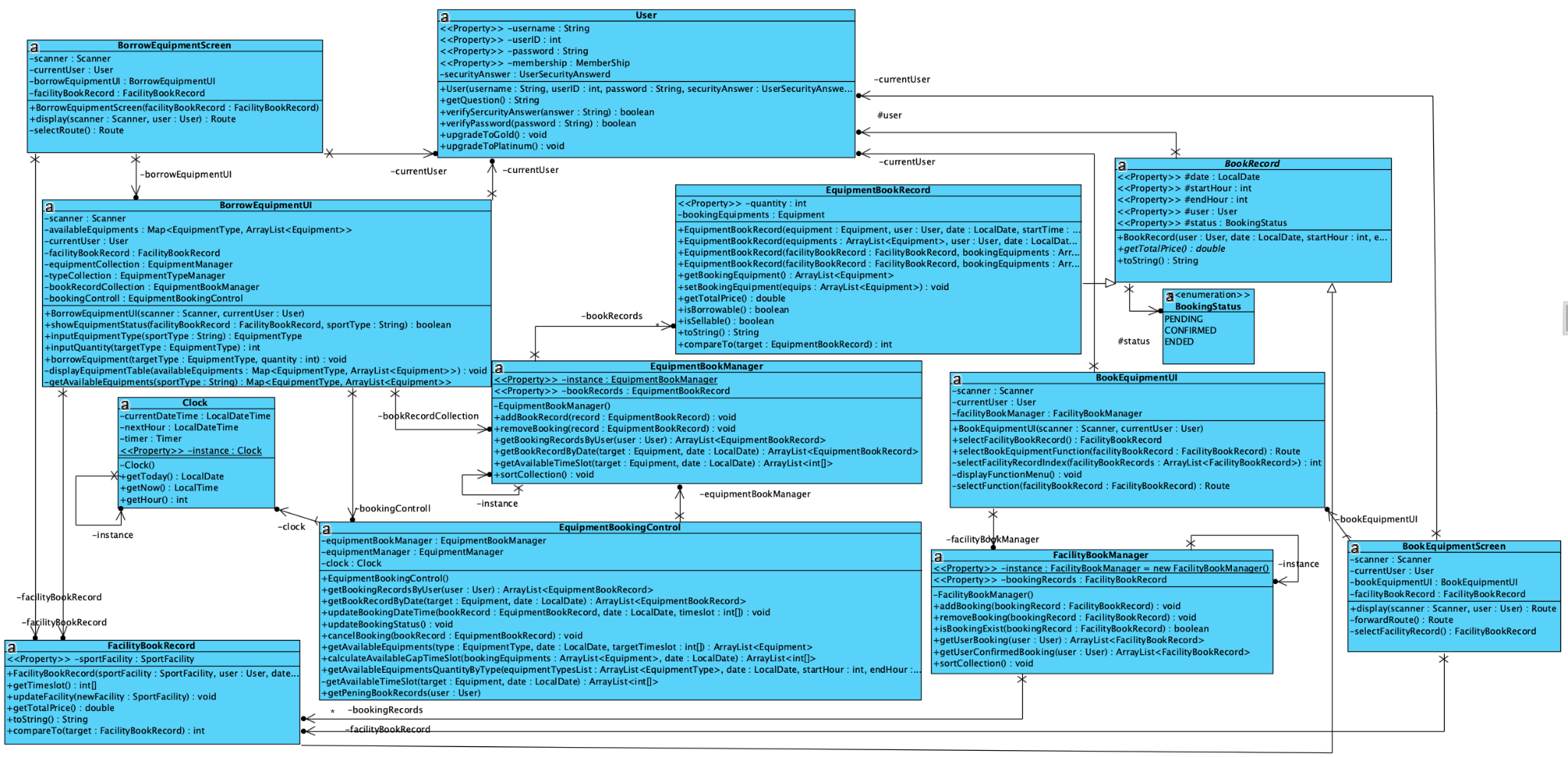
Equipment and Facility are treated as the oriented entity, thus, EquipmentBookRecord and FacilityBookRecord are built for the record modification, deletion. Both classes extended from the super class BookRecord.

## 3.4. Functions classes

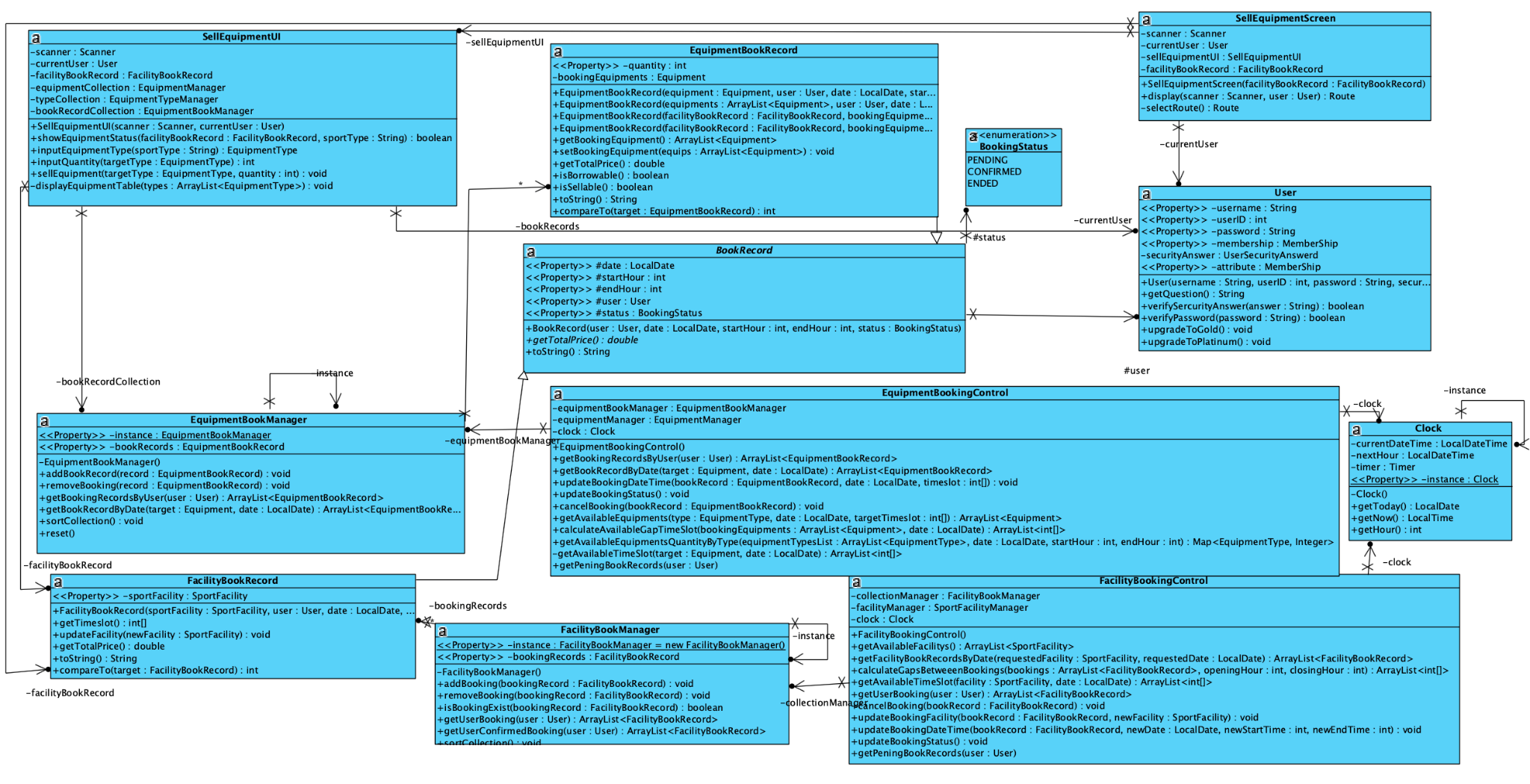
**Book Facility**



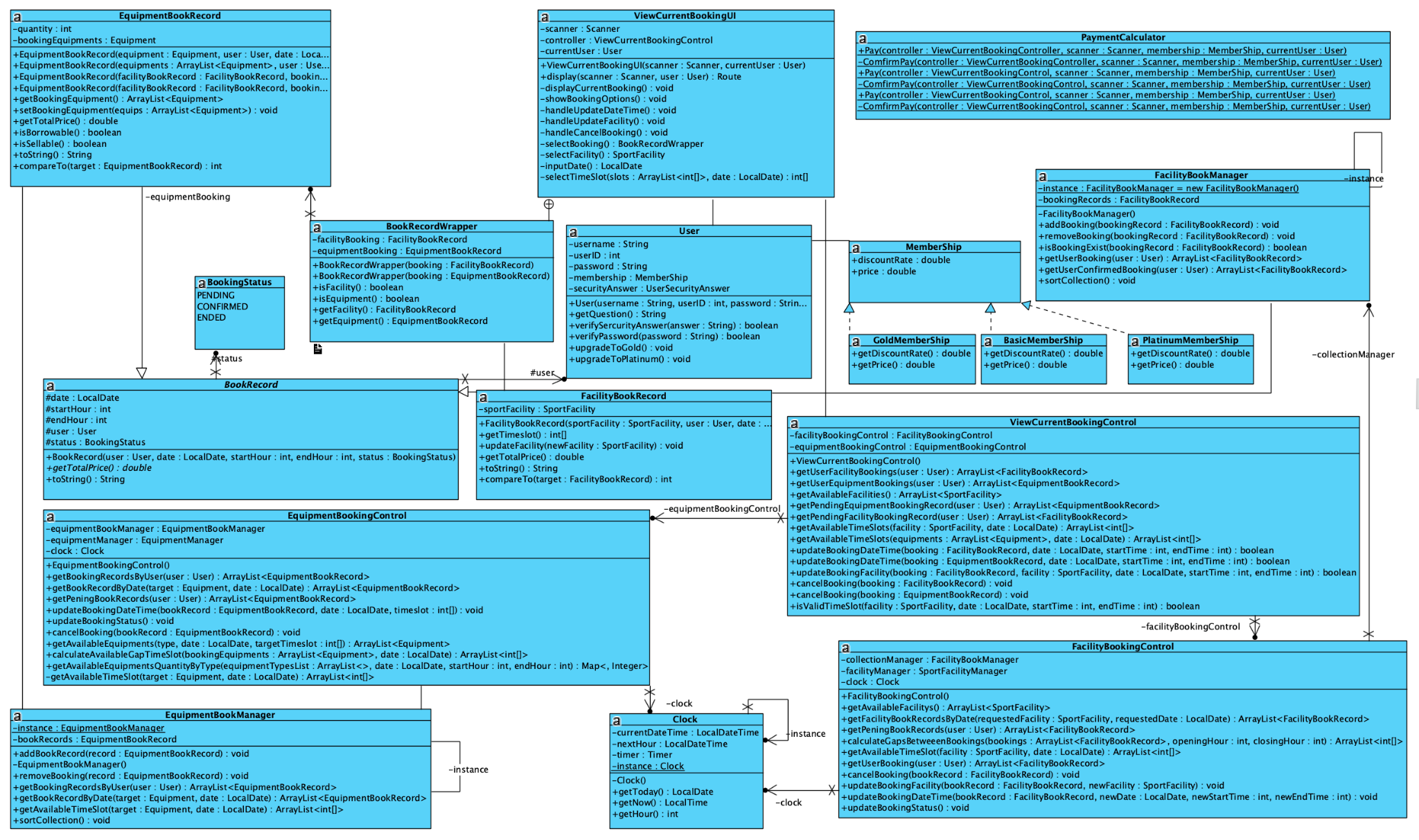
**Book Equipment**



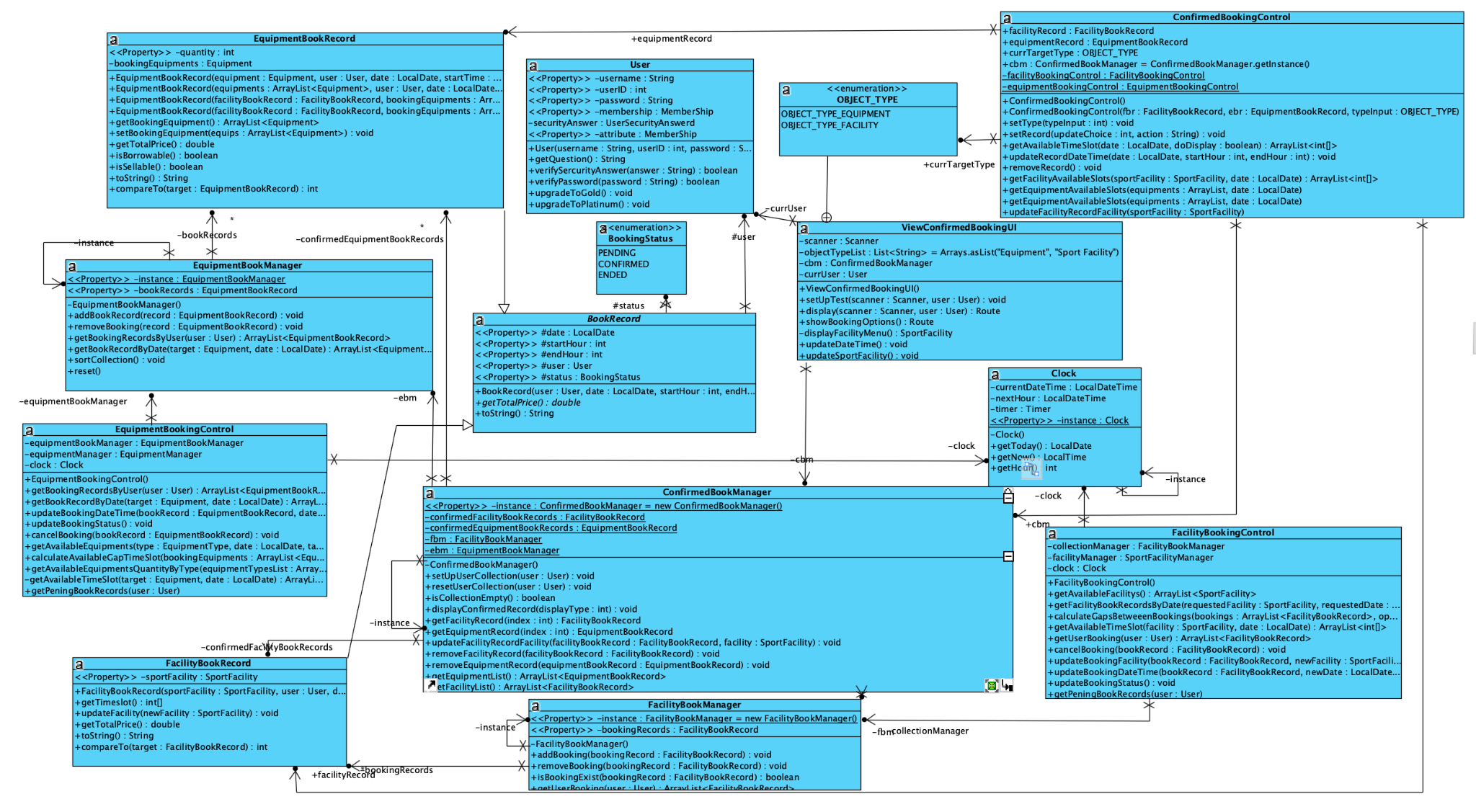
**Sell Equipment**



**View Current Booking Record**



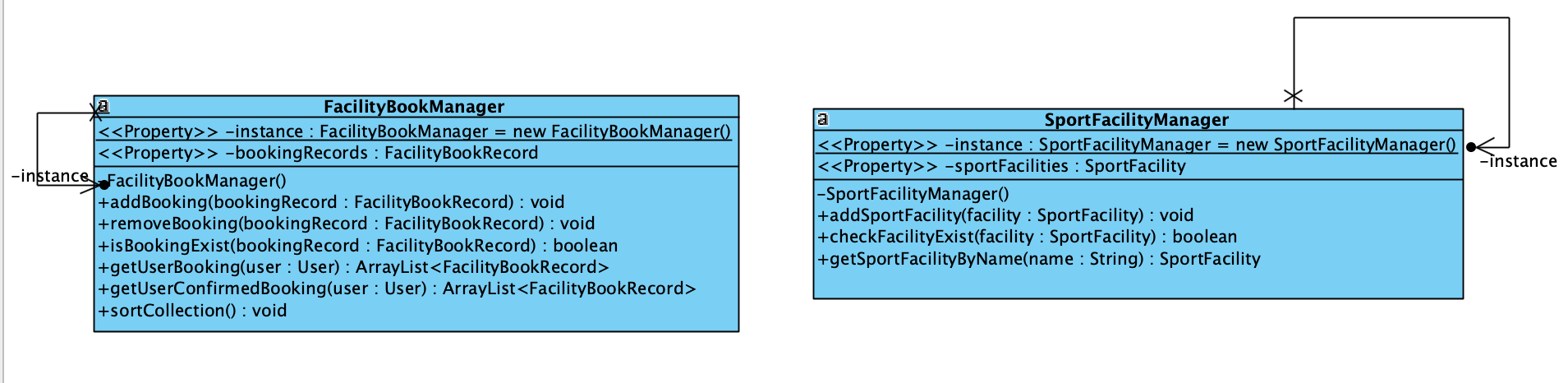
**View Confirmed Booking Record**



The class diagrams are separated in function. The order of the related-function classes are: Book Facility, Book Equipment, Sell Equipment, View Current Booking Record, and View Confirmed Booking Record. Each function diagram contains some of the basic classes. UI class for displaying certain function UI format, confirmedBookingUI for confirming booking record UI. Control for related control of the function needed when executing.

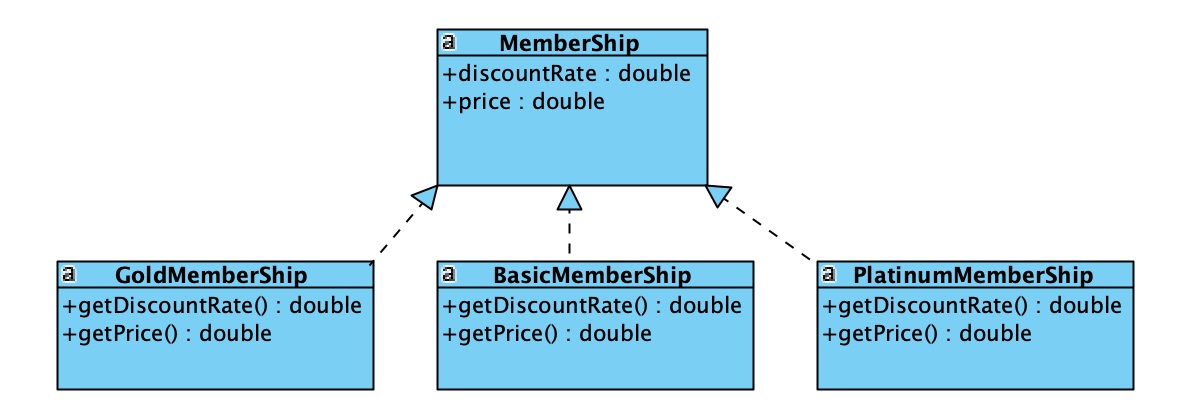
# 4. Design Pattern

## 4.1. Singleton Pattern



**Singleton pattern** is implemented in some classes to prevent the creation of multiple classes during the run of the program. For example, the singleton pattern is implemented in the manager related classes. The reason why singleton pattern is implemented as the role of manager class is to store related object classes, such as equipment book record is stored in BookEquipmentManager class. If the duplication of BookEquipmentManager is allowed, it may cause error, data inaccuracy and asymmetry. Since the singleton pattern only allows one class existence, the pattern eliminates the possibility of multiple same type of class creation.

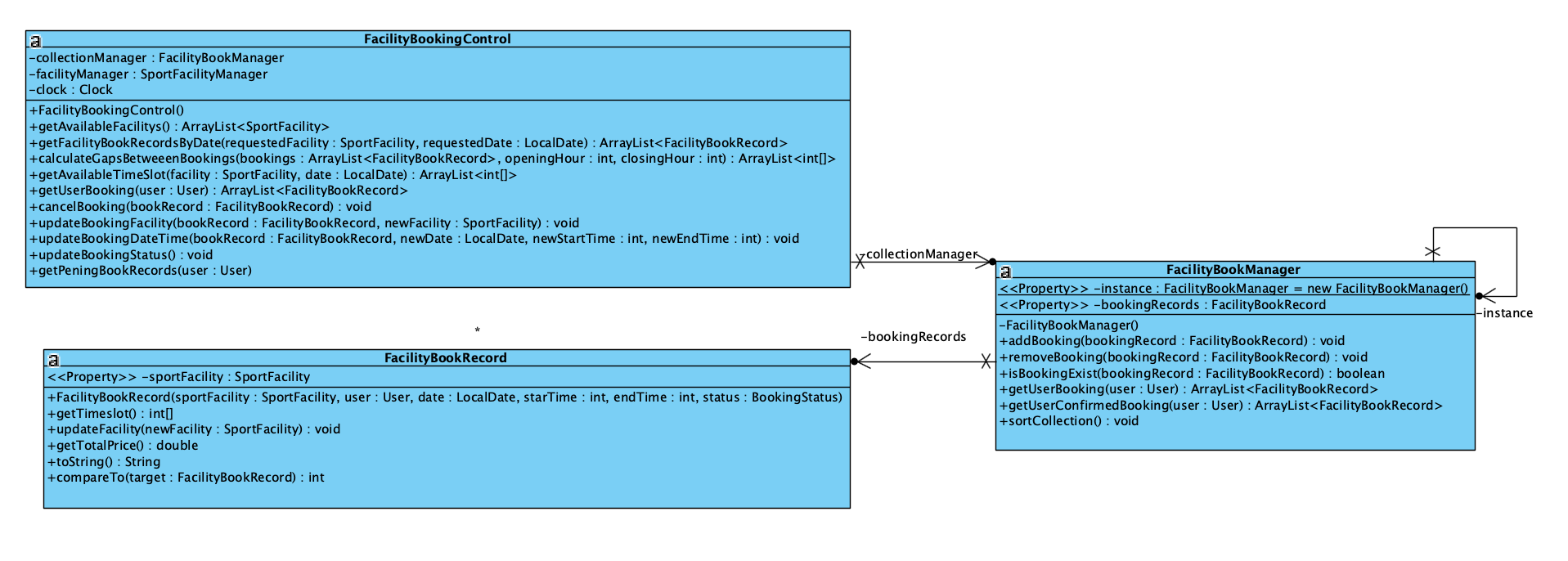
## 4.2. State Pattern



The implementation of user’s membership utilized the state pattern. There is a membership interface implemented in the program. Different levels of membership, Basic, Gold and Platinum implement the membership interface. By utilizing the state pattern, calculation of payment can be done easily by the object class behaviors. The behaviors of the program can be easily modified by simply changing the class.

# 5. Design Principle

## 5.1. Single Responsibility Principle

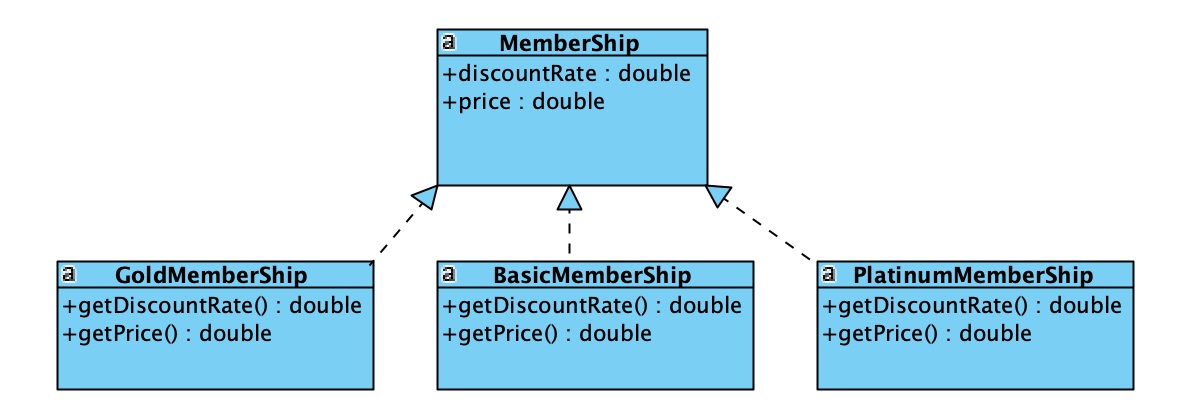


**Single Responsibility Principle** states that each class should have a single responsibility at most when designing class and its structure. In the program, different types of data handle multiple classes. For example, there are many different classes that are related to handling facility book records. FacilityBookManager for holding facility book records, FacilityBookControl for the related control to the manager, and FacilityBookUI for the UI related functions.

## 5.2. Open Closed Principle

There are two types of booking records in the system, equipment and facility. Both equipment book record and facility book record class are extended from the book record super class. Open Closed Principle encourages the program classes to be open for extension and be closed modification. The Class design follows the Open Closed Principle as when a correlated book record must be modified, it can be easily done by modifying to the type of the booking record.

## 5.3. Dependency Inversion Principle



The **interface membership** exists in the program and the following classes: BasicMembership, GoldMembership and PlatinumMembership have implemented it. The membership has two abstract functions: getDiscountRate() and getPrice() function, and the other class Calculator depends on the membership abstract methods, to do calculation of each membership price for the payment. This follows the DIP as the calculator class depends on the abstraction instead of low-level concrete methods.

# 6. Sequence diagram

## 6.1. Book facility

The sequence of booking a sport facility is initiated by the user choosing to book a specific facility. The process involves input validation for the date and time, followed by availability checks, and finally, the creation of a new booking record.

1. **Input Date**: The system receives the desired booking date from the user, validates it against the current date, and ensures the format is correct.
2. **Check Availability**: The system retrieves and displays the available time slots for the selected facility and date.
3. **Input Time Slot**: The system receives the desired start and end hours, validating them against business hours and the available slots.
4. **Create Booking**: Once all inputs are confirmed valid, a new booking record is created, added to the system, and a confirmation message is displayed.

## 

## 6.2. Book equipment

The sequence of borrowing equipment is initiated immediately after a facility is booked, using the context of the **Facility Book Record**. The core purpose of this sequence is to allow the **User** to select an equipment type associated with the booked sport and specify a quantity to borrow. The process is handled primarily by the **Borrow Equipment UI** component.

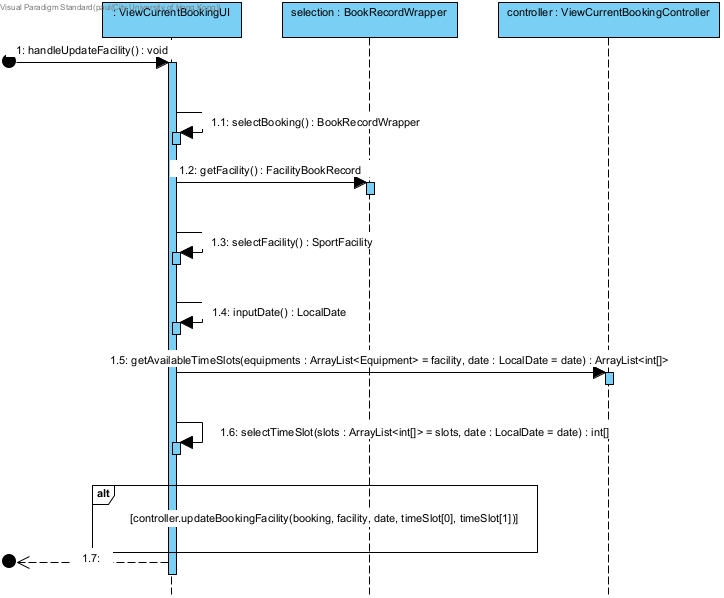


## 6.3. Handling Current Booking

Current booking is the booking that pending for payment and waiting to be confirmed after the payment process

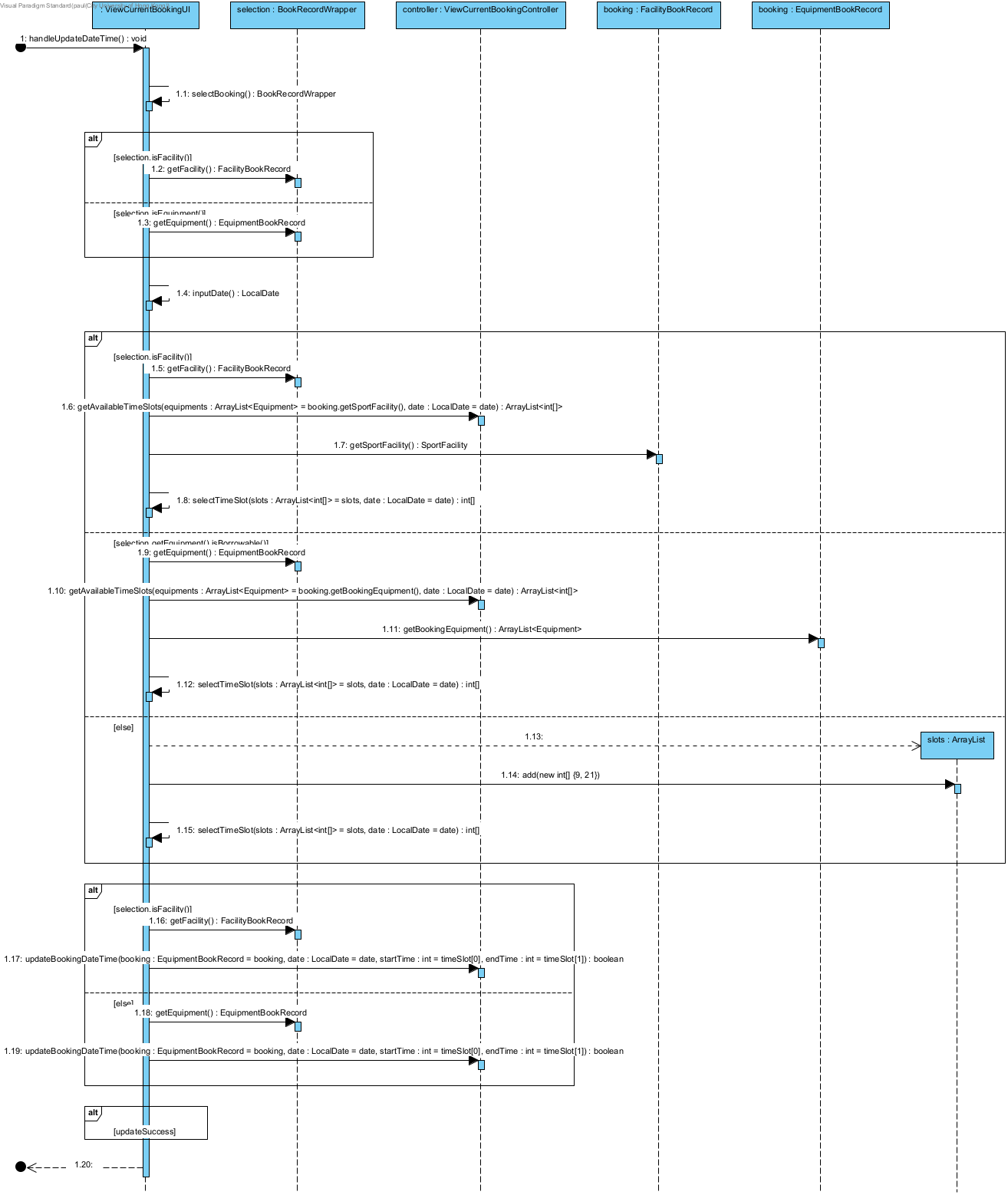
### Update booking facility

This sequence describes the process of a **User** modifying an existing **Facility Booking Record** (changing the facility, date, or time). The process includes selecting an existing booking, choosing a new facility, and then selecting a new valid time slot for that facility and date. The core logic is handled by a **Controller** component.



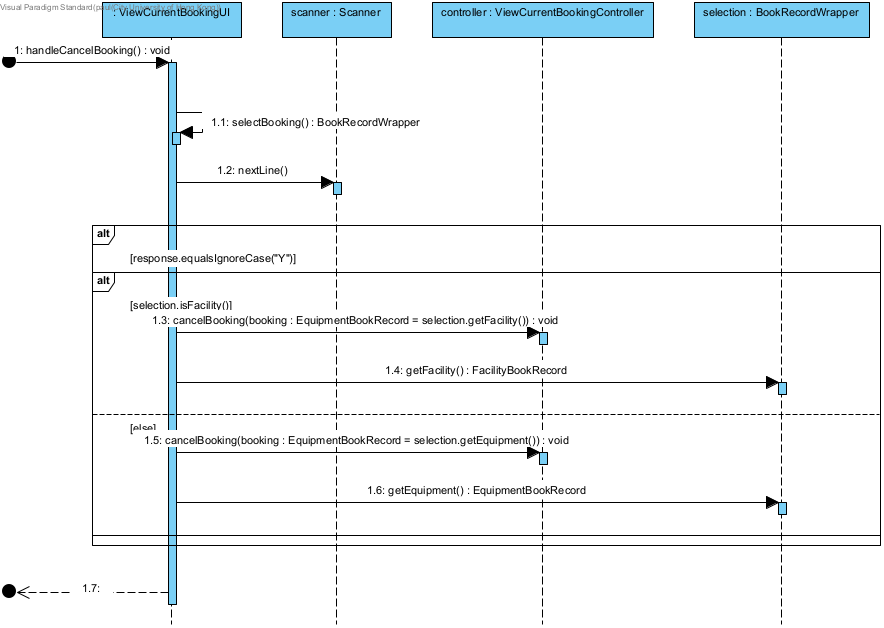
### Update booking date time

This sequence describes the generalized process of a User modifying the date and/or time of an existing booking, which could be either a Facility Booking or an Equipment Booking. The process includes selecting an existing record, validating the ability to modify it (cannot modify today's bookings), and then selecting a new date and time slot based on availability. The core transaction is handled by a Controller component.



### Cancel Booking

This sequence describes the process of a User cancelling an existing booking, which could be either a Facility Booking or an Equipment Booking. The process includes: select a booking, confirm the cancellation, and then delegate the actual cancellation transaction to a Controller component.

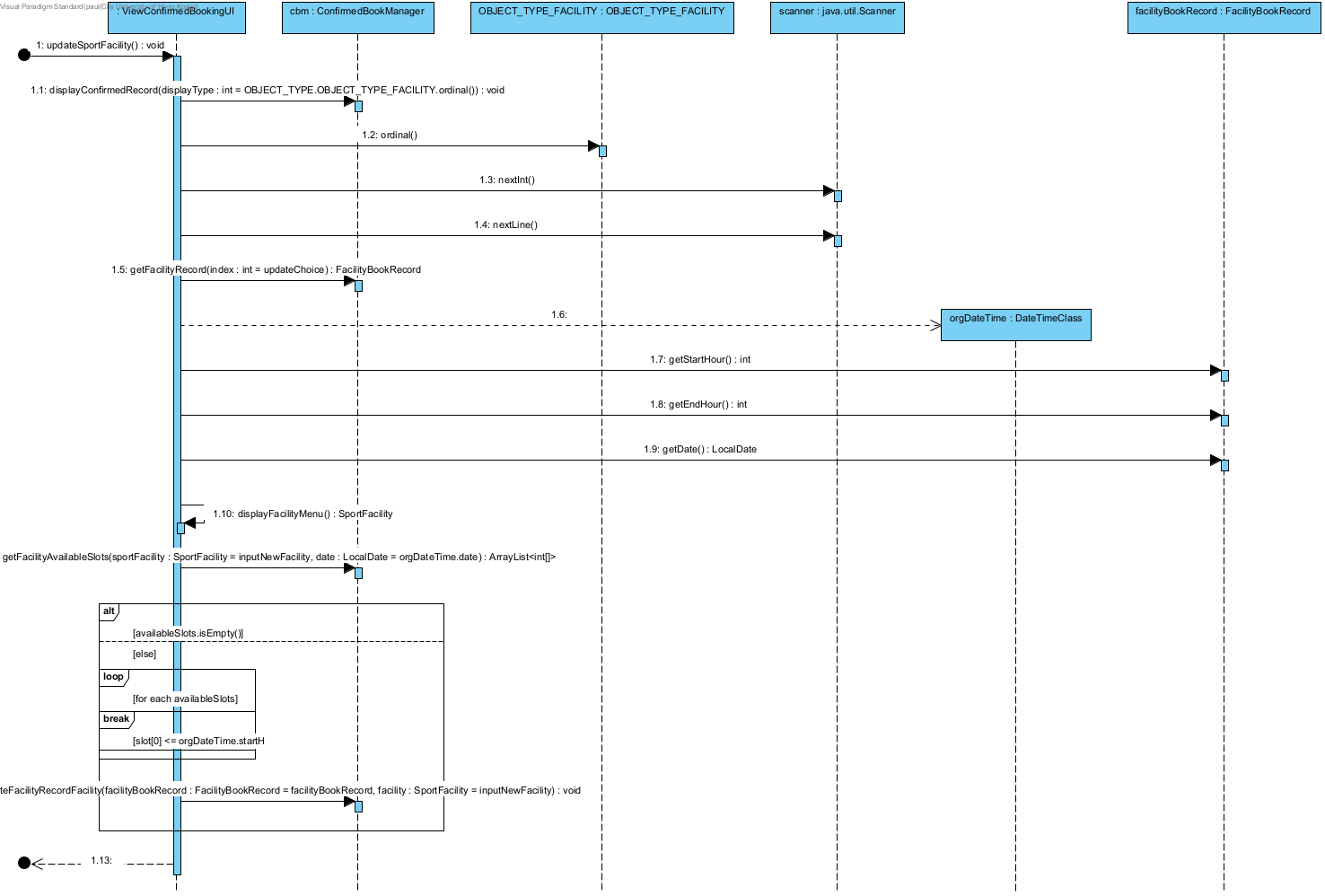


## 6.4. Handling confirmed booking

Confirmed booking is the booking paid but not yet started

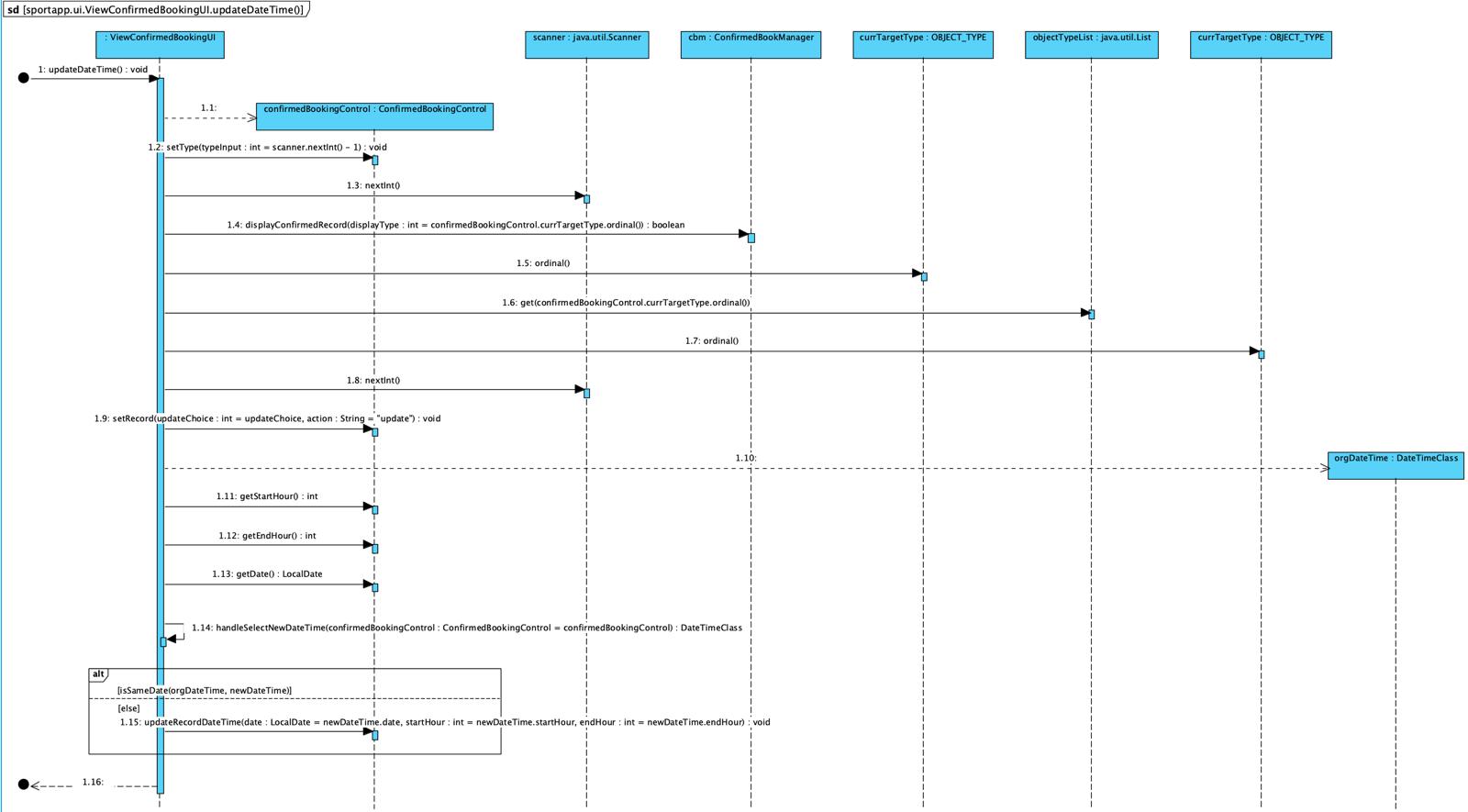
### Update Booking facility

This sequence describes the process of updating **only the Sport Facility** associated with an existing **Facility Book Record**, while keeping the original date and time. The flow involves displaying existing bookings, selecting one to update, validating the booking date, selecting a new facility, checking if the **original time slot** is available in the new facility, and finally, executing the update. The core interactions rely on the **Confirmed Booking Manager (CBM)**.



### Update booking date time

This sequence describes the generalized process of a **User modifying the date and/or time** of an existing confirmed booking. The process requires the **User** to specify the **type** of booking (Facility or Equipment), select a specific record, and then input and validate a new date and time slot. The core logic and final update are managed by the **Confirmed Booking Control (CBC)** component.



### Cancel Booking

This sequence describes the detailed process of a **User** cancelling an existing confirmed booking, whether it's a **Facility** or **Equipment** record. The process includes the **selection of the booking type**, the retrieval and display of records by the **Confirmed Booking Manager (CBM)**, and the final transactional logic executed by the **Confirmed Booking Control (CBC)** after user confirmation.

\