HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

EcoBikeRental (EBR)

Subject: ITSS Software Development

Group 03

Trịnh Thu Hải – 2018422

Nguyễn Huy Hoàng - 20184265

Bùi Thanh Tùng - 20184324

*Hanoi, November, 2021*

**Table of Contents**

[**Introduction**](#_heading=h.2et92p0) **4**

[Objective](#_heading=h.tyjcwt) 5

[Scope](#_heading=h.3dy6vkm) 5

[Glossary](#_heading=h.1t3h5sf) 6

[References](#_heading=h.4d34og8) 9

[Trang, N. T. (2019, September). Problem Statement: Eco Bike Rental. Retrieved from: https://www.dropbox.com/sh/2llptvvm9atklen/AADGszPxE-JdkRrsPnaRYeO6a/CapstoneProject?dl=0&preview=EcoBikeRental-ProblemStatement-EN.pdf&subfolder\_nav\_tracking=1](#_heading=h.2s8eyo1) 9

[**Overall Description**](#_heading=h.17dp8vu) **9**

[General Overview](#_heading=h.3rdcrjn) 10

[Usecase diagram](#_heading=h.26in1rg) 10

[Usecase “View Bike Info”](#_heading=h.2jxsxqh) 11

[Usecase “Rent Bike”](#_heading=h.z337ya) 12

[Usecase “Return Bike”](#_heading=h.3j2qqm3) 12

[Usecase “Deposit Rent”](#_heading=h.1y810tw) 13

[Usecase “Pay Rent”](#_heading=h.4i7ojhp) 13

[Assumptions/Constraints/Risks](#_heading=h.2xcytpi) 14

[Assumptions](#_heading=h.1ci93xb) 14

[Constraints](#_heading=h.3whwml4) 14

[Risks](#_heading=h.2bn6wsx) 15

[**System Architecture and Architecture Design**](#_heading=h.qsh70q) **16**

[Architectural Patterns](#_heading=h.3as4poj) 16

[Interaction Diagrams](#_heading=h.1pxezwc) 16

[View Bike Info](#_heading=h.49x2ik5) 16

[Rent Bike](#_heading=h.2p2csry) 16

[Return Bike](#_heading=h.fomvxklrfj5w) 18

[Deposit Rent](#_heading=h.3o7alnk) 19

[Pay Rent](#_heading=h.23ckvvd) 19

[Analysis Class Diagrams](#_heading=h.ihv636) 21

[View Dock](#_heading=h.32hioqz) 21

[View Bike Info](#_heading=h.1hmsyys) 22

[View Dock List:](#_heading=h.41mghml) 23

[Rent Bike](#_heading=h.2ag2n62i28i9) 24

[Return Bike](#_heading=h.2grqrue) 24

[Pay Rent](#_heading=h.vx1227) 25

[Unified Analysis Class Diagram](#_heading=h.3fwokq0) 26

[Security Software Architecture](#_heading=h.1v1yuxt) 26

[**Detailed Design**](#_heading=h.4f1mdlm) **26**

[User Interface Design](#_heading=h.2u6wntf) 28

[Screen Configuration Standardization](#_heading=h.19c6y18) 28

[Screen Transition Diagrams](#_heading=h.3tbugp1) 29

[Screen Specifications](#_heading=h.28h4qwu) 29

[Home Screen](#_heading=h.nmf14n) 29

[Dock Details Screen](#_heading=h.46r0co2) 31

[Rent Bike Screen](#_heading=h.4lku3n1c4i9u) 32

[View Rent Bike Screen](#_heading=h.aky70ch2xqi3) 32

[Return bike Screen](#_heading=h.206ipza) 33

[Invoice Screen](#_heading=h.3l18frh) 34

[Payment Screen](#_heading=h.98wnqo2sqli) 35

[Result Screen](#_heading=h.emt4l5szidbg) 36

[Error Screen](#_heading=h.87im6oiq4fai) 37

[Data Modeling](#_heading=h.2zbgiuw) 37

[Conceptual Data Modeling](#_heading=h.1egqt2p) 37

[Database Design](#_heading=h.3ygebqi) 37

[Database Management Systems](#_heading=h.2dlolyb) 37

[Logical Data Model](#_heading=h.sqyw64) 37

[Physical Data Model](#_heading=h.3cqmetx) 37

[Non-Database Management System Files](#_heading=h.1rvwp1q) 46

[Class Design](#_heading=h.4bvk7pj) 46

[General Class Diagram](#_heading=h.2r0uhxc) 46

[Class Diagrams](#_heading=h.1664s55) 47

[Class Diagram for Package “Controller”](#_heading=h.3q5sasy) 47

[Class Diagram for Package “View-handler”](#_heading=h.25b2l0r) 47

[Class Diagram for Package “Entity”](#_heading=h.kgcv8k) 48

[Class Diagram for Package “Exception”](#_heading=h.34g0dwd) 49

[Class Diagram for Subsystem “Interbank”](#_heading=h.63bces68vtf3) 49

[Class Diagram for Package “Utils”](#_heading=h.evm2dchme718) 50

[Class Design](#_heading=h.1jlao46) 50

[Class “App”](#_heading=h.ujzsk1c9zd43) 50

[Class “FXMLScreenHandler”](#_heading=h.vkc9i89qxx5t) 51

[Class “BaseScreenHandler”](#_heading=h.xvir7l) 53

[Class “BaseController”](#_heading=h.3hv69ve) 54

[Class “API”](#_heading=h.1x0gk37) 55

[Class “Utils”](#_heading=h.4h042r0) 56

[Class “MyMap”](#_heading=h.2w5ecyt) 57

[Interface “InterbankInterface”](#_heading=h.1baon6m) 59

[Class “InterbankSubsystemController”](#_heading=h.3vac5uf) 60

[Class “InterbankBoundary”](#_heading=h.2afmg28) 61

[Class “InterbankSubsystem”](#_heading=h.pkwqa1) 62

[Class “RentBikeController”](#_heading=h.39kk8xu) 64

[Class “CreditCard”](#_heading=h.1opuj5n) 65

[Class “Invoice”](#_heading=h.48pi1tg) 66

[Class “PaymentTransaction”](#_heading=h.2nusc19) 67

[Class “ReturnBikeController”](#_heading=h.1302m92) 68

[Class “InvoiceScreen”](#_heading=h.3mzq4wv) 69

[Class “ViewRentScreen”](#_heading=h.2250f4o) 70

[Class “PaymentScreen”](#_heading=h.haapch) 72

[Class “ViewStationListController”](#_heading=h.319y80a) 73

[Class “ViewStationInfoController”](#_heading=h.1gf8i83) 74

[Class “SearchStationController”](#_heading=h.40ew0vw) 75

[Class “ViewBikeInfoController”](#_heading=h.2fk6b3p) 76

[Class “ViewStationListScreen”](#_heading=h.upglbi) 77

[Class “ViewStationInfoScreen”](#_heading=h.iocme24ppfrf) 78

[Class “ViewBikeInfoScreen”](#_heading=h.3ep43zb) 79

[Class “Bike”](#_heading=h.qt0jpsz8s4xg) 81

[Class “Station”](#_heading=h.1tuee74) 84

[Class “StationList”](#_heading=h.4du1wux) 87

[**Design Considerations**](#_heading=h.36ei31r) **89**

[Goals and Guidelines](#_heading=h.1ljsd9k) 89

[Architectural Strategies](#_heading=h.45jfvxd) 89

[Coupling and Cohesion](#_heading=h.2koq656) 90

[Design Principles](#_heading=h.zu0gcz) 90

[Design Patterns](#_heading=h.3jtnz0s) 90

**List of Figures**

No table of figures entries found.

**List of Tables**

No table of figures entries found.

# Introduction

## Objective

This Software Design Document (SDD) is created to describe the technical details of the bike management system, intended to supplement maintainers or future innovators of the high level design, as well as implementation of the current system.

## Scope

The main target of this product is to automatically manage bikes in stations,as well as their customer interaction. The system will make use of a scanner for the purpose of recognizing barcodes on each bike, and from that information handles the business process, as well as correctly prompt and record different fees for each bike instance, namely standard bike, twin bike, standard e-bike, and twin e-bike.

Firstly, customers register an account on EcoBikeRental application, fill in appropriate information entries, then allow permissions where needed of the application, and finally set up a working payment method to pay charges, either through linking with inter-bank or an e-wallet. Upon bootstrapping, the position of the user and the location of nearby docking stations will be visible on a map on the screen. By navigating on the map, as well as searching on the dock list, the customer can see available docks there currently, and information on each bike. If the customer finds a suitable bike for his/her need, by scanning the barcode on said bike, the system will proceed with the monetary procedure.

To proceed with a payment, the application will first display the scanned bike information, then ask the customer to provide a payment method. The customer is required to deposit for at least 40% of the value of the bike. Upon agreeing to the payment details, the system will deduct the said amount from the customer's chosen monetary source, and finally open the bike’s lock, and start counting used time.

During the renting period, the customer accesses relevant information of their renting on the system through the application, such as their renting bike type, current renting time, the amount to be paid, and the bike status. To return a bike, the user pushes the bike into an empty docking point of a dock, then closes the lock. The system will automatically calculate the fee, and return any remaining deposit if there is any.

The system handles the validation process, as well as control of bike locks, and display of system information. If there is an error in processing, an error will be prompt to the user..

## Glossary

**A.**

*Actor:*  a role played by a user or any other system that interacts with the subject.

**B.**

*Barcode:* a method of representing data in a visual, machine-readable form. Initially, barcodes represented data by varying the widths and spacings of parallel lines

**C.**

**D.**

*Data*: the quantities, characters, or symbols on which operations are performed by a computer, being stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.

*Docking station*: Station that holds bikes.

*Dock marker*: mark of the map indicating a dock.

**E.**

**F.**

*Flow of events:* A sequence of events or actions within a use case.

**G.**

**H.  
I.**

*Information*: is processed, organised and structured data.

*Input***:** Data that is sent into a system or a program to be processed and executed.

*Inter-*bank: A bank to pay for transactions made by the customer.

**J.**

*Java Runtime Environment (JRE):* A set of software tools, which are included for development Java applications.

**L.**

*Locker:* A system that controls the opening and closing of physical locks and interacts with applications.

**M.**

**N.**

**O.**

**P.**

**Q.**

**R.**

**S.**

*Software Development Kit (SDK)***:** A set of software tools that allow some applications to be created for certain software packages or other similar development platforms

*System*: a hardware or software system, or combination, which has components as its structure and observable share data as its behavior

*Scanner*: A function of the software that takes barcode as input and checks bike information to display

*Standard bike*: a kind of bicycle that has 01 saddle, 01 pedal, and 01 rear seat in the back

*Standard e-bike:* a kind of bicycle that looks like a standard bike, but has an integrated electric motor for assisting propulsion and the rental fee costs 1.5 times that of a standard bike.

**T.**

*Twin bike:* a kind of bicycle that has 02 saddle, 02 pedal, and 01 rear seat with no integrated electric motor with rental fee costs 1.5 times that of a standard bike

**U.**

*Use case*: A sequence of actions a system performs that yields an observable result of value to a particular actor.

**V.**

**W.**

**X.**

**Y.**

**Z**.

## References

Centers for Medicare & Medicaid Services. (n.d.). *System Design Document Template.* Retrieved from Centers for Medicare & Medicaid Services: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx

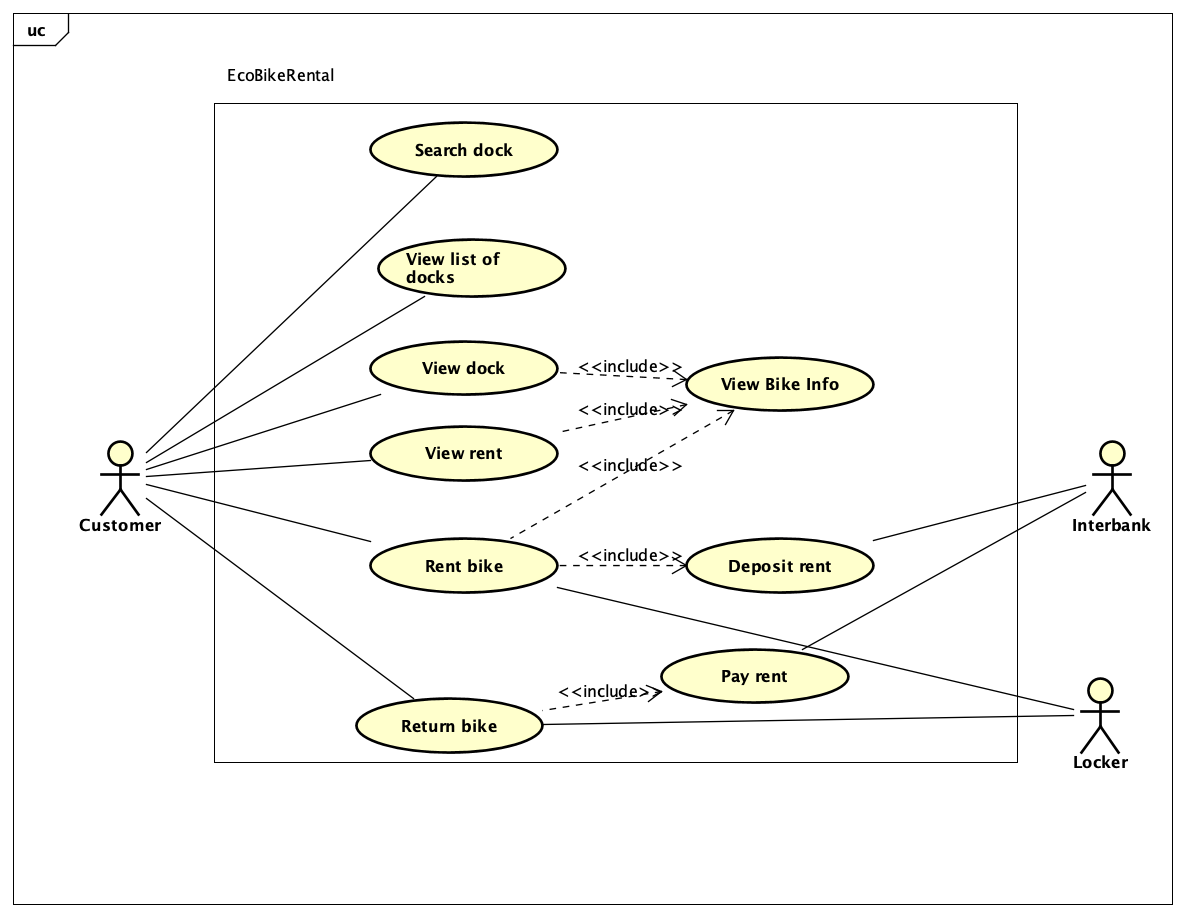
## Trang, N. T. (2019, September). Problem Statement: Eco Bike Rental. Retrieved from: https://www.dropbox.com/sh/2llptvvm9atklen/AADGszPxE-JdkRrsPnaRYeO6a/CapstoneProject?dl=0&preview=EcoBikeRental-ProblemStatement-EN.pdf&subfolder\_nav\_tracking=1

# Overall Description

## General Overview

The system is built with Java SDK to be used on multiple devices. The software will follow the client-server approach, where users will see information about bikes and docks on their devices. The requests will be sent to the server, where it will be proceeded and then sent back the result/error to the user interface. The system can also interact with outer devices and systems, namely the locker on dock stations for lock/unlocking the bikes, and the interbank API for payment processing.

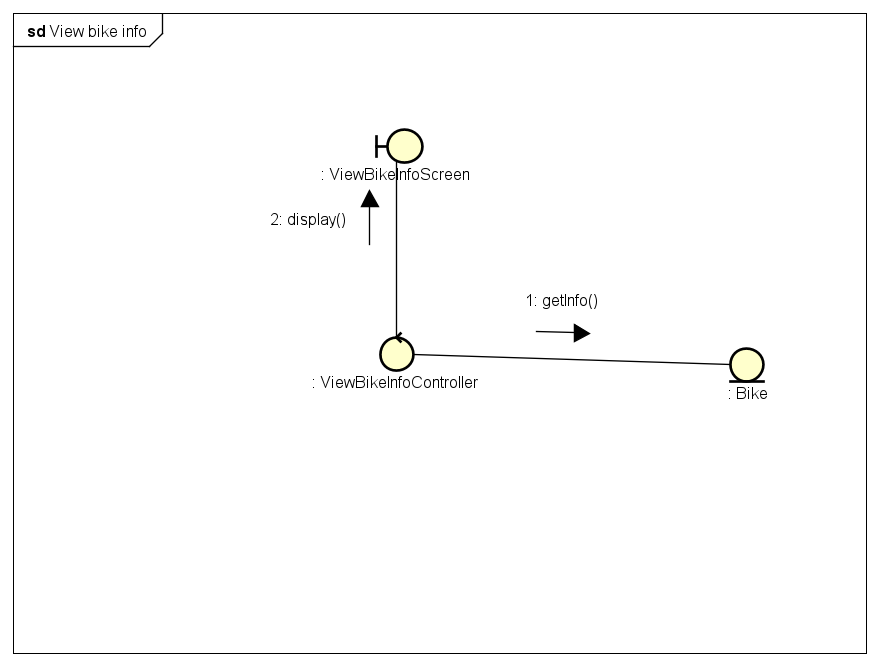
### Use Case diagram



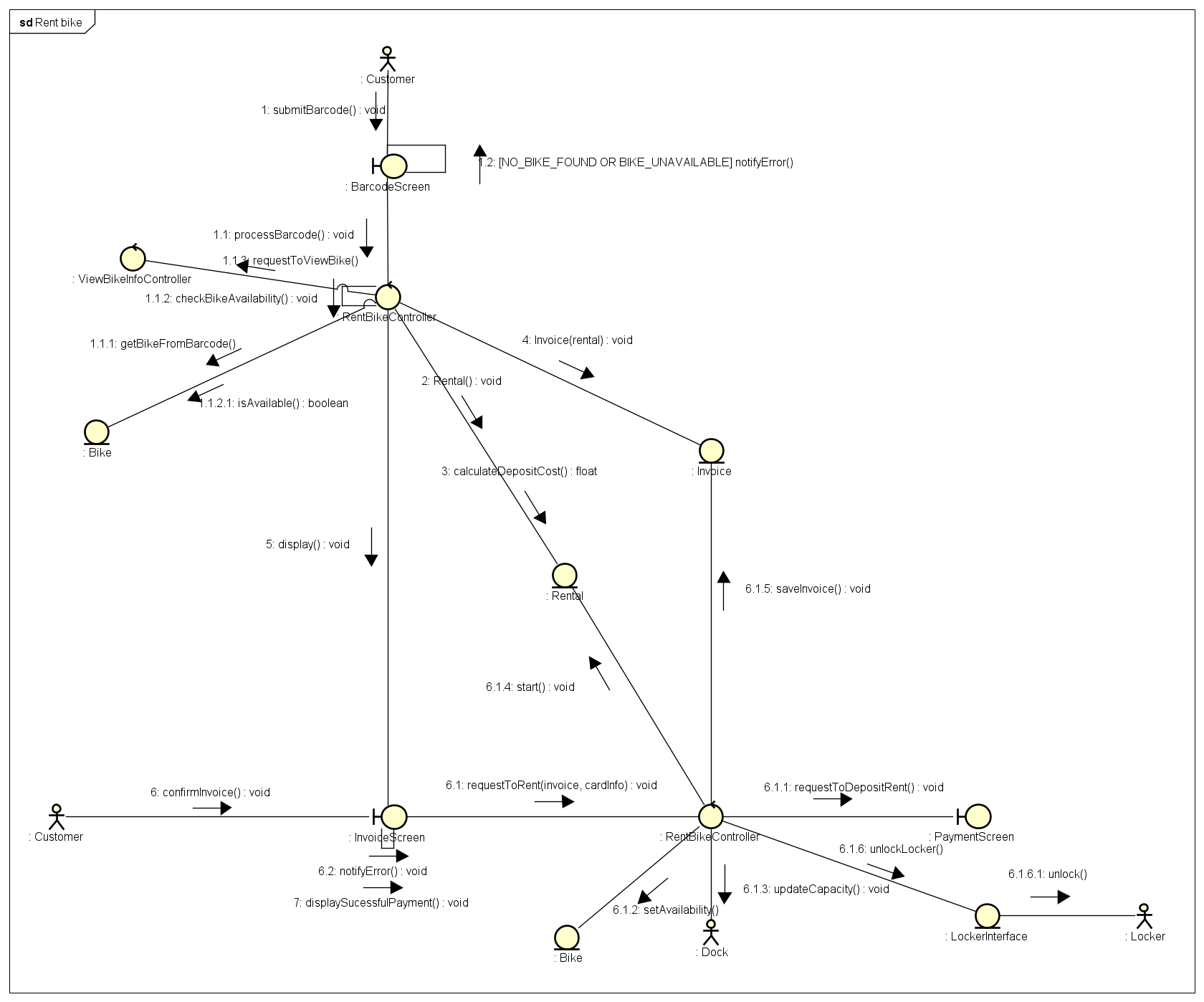
### 

### 

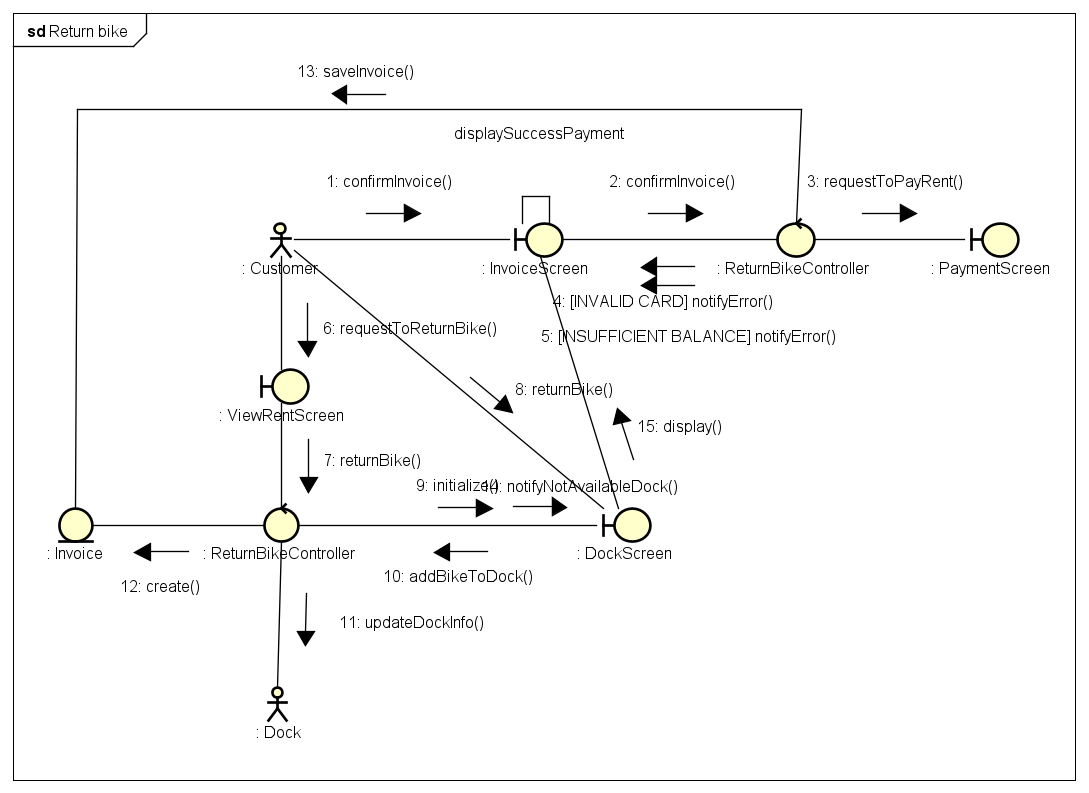
### Use Case “View Bike Info”



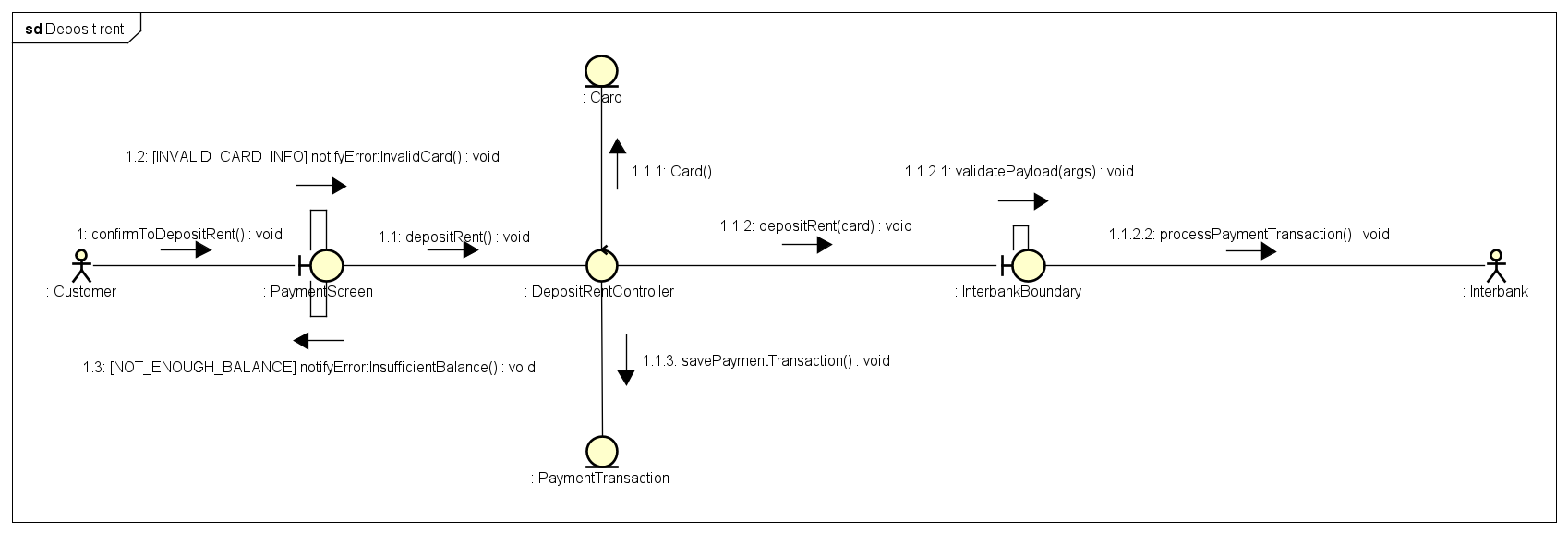
### Use Case “Rent Bike”



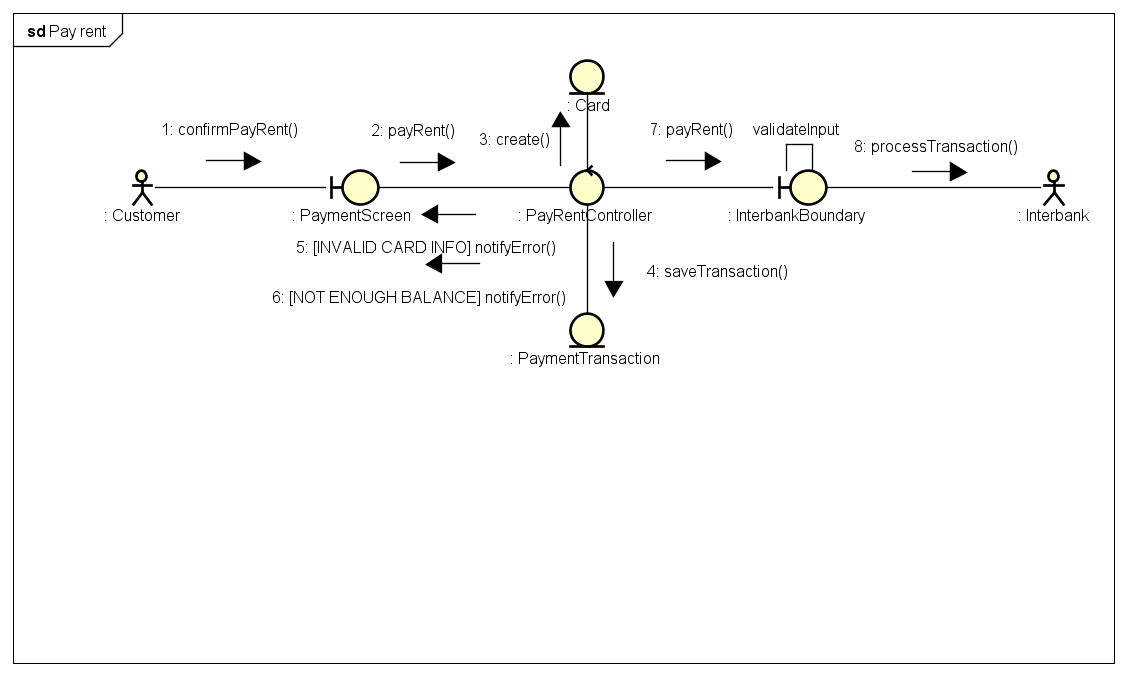
### Use Case “Return Bike”



### Use Case “Deposit Rent”



### Use Case “Pay Rent”



## Assumptions/Constraints/Risks

### Assumptions

The system pre-suppose that the users have internet connection on their devices, as well as a working payment method (through interbank or e-wallet) if they want to actually rent a bike.

The device that the software will be installed in is required to have Java SDK, have at least 50MB of memory. The dock station also requires an internet connection, as well as electricity on proceeding the rent.

The user should know the basic flow of how to rent and return a bike.

Possible changes in the future will mostly be related to the display of the software to the user interface.

### Constraints

*<Describe any global limitations or constraints that have a significant impact on the design of the system’s hardware, software and/or communications, and describe the associated impact. Such constraints may be imposed by any of the following (the list is not exhaustive):*

* *Hardware or software environment:* the software needs to run on JAVA Virtual Machine, ideally takes small enough memory and network bandwidth to be usable on mobile devices.
* *End-user environment:* Users are not required to know too much about underlying system
* *Availability or volatility of resources:* The software should run relatively stable, and calculate the rent time correctly.
* *Standards compliance:* The system should not collect customer information without consent
* *Interoperability requirements:* The system can read user input in ascii
* *Interface/protocol requirements:* The network request should take at most 1 second
* *Licensing requirements: None*
* *Data repository and distribution requirements:* Include business flow and basic instruction on how to use and install the system.
* *Security requirements:* The software should have validation on user input.
* *Memory or other capacity limitations:* Should take less than 50MB to fit on a mobile device.
* *Performance requirements:* The system is quick to navigate.
* *Network communications:* Require regular network communication.

### Risks

There are some possible risks on deploying this system:

* User input

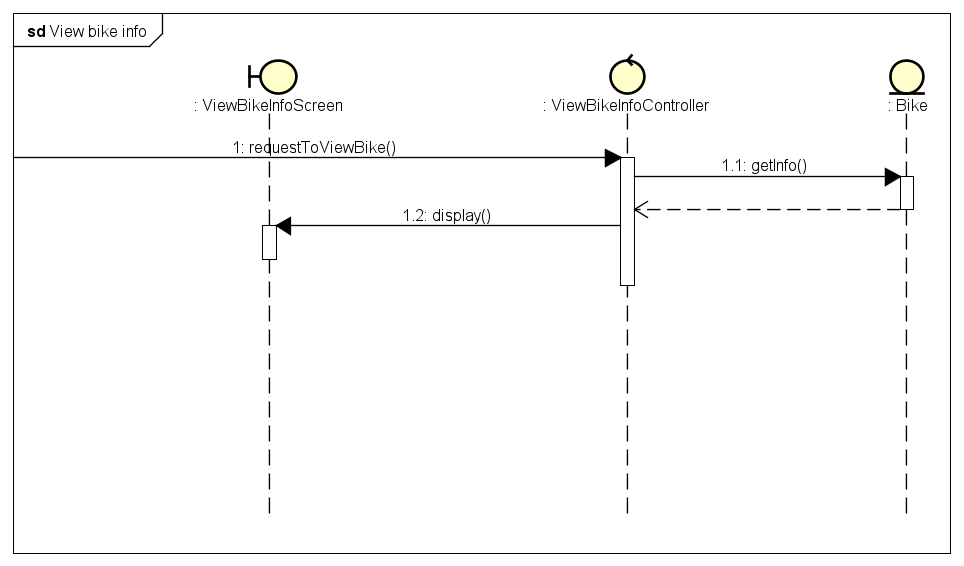
# System Architecture and Architecture Design

## Architectural Patterns

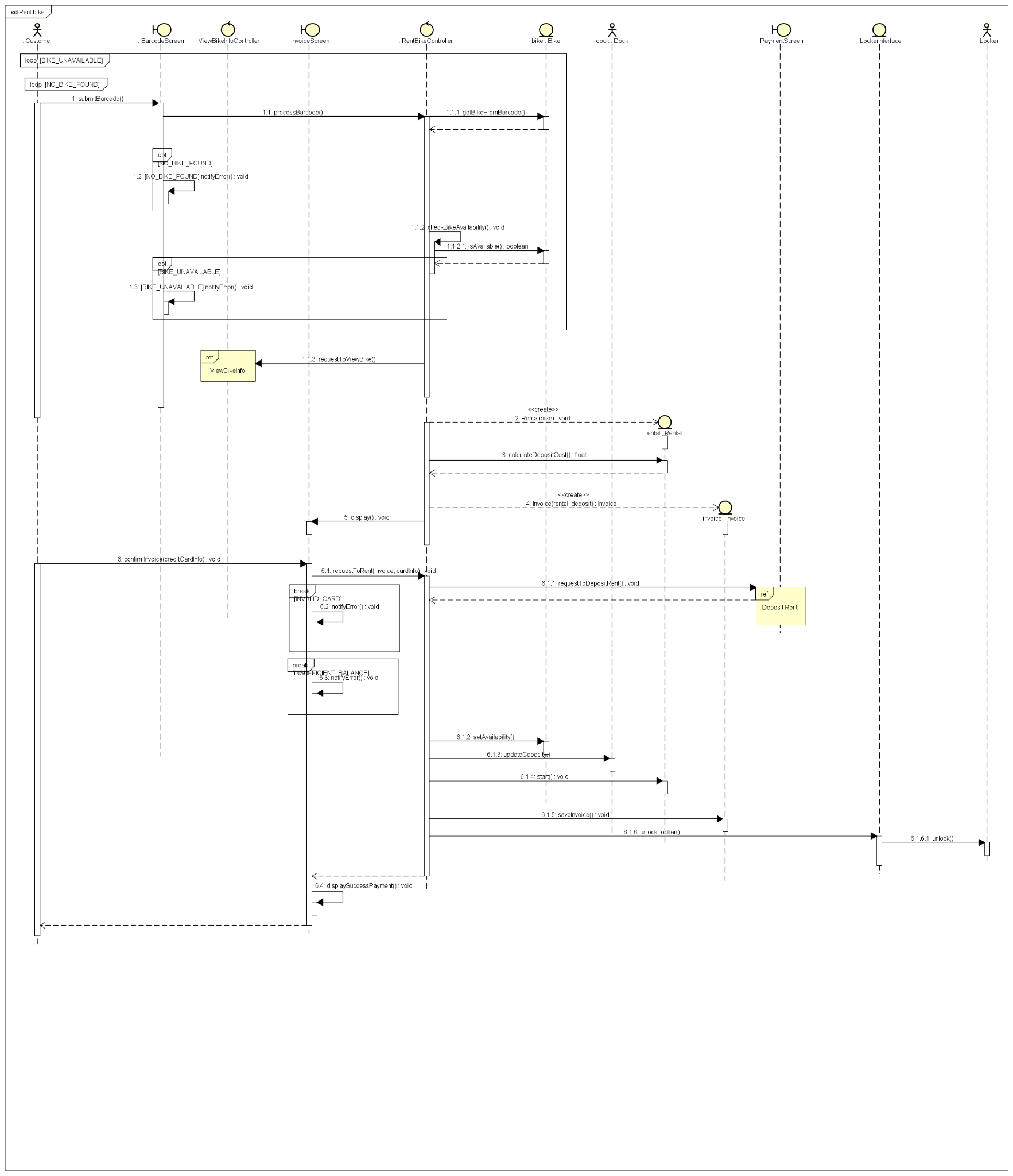
*<Specify and briefly describe the chosen architectural patterns and the reasons why they were chosen>*

## Interaction Diagrams

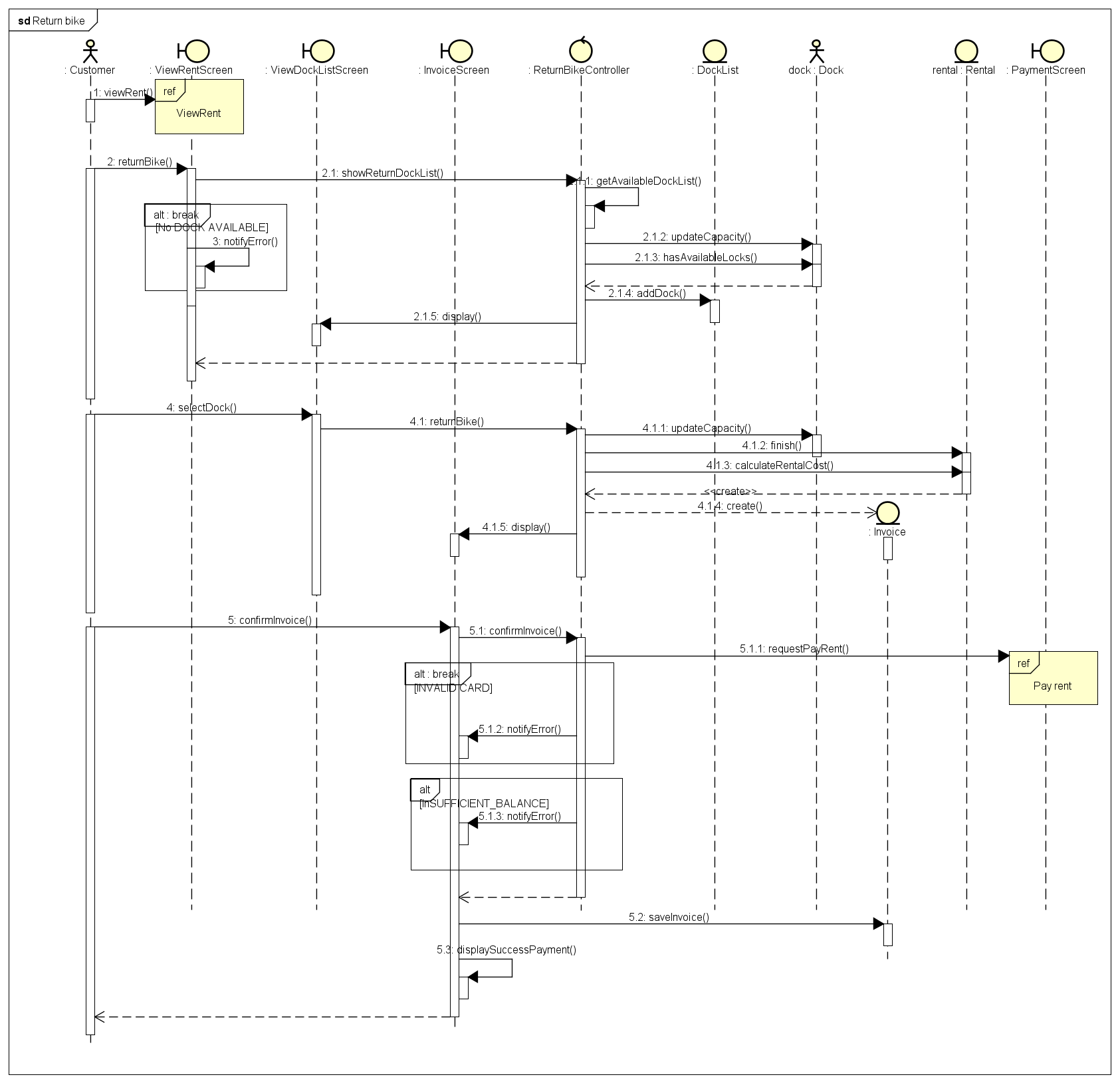
### View Bike Info



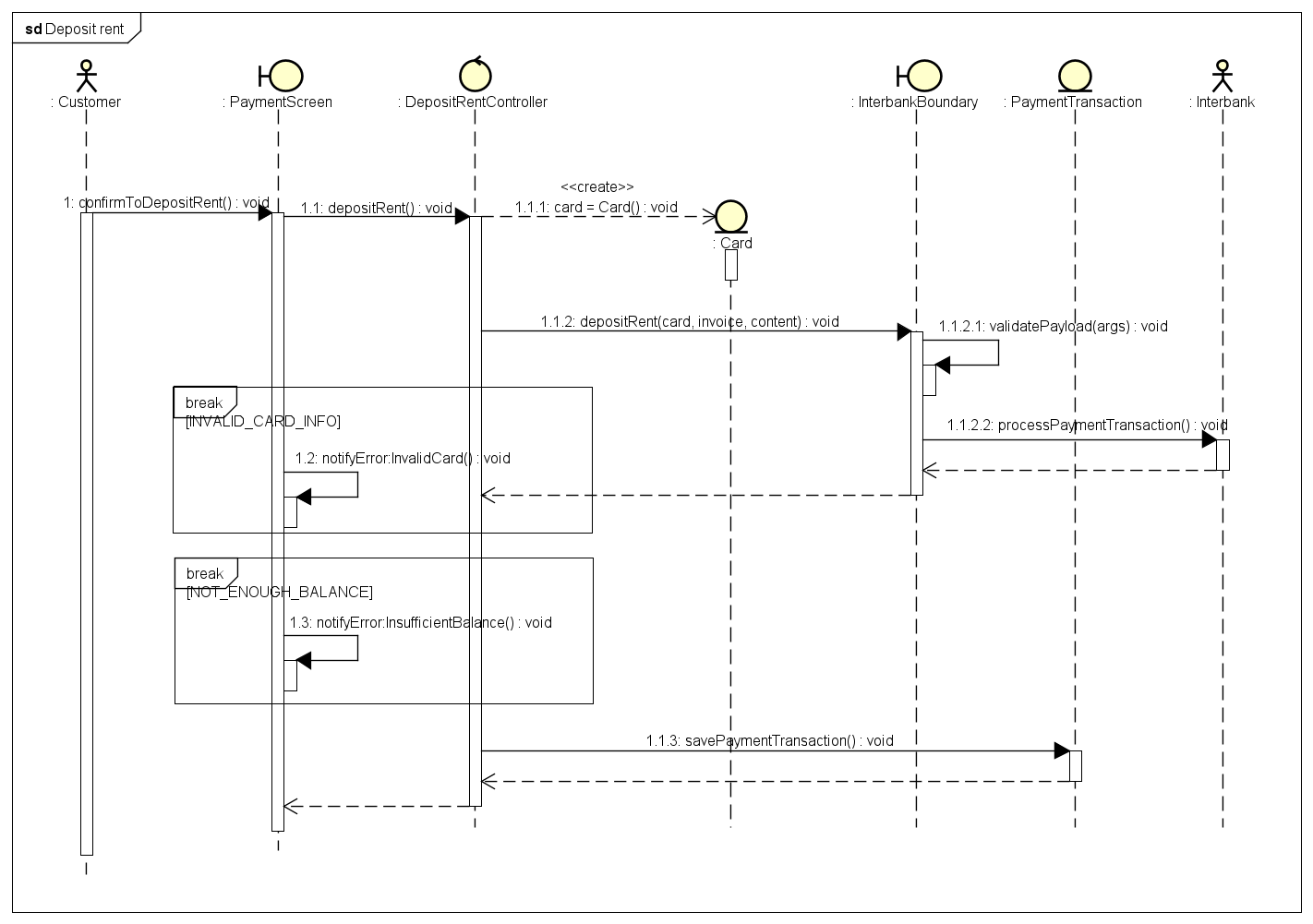
### Rent Bike



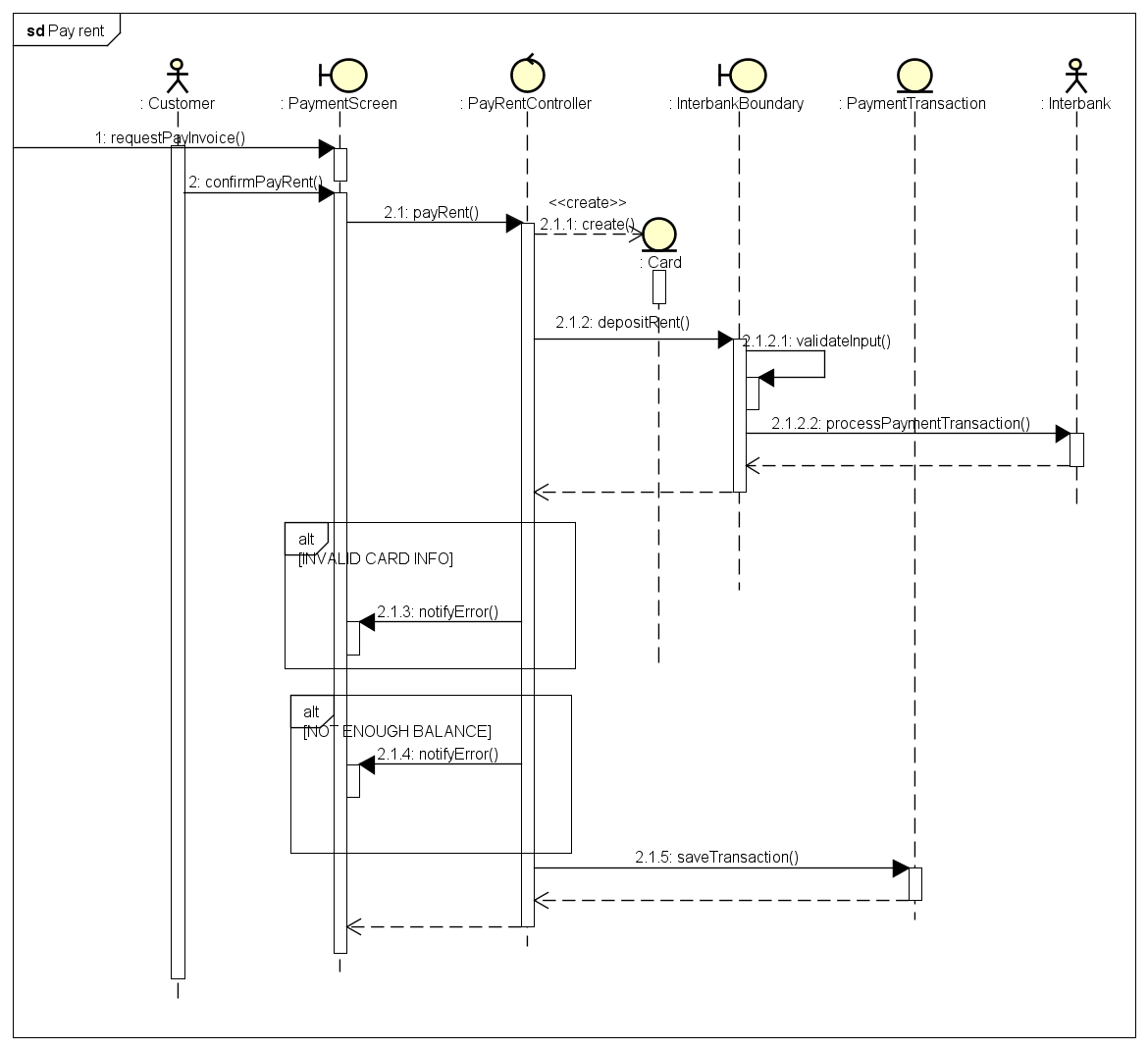
### Return Bike



### Deposit Rent

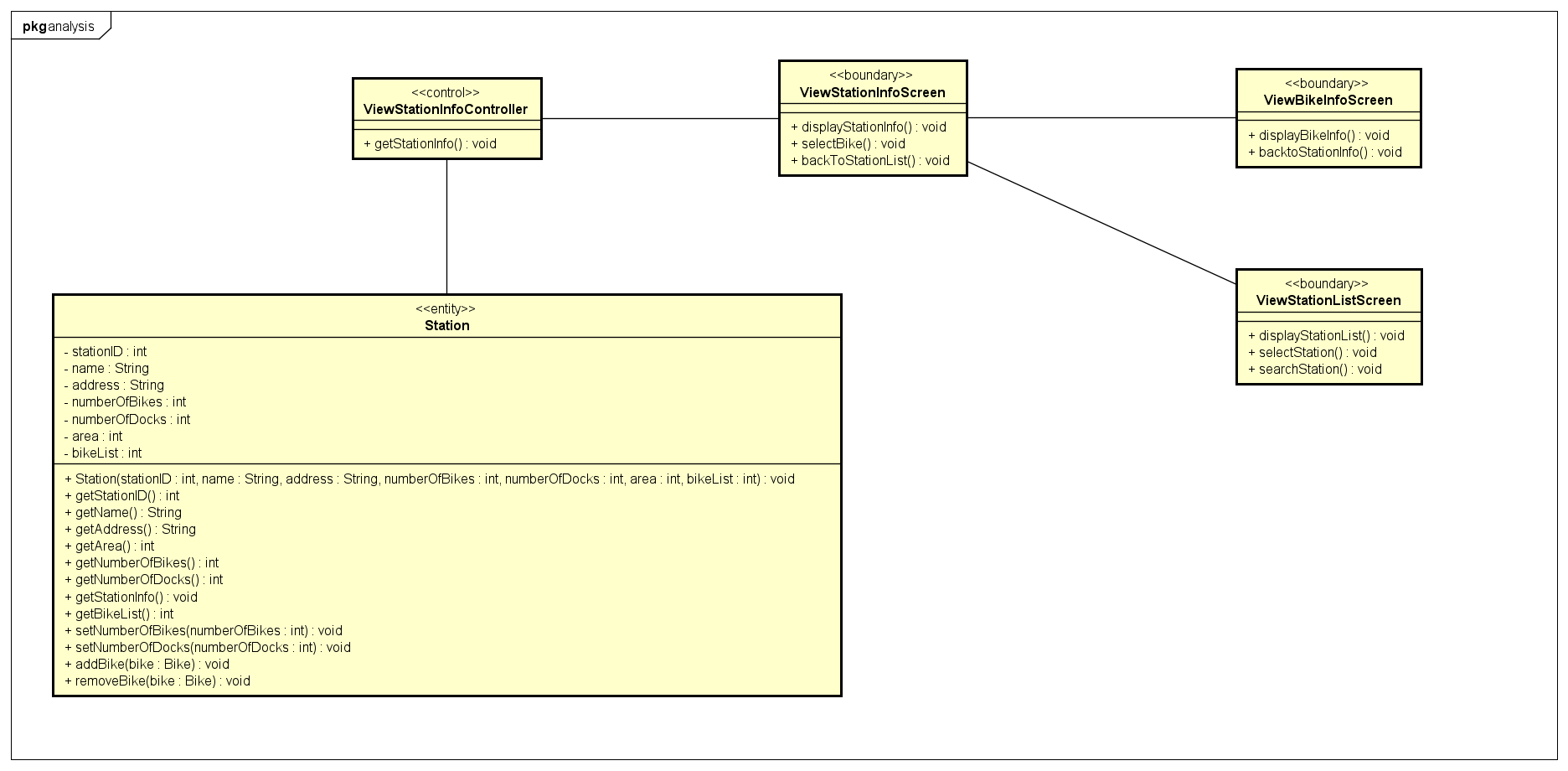


### Pay Rent

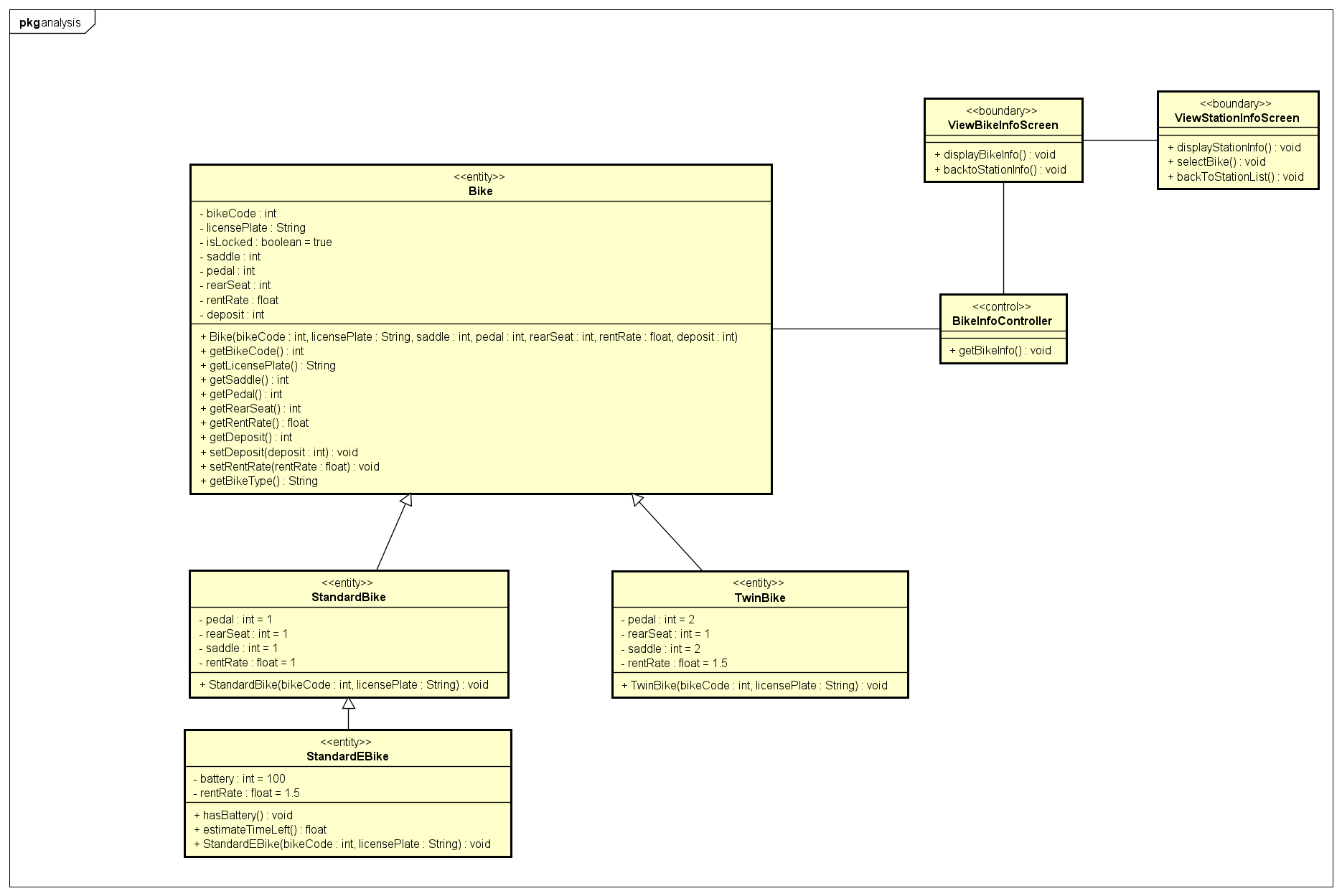


## Analysis Class Diagrams

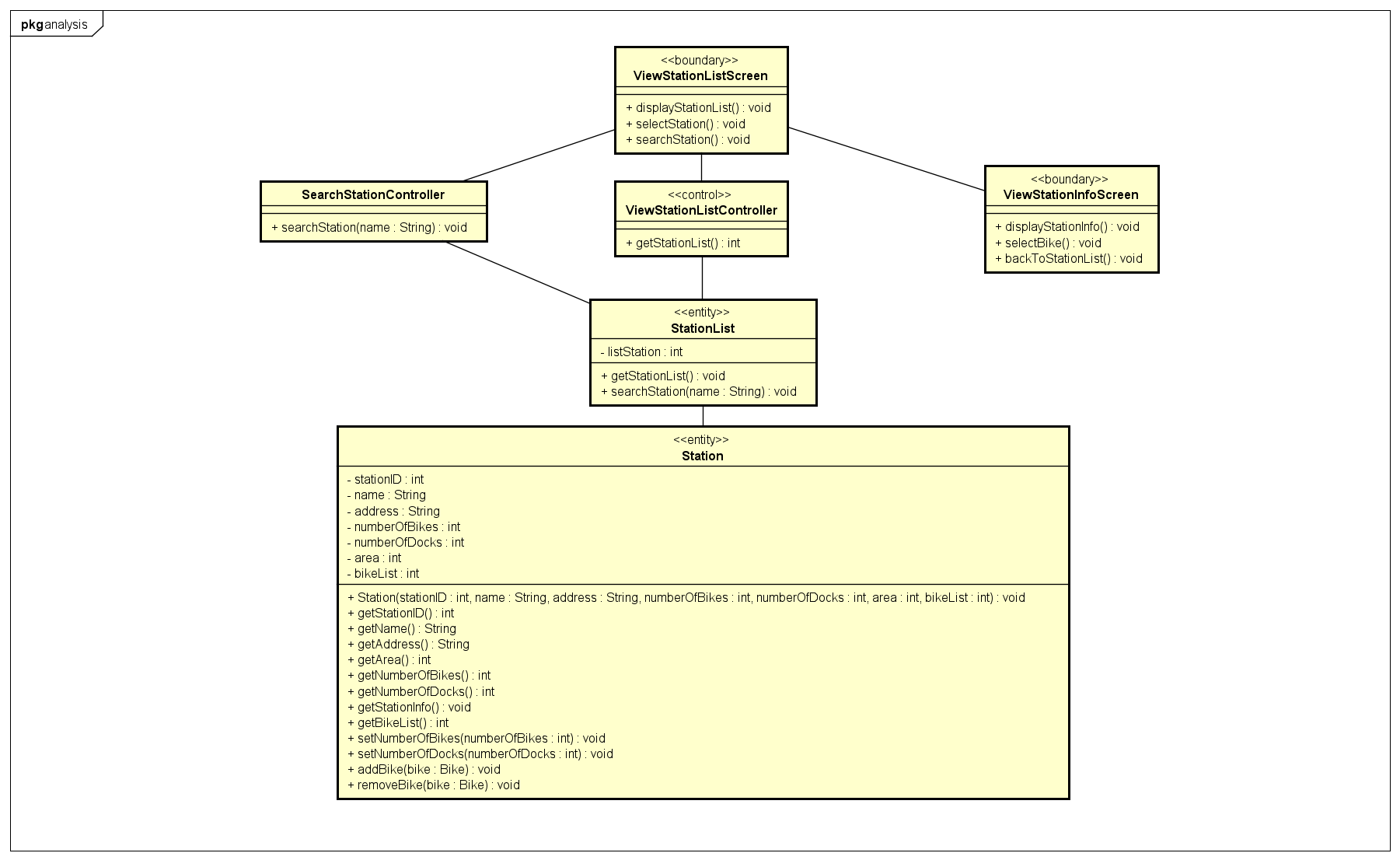
### View Dock



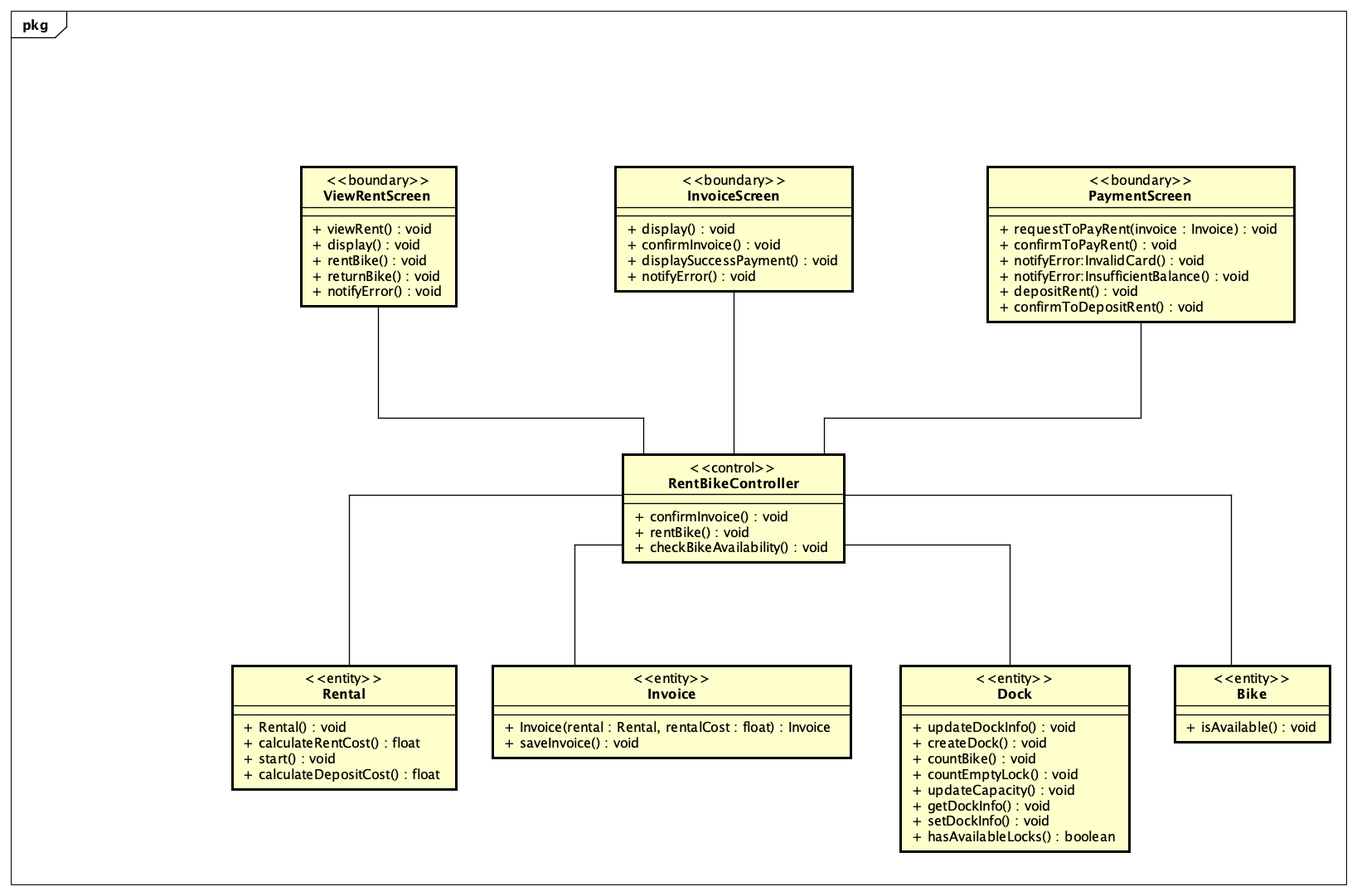
### View Bike Info



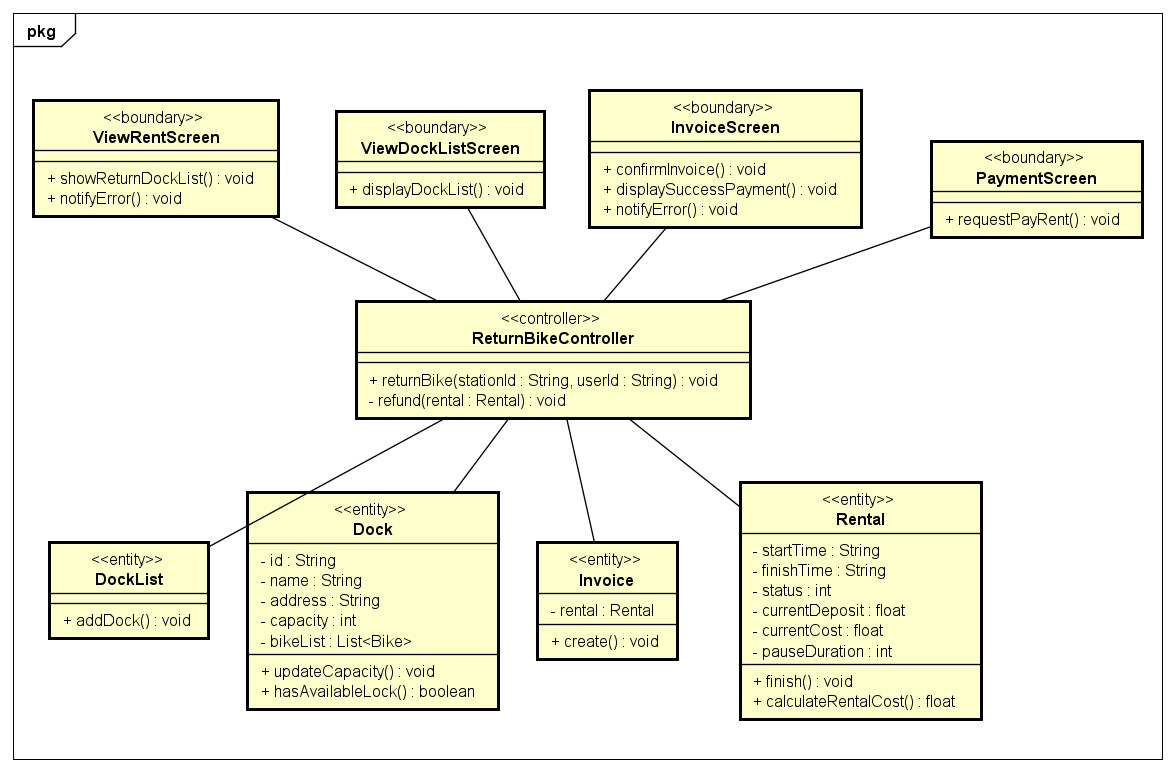
### View Dock List:



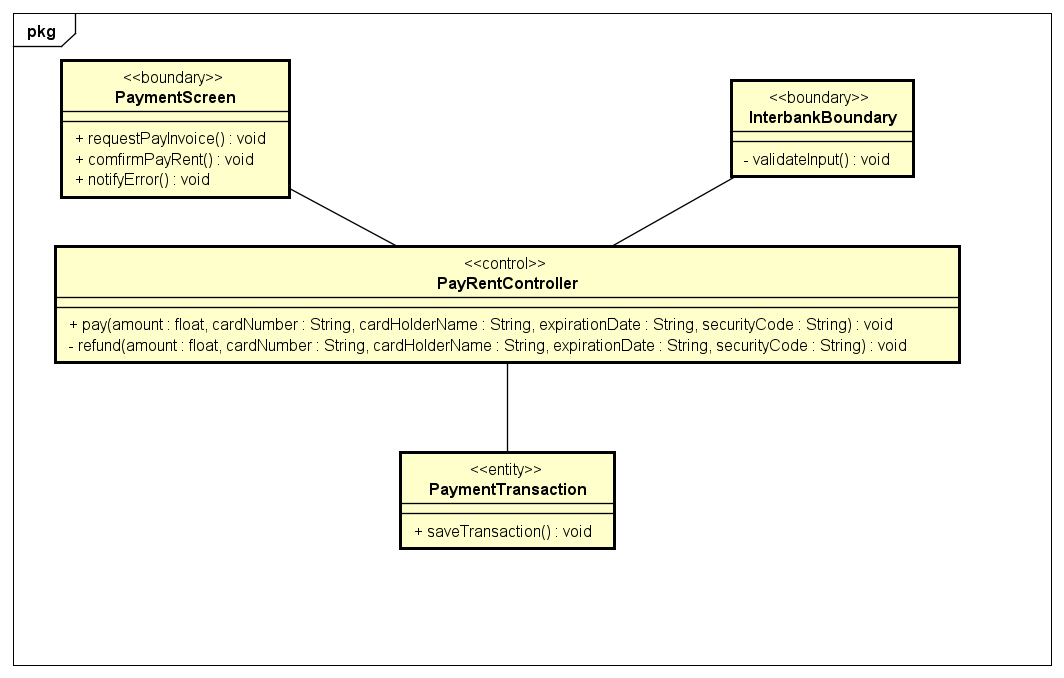
### Rent Bike



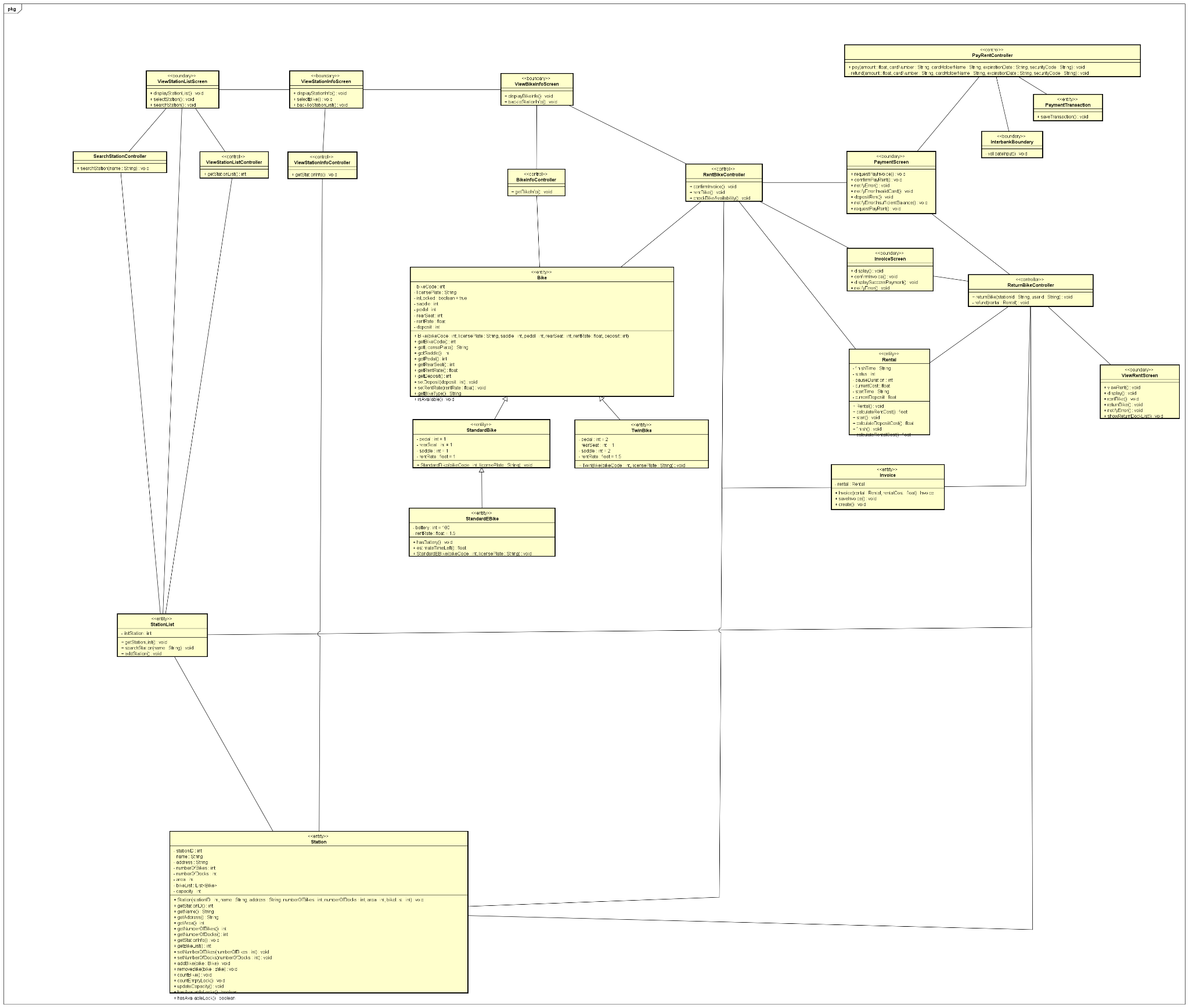
### Return Bike



### Pay Rent



## Unified Analysis Class Diagram



## Security Software Architecture

*<Describe the software components and configuration supporting the security and privacy of the system. Specify the architecture for (1) authentication to validate user identity before allowing access to the system;(2) authorization of users to perform functional activity once logged into the system, (3) encryption protocol to support the business risks and the nature of information, and (4) logging and auditing design, if required.>*

# Detailed Design

## User Interface Design

### Screen Configuration Standardization

**Display**

Number of colors supported: 16,777,216 colors

Resolution: 1366 × 768 pixels

**Screen**

Location of standard buttons: At the bottom (vertically) and in the middle (horizontally) of the frame.

Location of the messages: Starting from the top vertically and in the middle horizontally of the frame down to the bottom.

Display of the screen title: The title is located at the top of the frame on the left-hand.

Consistency in expression of alphanumeric numbers: comma for separator of thousand while strings only consist of characters, digits, commas, dots, spaces, underscores, and hyphen symbol.

**Control**

Size of the text: medium size (mostly 24px). Font: Segoe UI. Color: #000000

Input check process: Should check if it is empty or not. Next, check if the input is in the correct format or not

Sequence of moving the focus: There will be no stack frames. Each screen will be separated except for the error screen. The error screen is considered a popup, as the main screen cannot be operated while the error screen is shown. After the opening screen, the app will start with the first screen (home screen).

Sequences of the system screens:

1. Home screen – Dock list screen

3. Dock details screen – view detail information of dock station

4. Rent bike screen – input the bike barcode

5. View rent bike screen – view details of current renting

6. Return bike screen

7. Invoice screen – view invoice details

8. Payment screen – fill in payment information

9. Result screen - view the result of the payment transaction

10. Error screen - view information of errors occurred

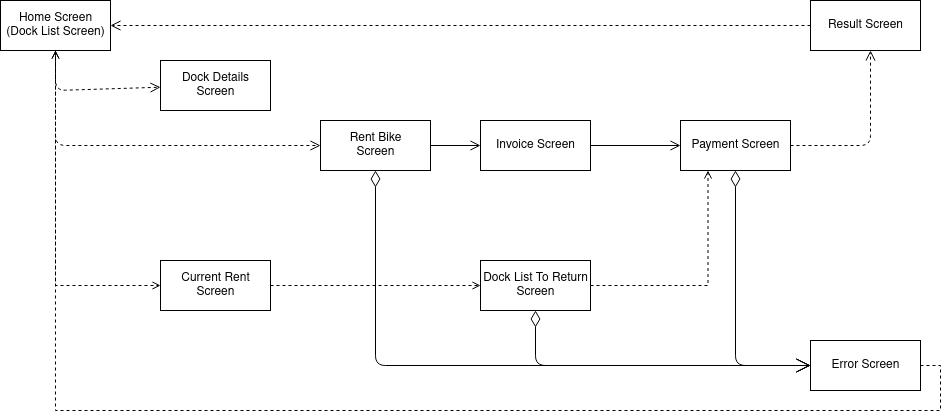
**Direct input from the keyboard**

There will be no shortcuts. There are back buttons to move back to the previous screen. Also, there is the close button “X” located at the title bar to the right to close the screen.

**Error**

A message will be given to notify the users what is the problem.

### Screen Transition Diagrams



### Screen Specifications

#### Home Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Home screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying search dock bar | Initial | Display the search bar to search for docks | |
| Area for displaying docks | Initial | Display docks that match the search | |
| Area for display bike code search | Click | Display the bike code search | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Search dock bar | 120 | String | Black | Left-justified |
| Bike code search | 20 | String | Black | Left-justified |
| Station Name | 30 | String | Black | Left-justified |
| No. of bikes | 5 | Numeral | Black | Center-justified |
| Address | 100 | String | Black | Left-justified |

#### Dock Details Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Dock details screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying dock information | Initial | Display dock information | |
| Area for bike information | Initial | Display information of bikes in dock | |
| Rent button | Click | Go to rent screen of bike with the code | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Dock name | 50 | String | Black | Left-justified |
| Dock address | 100 | String | Black | Left-justified |
| Number of available bikes | 20 | Numeral | Black | Left-justified |
| Number of available slots | 20 | Numeral | Black | Left-justified |
| Bike code | 20 | String | Black | Center-justified |
| Bike type | 50 | String | Black | Center-justified |
| Bike battery | 50 | String | Black | Center-justified |
| Time left | 50 | Numeral | Black | Center-justified |

#### Rent Bike Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Rent bike screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for display bike info | Initial | Display information of bike | |
| Rent bike button | Click | Rent the current bike | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Specs | 100 | String | Black | Left-justified |
| Bar code | 20 | String | Black | Left-justified |
| Liscence plate | 20 | String | Black | Left-justified |
| Current battery | 50 | String | Black | Left-justified |
| Time left | 50 | Numeral | Black | Left-justified |
| Deposit | 50 | String | Black | Left-justified |
| 24-hour pass | 1 | Boolean | Black | Left-justified |

#### View Rent Bike Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | View Rent bike screen | 31/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for display renting information | Initial | Display renting information | |
| Area for display bike image | Initial | Display bike image | |
| Return bike button | Click | Renturn bike | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Bike code | 20 | String | Black | Right-justified |
| Renting time | 50 | String | Black | Right-justified |
| Current fee | 20 | Numeral | Black | Right-justified |
| Current battery | 50 | String | Black | Right-justified |
| Pause time | 20 | String | Black | Right-justified |
| Bike status | 50 | String | Black | Right-justified |

#### Return bike Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Return bike screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying dock | Initial | Display dock information | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Dock name | 50 | String | Black | Right-justified |
| Dock address | 100 | String | Black | Right-justified |
| Number of available slots | 20 | Numeral | Black | Right-justified |

#### Invoice Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Invoice screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for renting information | Initial | Display renting information | |
| Confirm button | Click | Confirm payment | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Rental code | 20 | String | Black | Right-justified |
| Bike code | 20 | String | Black | Right-justified |
| Bike type | 50 | String | Black | Right-justified |
| Customer name | 50 | String | Black | Right-justified |
| Deposit | 20 | Numeral | Black | Right-justified |
| Card number | 50 | String | Black | Right-justified |

#### Payment Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Payment screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying card information | Initial | Display card information | |
| Confirm button | Click | Confirm Card | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Card number | 50 | String | Black | Left- justified |
| Card Holder Name | 100 | String | Black | Left- justified |
| Expiration month | 2 | Numeral | Black | Left- justified |
| Expiration year | 2 | Numeral | Black | Left- justified |
| CVV | 6 | String | Black | Centered |

#### Result Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Result screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for result information | Initial | Display result of renting | |
| OK button | Click | Direct to view rent screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Rental code | 20 | String | Black | Right- justified |
| Bike code | 20 | String | Black | Right- justified |
| Bike type | 50 | String | Black | Right- justified |
| Renting period | 50 | String | Black | Right- justified |
| Deposit returned | 50 | String | Black | Right- justified |
| Total payment | 50 | String | Black | Right- justified |
| Time returned | 50 | Datetime | Black | Right- justified |

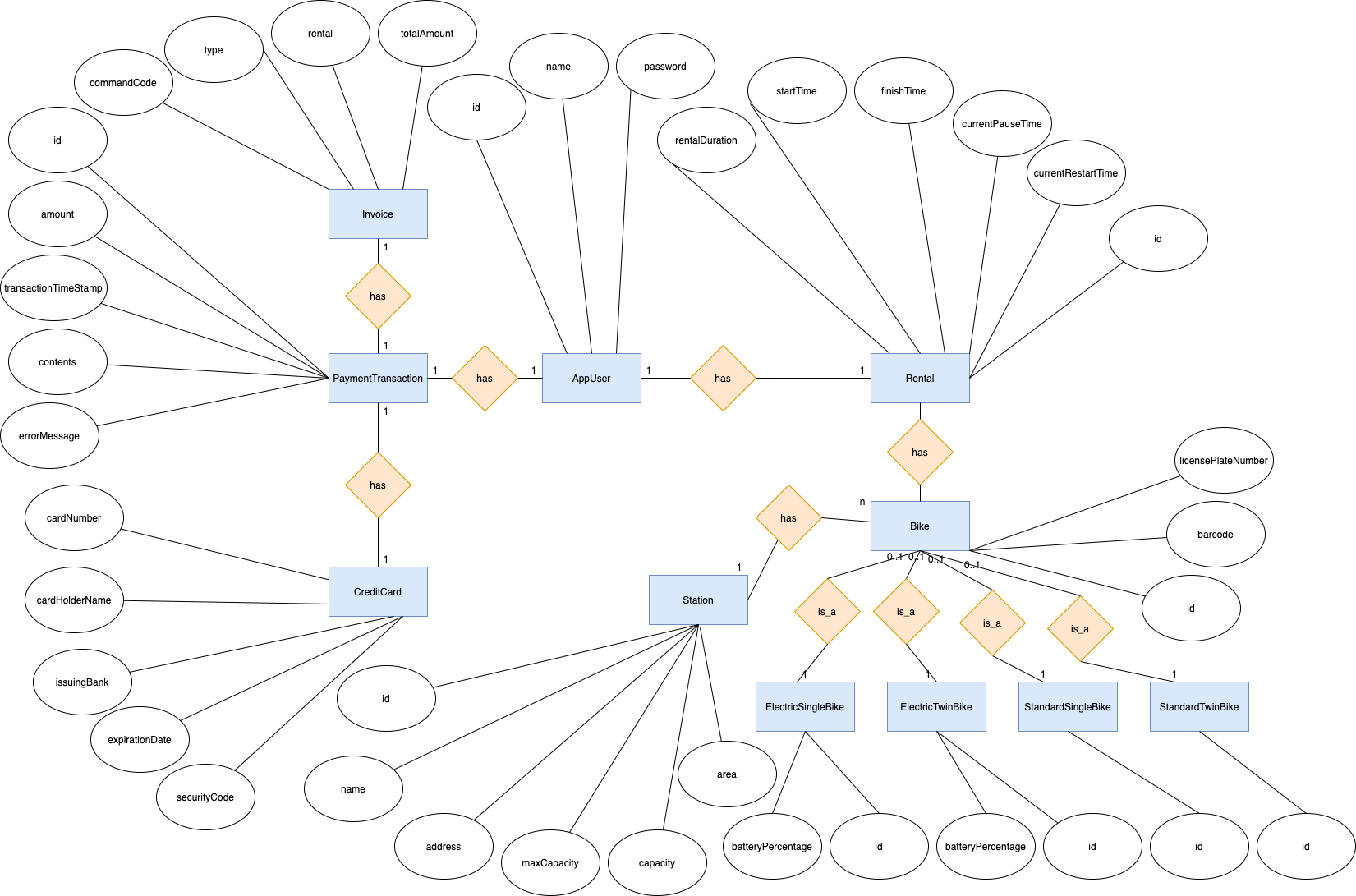
#### Error Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Error screen | 30/10/2021 |  |  | Nguyen Huy Hoang |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying error | Initial | Display the error | |
| OK button | Click | Go to home screen | |

#### 

## Data Modeling

### Conceptual Data Modeling



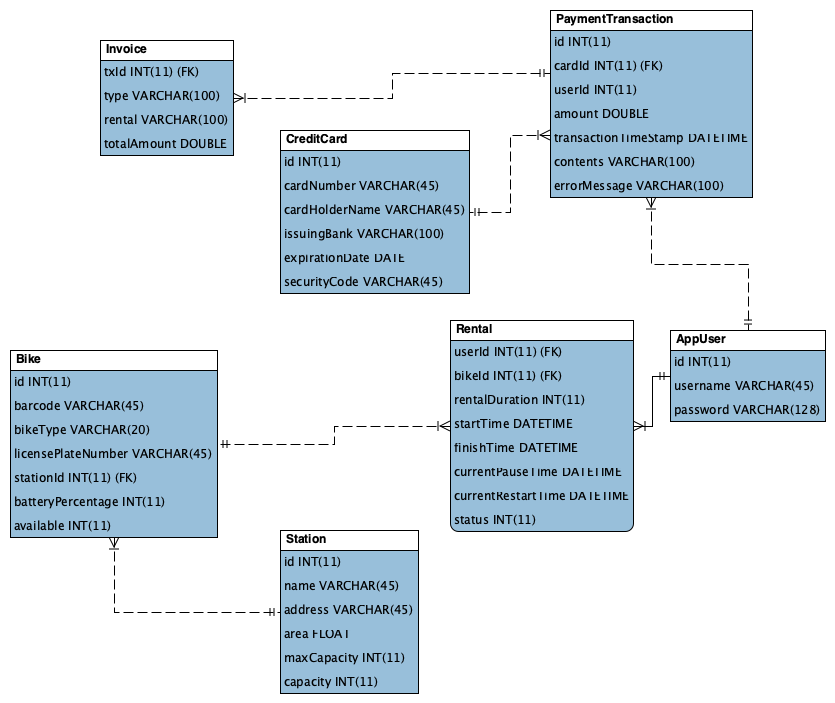
### Database Design

#### Database Management Systems

Database Management System: MySQL

MySQL is an open-source relational database management system.

#### Logical Data Model



#### Physical Data Model

**AppUser**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | Integer | Yes | Id, auto increment |
| 2 |  |  | username | Varchar(45) | Yes | Name of user |
| 3 |  |  | password | Varchar(128) | Yes | Password of user |

**Rental**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | x | x | userId | Integer | Yes | User’s ID |
| 2 | x | x | bikeID | Integer | Yes | Bike’s ID |
| 3 |  |  | rentalDuration | Integer | Yes | Duration of renting |
| 4 |  |  | startTime | Datetime | Yes | Time start renting |
| 5 |  |  | finishTime | Datetime | Yes | Time finish renting |
| 6 |  |  | currentPauseTime | Datetime | Yes | Current time of pausing |
| 7 |  |  | currentRestartTime | Datetime | Yes | Current time of restarting |
| 8 |  |  | status | Integer | Yes | Status of renting |

**Bike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | Integer | Yes | Id, auto increment |
| 2 |  |  | barcode | Varchar(45) | Yes | Barcode of bike |
| 3 |  |  | bikeType | Varchar(20) | Yes | The type of the bike |
| 4 |  |  | licensePlateNumber | Varchar(45) | Yes | License plate number of bike |
| 5 |  | X | stationId | Integer | No | station’s ID |
| 6 |  |  | batteryPercentage | Integer | No | The battery percentage of the bike |
| 7 |  |  | available | Integer | Yes | The availability of the bike |

**Station**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | Integer | Yes | Id, auto increment |
| 2 |  |  | name | Varchar(45) | Yes | Name of station |
| 3 |  |  | address | Varchar(45) | Yes | Address of station |
| 4 |  |  | area | Float | Yes | The area of the station |
| 5 |  |  | maxCapacity | Integer | Yes | Max capacity of station |
| 6 |  |  | capacity | Integer | Yes | Current capacity of station |

**PaymentTransaction**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | Integer | Yes | Id, auto increment |
| 2 |  | x | cardId | Integer | Yes | Card’s ID |
| 3 |  | x | userId | Integer | Yes | User’s ID |
| 4 |  |  | amount | Integer | Yes | Amount of transaction |
| 5 |  |  | transactionTimeStamp | Datetime | Yes | Time of transaction |
| 6 |  |  | contents | Varchar(100) | Yes | Contents of transaction |
| 7 |  |  | errorMessage | Varchar(100) | Yes | Error message of transaction |

**Credit Card**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | Integer | Yes | Id, auto increment |
| 2 |  |  | cardNumber | Varchar(45) | Yes | Number of card |
| 3 |  |  | cardHolderName | Varchar(45) | Yes | Holder name of card |
| 4 |  |  | issuingBank | Varchar(100) | Yes | Issuing bank of card |
| 5 |  |  | expirationDate | Date | Yes | Error message of transaction |
| 6 |  |  | securityCode | Varchar(45) | Yes | Expiration Date of card |

**Invoice**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | x |  | txId | Integer | Yes | Id, auto increment |
| 2 |  |  | type | Varchar(100) | Yes | Type of payment: deposit, payment, refund |
| 3 |  |  | rental | Varchar(100) | Yes | Status: pay or refund |
| 4 |  |  | totalAmount | Varchar(100) | Yes | Renting money |

**Database Script:**

*CREATE TABLE AppUser(*

*id INTEGER PRIMARY KEY AUTO\_INCREMENT NOT NULL,*

*username VARCHAR(45) NOT NULL,*

*password VARCHAR(128) NOT NULL*

*);*

*CREATE TABLE Station(*

*id INTEGER PRIMARY KEY AUTO\_INCREMENT NOT NULL,*

*name VARCHAR(45) NOT NULL,*

*address VARCHAR(45) NOT NULL,*

*area FLOAT NOT NULL,*

*maxCapacity INTEGER NOT NULL,*

*capacity INTEGER NOT NULL*

*);*

*CREATE TABLE Bike(*

*id INTEGER PRIMARY KEY AUTO\_INCREMENT NOT NULL,*

*barcode VARCHAR(45) NOT NULL,*

*bikeType VARCHAR(20) NOT NULL,*

*licensePlateNumber VARCHAR(45) NOT NULL,*

*stationId INTEGER,*

*batteryPercentage INTEGER DEFAULT 0,*

*available INTEGER DEFAULT 1,*

*CONSTRAINT fk\_Bike\_Station1 FOREIGN KEY (stationId) REFERENCES Station(id)*

*);*

*CREATE TABLE Rental(*

*userId INTEGER NOT NULL,*

*bikeId INTEGER NOT NULL,*

*rentalDuration INTEGER DEFAULT 0,*

*startTime DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*finishTime DATETIME,*

*currentPauseTime DATETIME,*

*currentRestartTime DATETIME,*

*status INTEGER DEFAULT 0,*

*PRIMARY KEY (userId, startTime),*

*CONSTRAINT fk\_Rental\_User1 FOREIGN KEY(userId) REFERENCES AppUser(id),*

*CONSTRAINT fk\_Rental\_Bike1 FOREIGN KEY(bikeId) REFERENCES Bike(id)*

*);*

*CREATE TABLE CreditCard(*

*id INTEGER PRIMARY KEY AUTO\_INCREMENT NOT NULL,*

*cardNumber VARCHAR(45) NOT NULL,*

*cardHolderName VARCHAR(45) NOT NULL,*

*issuingBank VARCHAR(100) NOT NULL,*

*expirationDate DATE NOT NULL,*

*securityCode VARCHAR(45) NOT NULL*

*);*

*CREATE TABLE PaymentTransaction(*

*id INTEGER PRIMARY KEY AUTO\_INCREMENT NOT NULL,*

*cardId INTEGER NOT NULL,*

*userId INTEGER NOT NULL,*

*amount REAL NOT NULL,*

*transactionTimeStamp DATETIME NOT NULL,*

*contents VARCHAR(100) NOT NULL,*

*errorMessage VARCHAR(100) NOT NULL,*

*CONSTRAINT fk\_PaymentTransaction\_CreditCard1 FOREIGN KEY(cardId)REFERENCES CreditCard(id),*

*CONSTRAINT fk\_PaymentTransaction\_User1 FOREIGN KEY(cardId)REFERENCES AppUser(id)*

*);*

*CREATE TABLE Invoice(*

*txId INTEGER DEFAULT NULL,*

*type VARCHAR(100) NOT NULL,*

*rental VARCHAR(100) NOT NULL,*

*totalAmount REAL NOT NULL,*

*PRIMARY KEY(type,rental),*

*CONSTRAINT fk\_Invoice\_PaymentTransaction1 FOREIGN KEY(txId)REFERENCES PaymentTransaction(id)*

*);*

## Non-Database Management System Files

* Config file for bike types:

Attributes recorded for each bike type:

*"bikeTypeName",*

*"monetaryValue",*

*"electricBike",*

*"noOfSaddles”,*

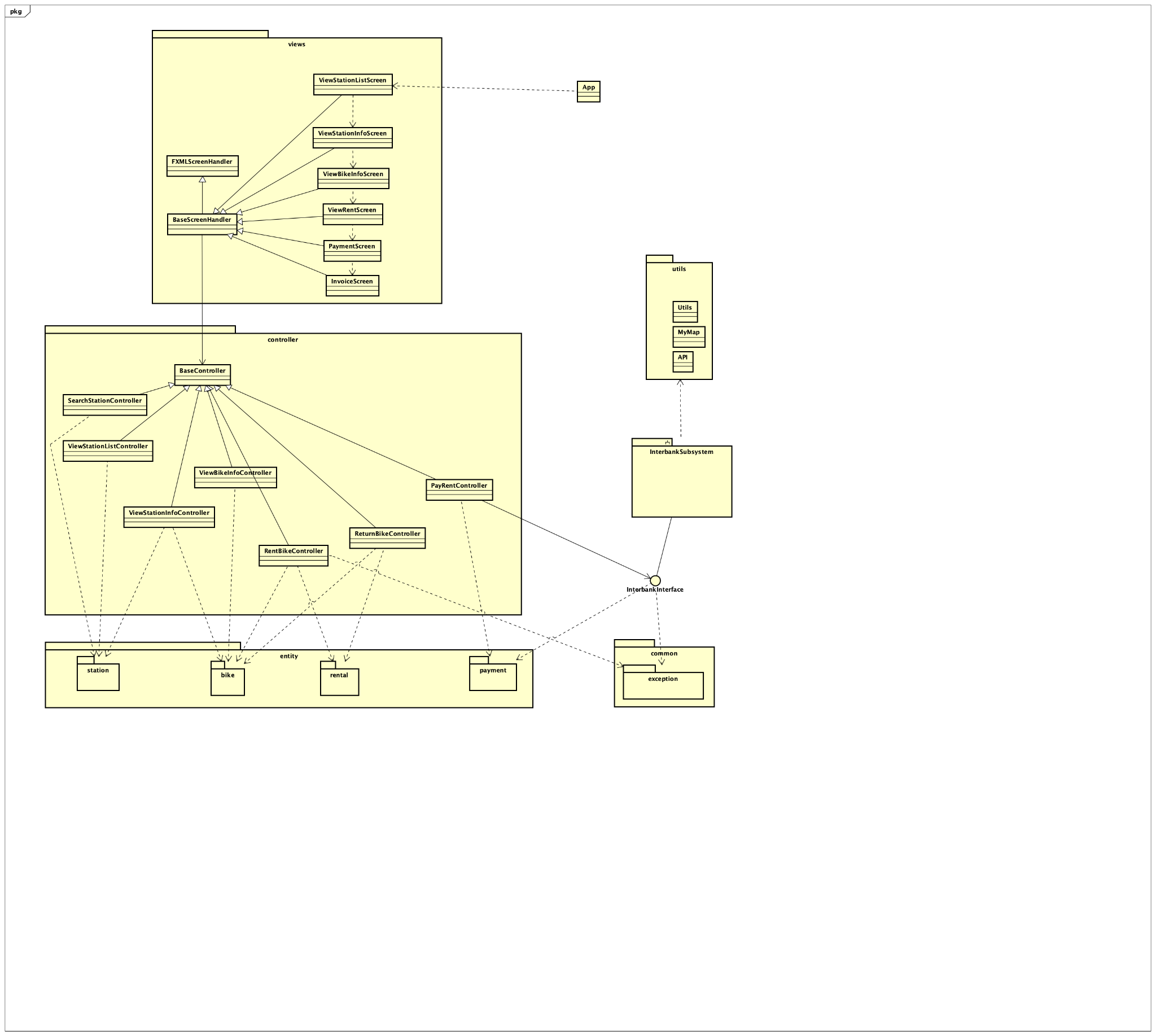
*"noOfPedals”,*

*"noOfRearSeats”,*

*"description"*

## Class Design

### General Class Diagram

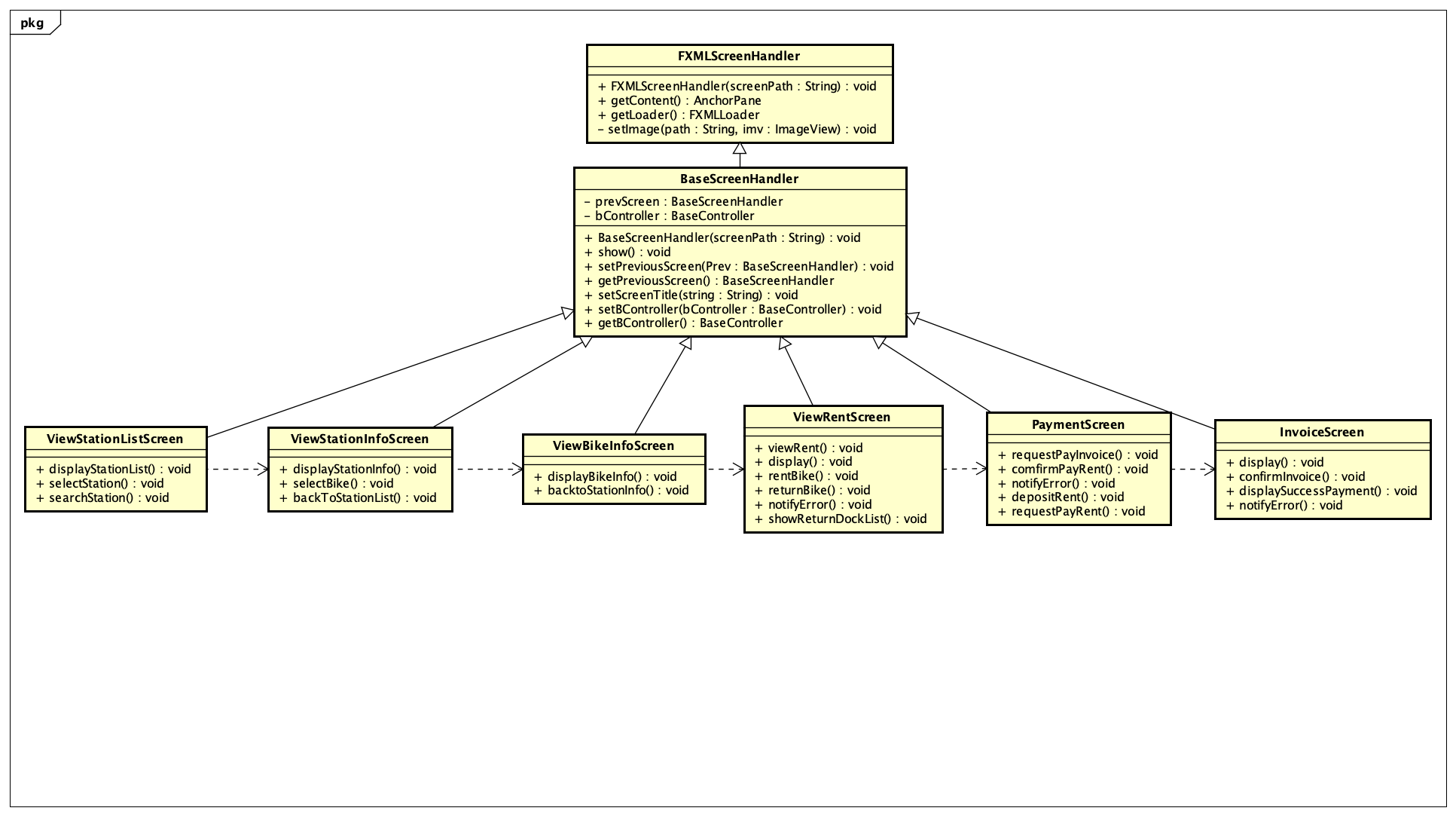


### Class Diagrams

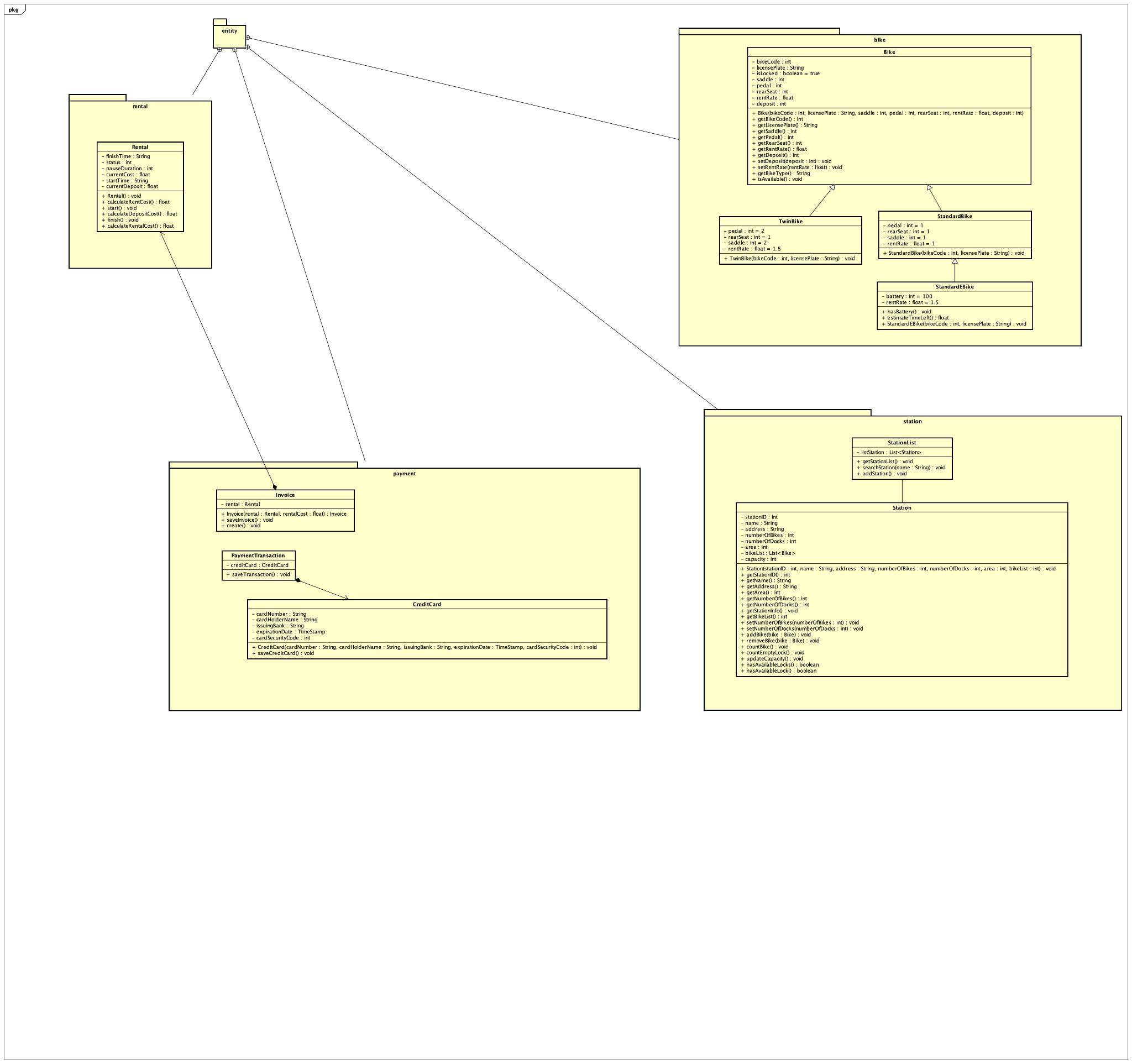
#### Class Diagram for Package “Controller”



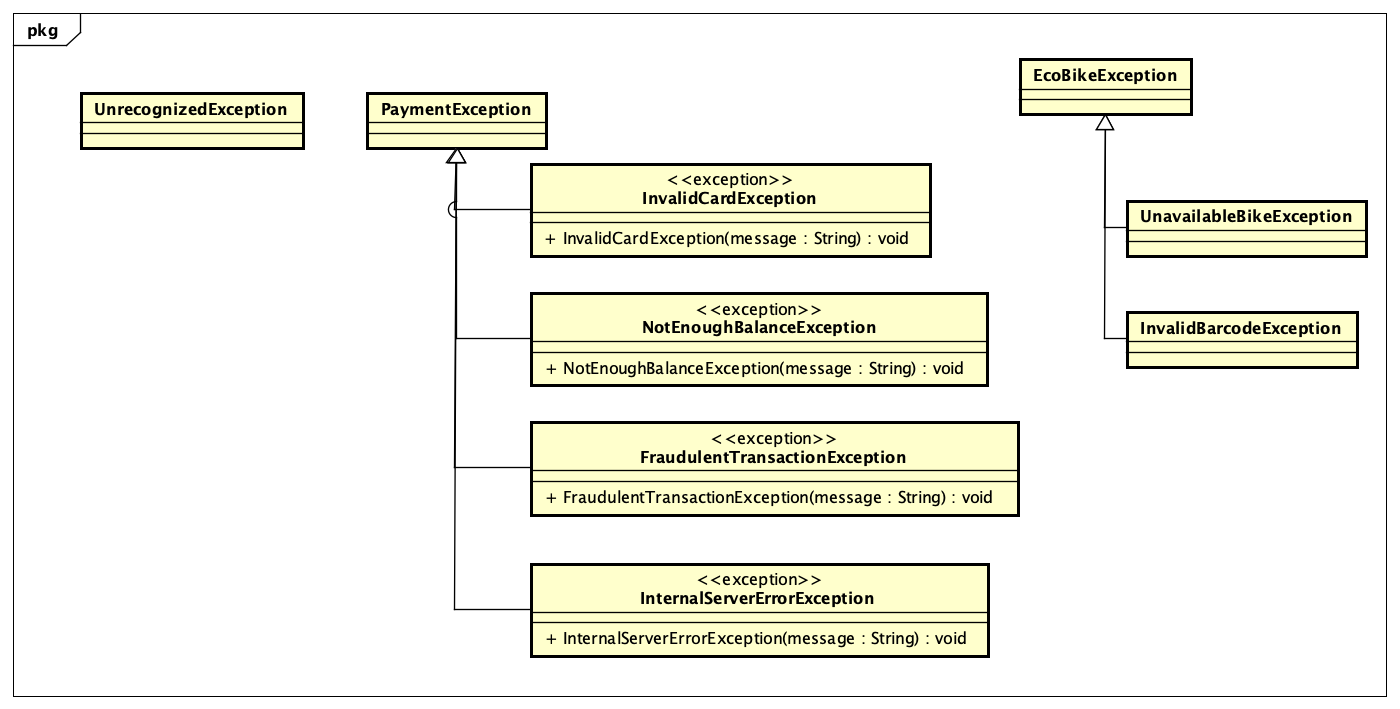
#### Class Diagram for Package “View-handler”



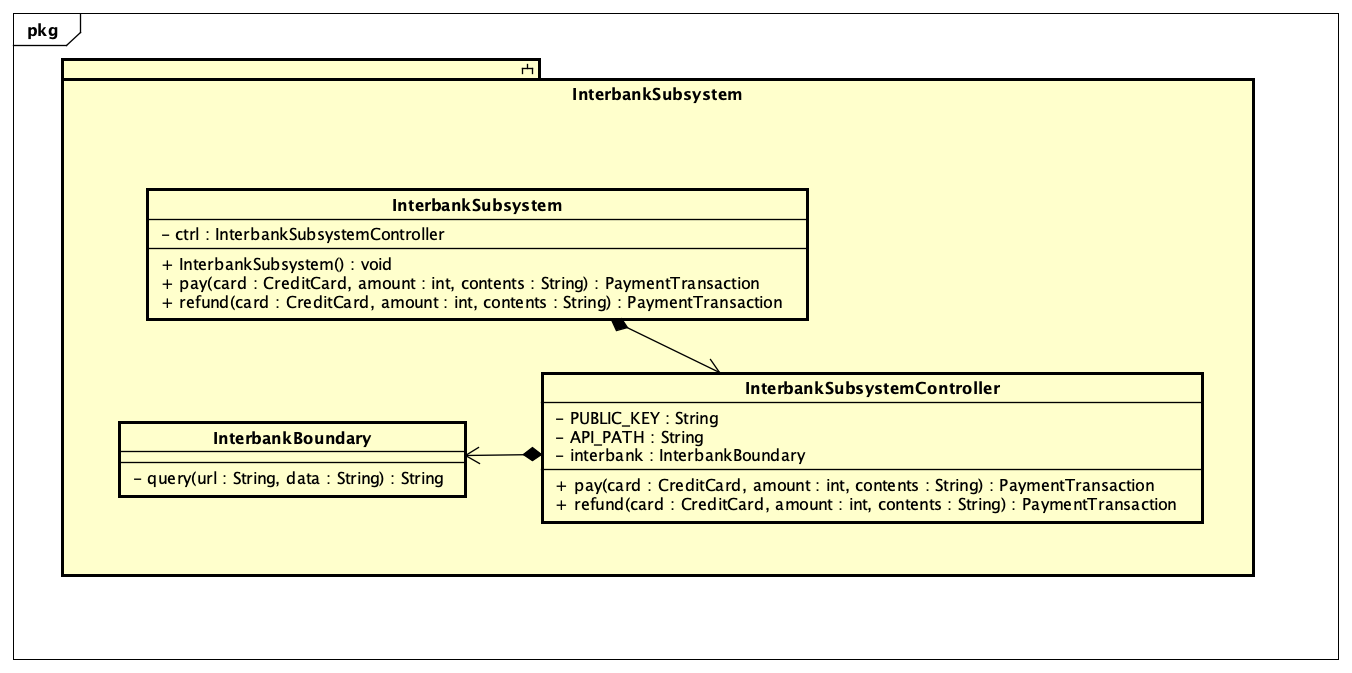
#### Class Diagram for Package “Entity”



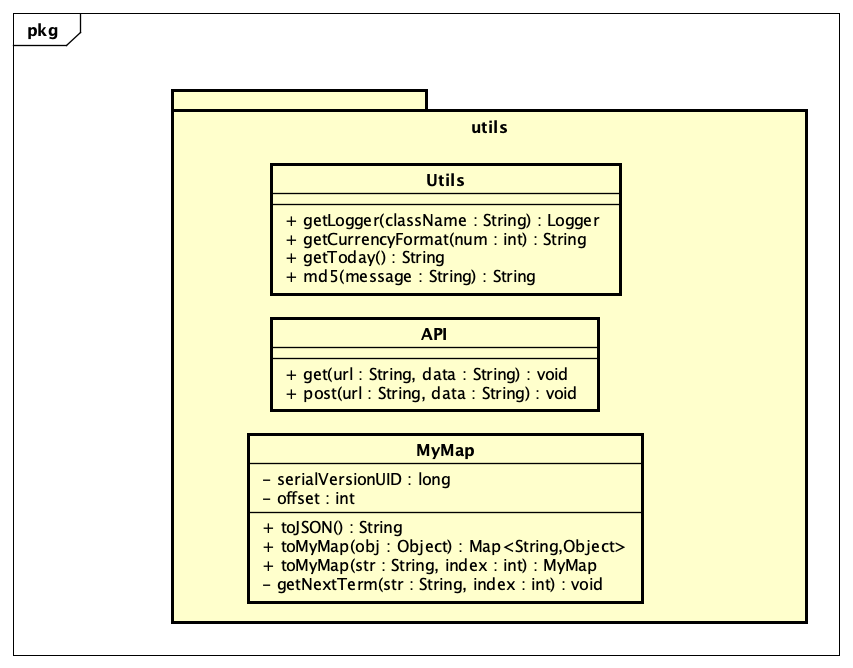
#### Class Diagram for Package “Exception”



#### Class Diagram for Subsystem “Interbank”

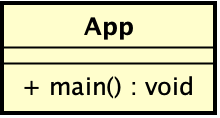


#### Class Diagram for Package “Utils”



### Class Design

#### Class “App”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | main | void | The main entry point for the application |

*Parameter:*

None

*Exception:*

None

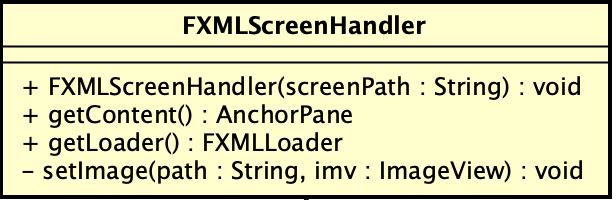
**Method**

None

**State**

None

#### Class “FXMLScreenHandler”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | FXMLScreenHandler | void | Constructs a FXMLScreenHandler from a screenPath |
| 2 | getContent | AnchorPane | Gets the root of the screen |
| 3 | getLoader | FXMLLoader | Gets the loader |

*Parameter:*

* screenPath: the path to the fxml file of the screendock

*Exception:*

* None

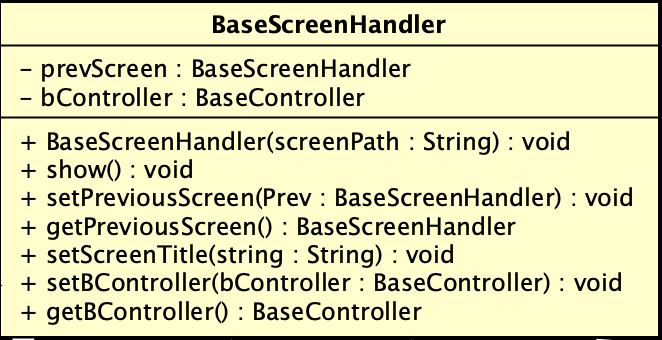
**Method**

* setImage: Sets image file in *path* to the ImageView *imv*

**State**

None

#### Class “BaseScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | prevScreen | BaseScreenHandler |  | Represents the previous screen (in case the user go back) |
| 2 | bController | BaseController |  | Represents the BaseController for the screen |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | BaseScreenHandler | void | Constructs a FXMLScreenHandler from a screenPath |
| 2 | show |  | Shows the screen to the user |

*Parameter:*

* screenPath: the path to the fxml file of the screen

*Exception:*

* None

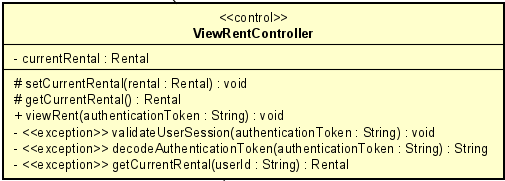
**Method**

* setPreviousScreen: set the screen in BaseScreenHandler *prev* as the previous screen of the current screen
* setPreviousScreen: get the previous screen as a BaseScreenHandler object
* setScreenTitle: set *string* as the title of the screen
* setBController: set the BaseController for the screen
* getBController: get the BaseController of the screen

**State**

None

#### Class “BaseController”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | BaseController | void | Constructs a BaseController |

*Parameter:*

None

*Exception:*

None

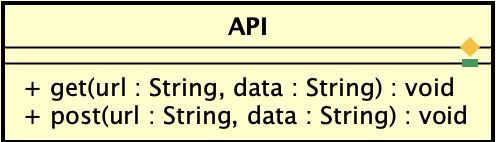
**Method:**

None

**State:**

None

#### Class “API”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | get | String | Make a HTTP GET request to the given url with the given data |
| 2 | post | String | Make a HTTP POST request to the given url with the given data |

*Parameter:*

* url: the destination url to send the request
* data: the data of the request

*Exception:*

* None

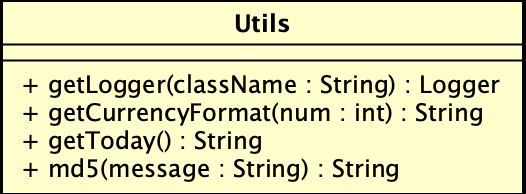
**Method:**

None

**State:**

None

#### Class “Utils”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getLogger | Logger | Gets the logger of the class |
| 2 | getCurrencyFormat | String | Return the formatted string according to the currency |
| 3 | getToday | String | Return a string representing the current time in the format yyyy-MM-dd HH:mm:ss |
| 4 | md5 | String | Return the hash value of the message |

*Parameter:*

* className: the name of the class in string format
* num: the amount of currency
* message: the message in string format

*Exception:*

None

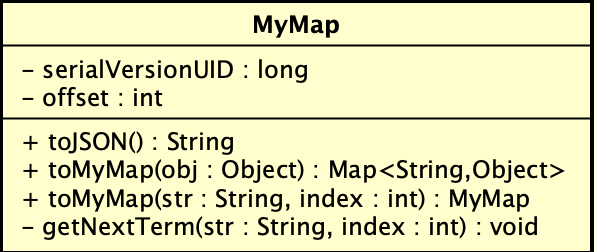
**Method**

None

**State**

None

#### Class “MyMap”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | serialVersionUID | long | 1 | The serial version UID |
| 2 | offset | int | 0 | Keep track of the current index when calling a function |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | toJSON | String | Return a String that represents a JSON object of the MyMap object |
| 2 | toMyMap | Map<String, Object> | Return a Map that represents the mapping among attribute names and their values of the object obj |
| 3 | toMyMap | MyMap | Return a MyMap that represents the interested substring in a String |

*Parameter:*

* obj: the object of interest
* string: the atring of interest
* idx: the index of the character in the original string where the substring begins

*Exception:*

None

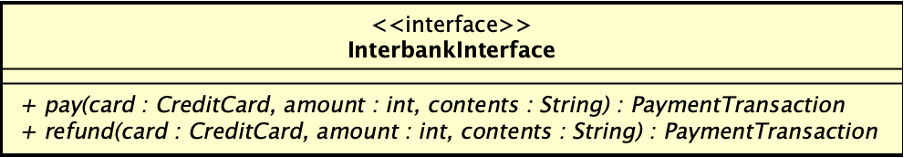
**Method**

* getNextTerm: Return a string that represents the term in between the double quotes inside a string

**State**

None

#### Interface “InterbankInterface”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | pay | PaymentTransaction | Pay rent, then return the payment transaction |
| 2 | refund | PaymentTransaction | Refund, then return the payment transaction |

*Parameter:*

* card: the credit card used for payment/refund
* amount: the amount to pay/refund
* contents: the transaction contents

*Exception:*

* PaymentException: if responded with a pre-defined error code
* UnrecognizedException: if responded with an unknown error code or something goes wrong

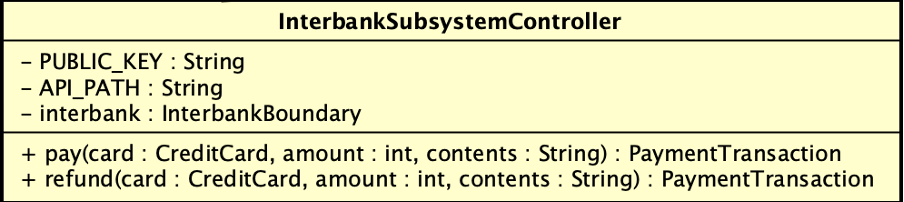
**Method:**

None

**State:**

None

#### Class “InterbankSubsystemController”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | PUBLIC\_KEY | String | supply by subsystem | Being used to encode the data sent to interbank api gateway |
| 2 | API\_PATH | String | supply by subsystem | Path to interbank api gateway |
| 3 | interbank | InterbankBoundary | Injected by IoC | Communicate to the interbank api gateway |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | pay | PaymentTransaction | Pay rent, then return the payment transaction |
| 2 | refund | PaymentTransaction | Refund, then return the payment transaction |

*Parameter:*

* card: the credit card used for payment/refund
* amount: the amount to pay/refund
* contents: the transaction contents

*Exception:*

* PaymentException: if responded with a pre-defined error code
* UnrecognizedException: if responded with an unknown error code or something goes wrong

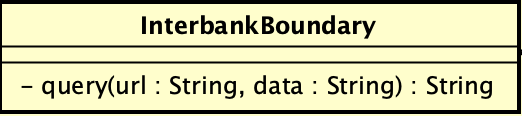
**Method:**

* generateURL: Generate correctly-form URL satisfied the standard interbank system design
* handleResponse: Read the response returned from the interbank by converting JSON-formatted string into separated list of information

**State:**

None

#### Class “InterbankBoundary”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | query | String | Connect to the Interbank API, make query and receive response |

*Parameter:*

* url: the destination url to send the request
* data: the data of the request

*Exception:*

* UnrecognizedException: if responded with an unknown error code or something goes wrong

**Method:**

None

**State:**

None

#### Class “InterbankSubsystem”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | ctrl | InterbankSubsystemController |  | Represent the controller of the system |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | InterbankSubsystem |  | Create a new InterbankSubsystem with its InterbankSubsystemController |
| 2 | pay | PaymentTransaction | Pay rent, then return the payment transaction |
| 3 | refund | PaymentTransaction | Refund, then return the payment transaction |

*Parameter:*

* card: the credit card used for payment/refund
* amount: the amount to pay/refund
* contents: the transaction contents

*Exception:*

* PaymentException: if responded with a pre-defined error code
* UnrecognizedException: if responded with an unknown error code or something

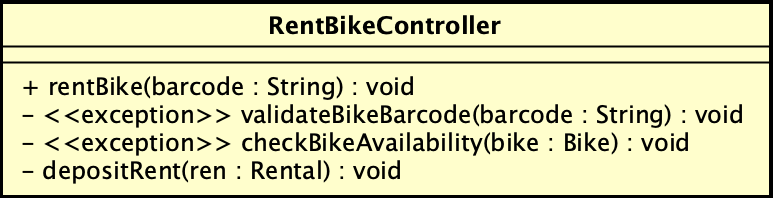
**Method:**

None

**State:**

None

#### Class “RentBikeController”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | rentBike | Rental | Rent bike, and then return the corresponding rental |

*Parameter:*

* barcode: the barcode of the bike
* rental: the rental information

*Exception:*

* InvalidBarcodeException: if the input barcode is not valid
* UnavailableBikeException: if the bike is not available for renting

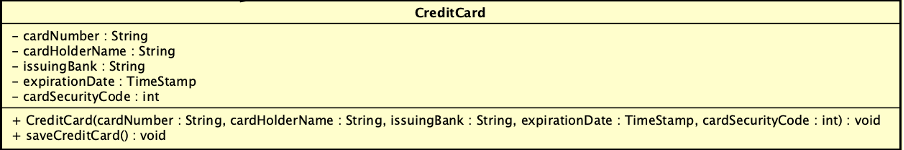
**Method:**

* validateBikeBarcode: validate the bike barcode from input
* checkBikeAvailability: check whether the bike is available for renting
* depositRent: deposit the initial renting fee

**State:**

None

#### Class “CreditCard”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | cardNumber | String |  | Controller of payment |
| 2 | cardHolderName | String |  | Number of card |
| 3 | issuingBank | String |  | Holder name of card |
| 4 | expirationDate | Timestamp |  | Security code of card |
| 5 | cardSecurityCode | int |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | CreditCard | Invoice | Create credit card instance |
| 2 | saveCreditCard |  | Save the card information |

*Parameter:*

* cardNumber
* cardHolderName
* issuingBank
* expirationDate
* cardSecurityCode

*Exception:*

None

**Method:**

None

**State:**

None

#### Class “Invoice”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | cardNumber | String |  | Controller of payment |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | Invoice | Invoice | Create invoice instance |
| 2 | saveInvoice | void | Save the invoice information |
| 3 | create | void | Create invoice instance with default value |

*Parameter:*

* rental: rental linked to invoice instance
* rentalCost: cost of rental to add to invoice

*Exception:*

None

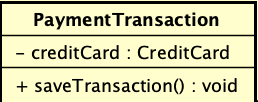
**Method:**

None

**State:**

None

#### Class “PaymentTransaction”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | creditCard | CreditCard | null | Represent the credit card used in this PaymentTransation |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | saveTransaction | void | Save the transaction |

*Parameter:*

None

*Exception:*

None

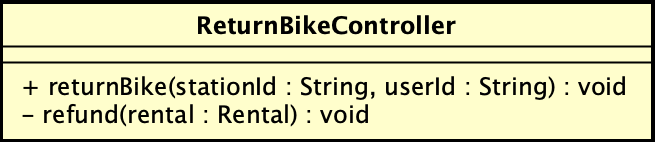
**Method:**

None

**State:**

None

#### Class “ReturnBikeController”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | refund | void | Confirm payment |
| 2 | renturnBike | void | Proceed rent bike |

*Parameter:*

None

*Exception:*

None

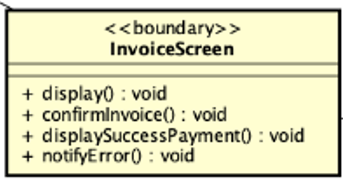
**Method:**

None

**State:**

None

#### Class “InvoiceScreen”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | display | void | Display invoice page UI |
| 2 | confirmInvoice | void | Confirm selected invoice |
| 3 | displaySuccessPayment | void | Display payment information after invoice is proceeded |
| 4 | notifyError | void | Notify error when proceeding |

*Parameter:*

None

*Exception:*

None

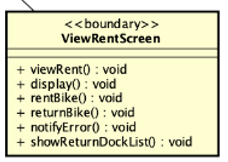
**Method:**

None

**State:**

None

#### Class “ViewRentScreen”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | viewRent | void | Display rent information |
| 2 | display | void | Display view rent screen UI |
| 3 | returnBike | void | Return currently renting bike |
| 4 | rentBike | void | Rent selected bike |
| 5 | notifyError | void | Notify error when processing rent or return bike |
| 6 | showReturnDockList | void | Display available dock to return bike |

*Parameter:*

None

*Exception:*

None

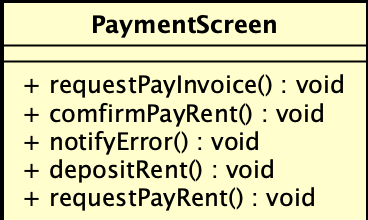
**Method:**

None

**State:**

None

#### Class “PaymentScreen”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | requestPayInvoice | void | Start paying invoice processing |
| 2 | confirmPayRent | void | Process pay rent with current rent information |
| 3 | notifyError | void | Display error from processing |
| 4 | depositRent | void | Deposit an amount from user account |
| 5 | requestPayRent | void | Start processing current info to display rent |

*Parameter:*

None

*Exception:*

None

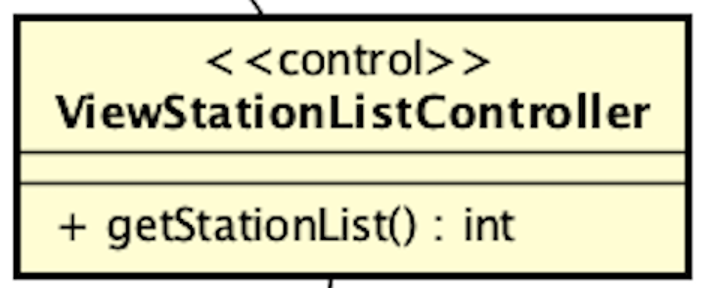
**Method:**

None

**State:**

None

#### Class “ViewStationListController”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | getStationList | void | Get list of stations |

*Parameter:*

None

*Exception:*

None

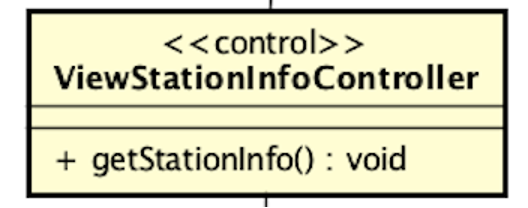
**Method:**

None

**State:**

None

#### Class “ViewStationInfoController”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getStationInfo | void | Get info of the current station |

*Parameter:*

None

*Exception:*

None

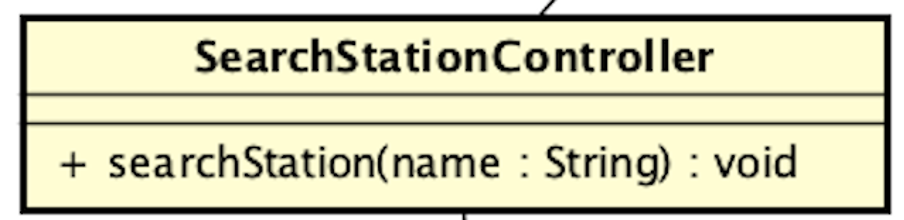
**Method:**

None

**State:**

None

#### Class “SearchStationController”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | searchStation | void | Search for a station by its name |

*Parameter:*

* name - String: name of the station you want to search for

*Exception:*

None

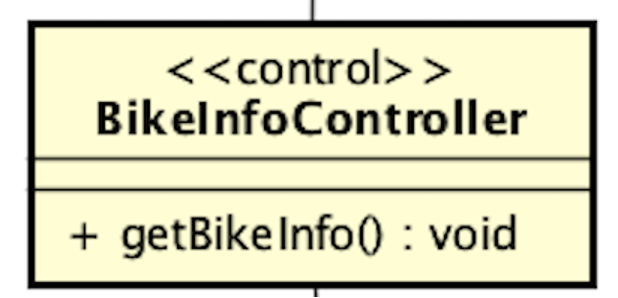
**Method:**

None

**State:**

None

#### Class “ViewBikeInfoController”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getBikeInfo | void | Get info of the current bike |

*Parameter:*

None

*Exception:*

None

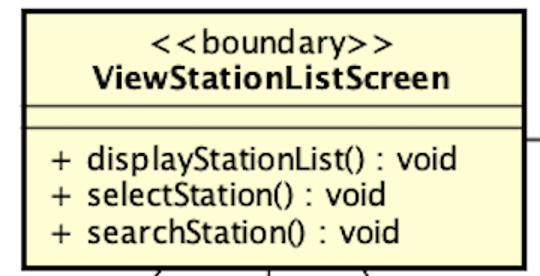
**Method:**

None

**State:**

None

#### Class “ViewStationListScreen”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | selectStation | void | Select a station from the list |
| 2 | searchStation | void | Call SearchStationController to search station by name or address |
| 3 | displayStationList | void | Display passed list of stations |

*Parameter:*

None

*Exception:*

None

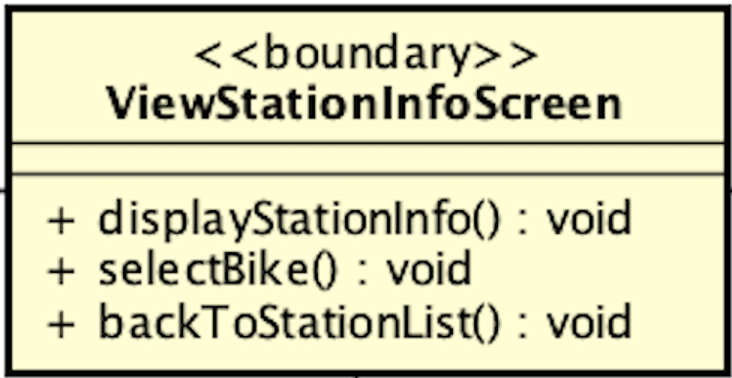
**Method:**

* selectStation: Get station from user selection and call getStationInfo() in ViewStationInfoController to display station info in ViewStationInfoScreen
* searchStation: Input a name and search for a station using the searchStation() in SearchStationController

**State:**

None

#### Class “ViewStationInfoScreen”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | displayStationInfo | void | Display the info of the current station |
| 2 | selectBike | void | Select a bike from the list |
| 3 | backToStationList | void | Back to the previous station list screen |

*Parameter:*

None

*Exception:*

None

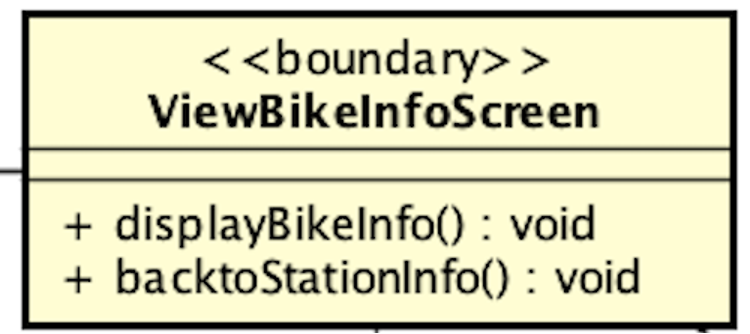
**Method:**

* selectBike: Get bike from user selection and call getBikeInfo() in ViewBikeInfoController to display bike info in ViewBikeInfoScreen

**State:**

None

#### Class “ViewBikeInfoScreen”

****

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | displayBikeInfo | void | Display the information of the current bike |
| 2 | backToStationInfo | void | Return to the previous station info screen |

*Parameter:*

None

*Exception:*

None

**Method:**

None

**State:**

None

#### Class “Bike”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | bikeCode | int | NULL | Code of the bike in the system |
| 2 | licensePlate | String | NULL | License plate of the bike |
| 3 | isLocked | boolean | True | Lock condition of the bike |
| 4 | saddle | int | NULL | Number of saddles |
| 5 | pedal | int | NULL | Number of pedals |
| 6 | rearSeat | int | NULL | Number of rear seats |
| 7 | rentRate | float | NULL | Rate of the rent price |
| 8 | deposit | int | NULL | The deposit amount needed to rent the bike |
| 9 | battery | int | 100 | Percentage of battery left |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getBikeCode | int | Get the code of the bike |
| 2 | getLicensePlate | String | Get the license plate of the bike |
| 3 | getSaddle | int | Get number of saddles of the bike |
| 4 | getPedal | int | Get number of pedals of the bike |
| 5 | getRearSeat | int | Get number of rear seats of the bike |
| 6 | getRentRate | float | Get rentRate of the bike |
| 7 | getDeposit | int | Get deposit of the bike |
| 8 | setDeposit | void | Set the deposit of the bike |
| 9 | setRentRate | void | Set the rentRate of the bike |
| 10 | getBikeType | String | Get the type of the bike |
| 11 | isAvailable | boolean | Check the available condition of the bike |
| 12 | getBattery | int | Get the percentage of battery left |
| 13 | estimateTimeLeft | int | Get the number of minutes left the electric bike can run |

*Parameter:*

None

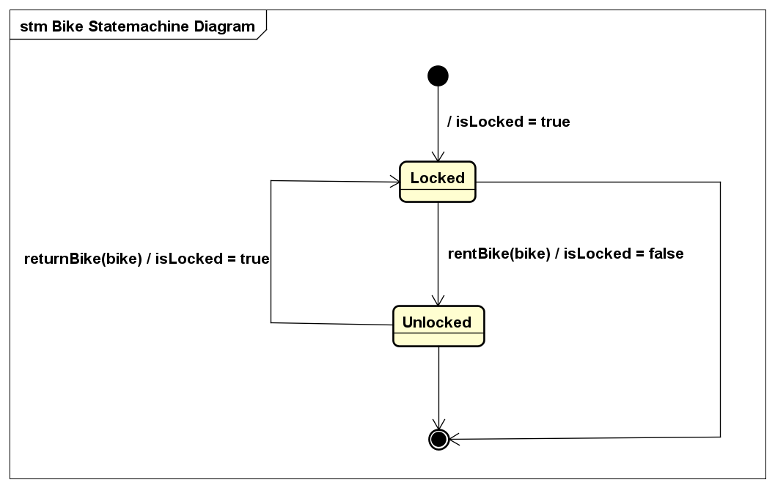
*Exception:*

None

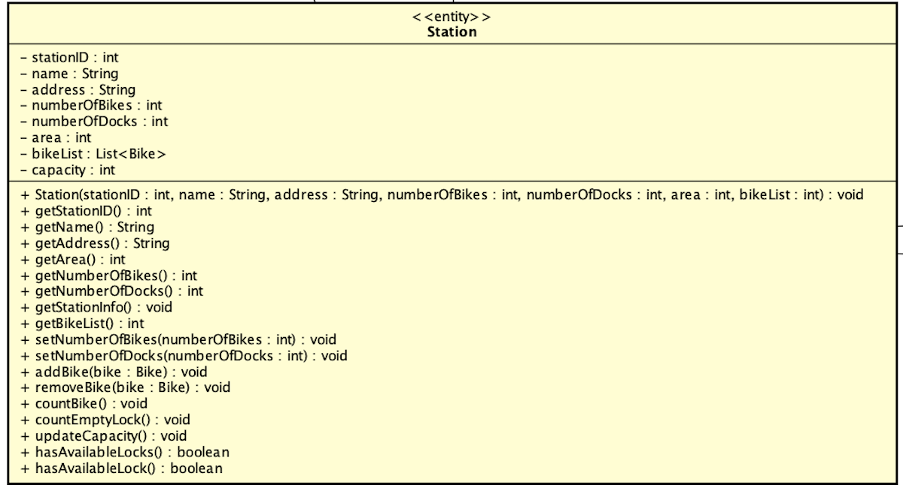
**Method:**

None

**State:**



#### Class “Station”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | stationID | int | NULL | ID of the station in the system |
| 2 | name | String | NULL | Name of the station |
| 3 | address | String | NULL | Address of the station |
| 4 | numberOfBikes | int | NULL | Number of bikes currently in the station |
| 5 | numberOfDocks | int | NULL | Number of docks in the station |
| 6 | area | int | NULL | Area size of the station |
| 7 | bikeList | List<Bike> | NULL | List of the bikes currently in the station |
| 8 | capacity | int | NULL | The number of available docks left in the station |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getStationID | int | Get the stationID |
| 2 | getName | String | Get the name of the station |
| 3 | getAddress | String | Get the address of the station |
| 4 | getNumberOfBikes | int | Get the number of bikes currently in the station |
| 5 | getNumberOfDocks | int | Get the number of docks in the station |
| 6 | getBikeList | List <Bike> | Get the list of bikes currently in the station |
| 7 | addBike | Bike | Add a bike to the list of bikes currently in the station |
| 8 | removeBike | Bike | Remove of bike from the list of bikes currently in the station |
| 9 | updateCapacity | void | Update the capacity of the station according to numberOfBikes and numberOfDocks |

*Parameter:*

* bike – Bike: A particular bike object
* rentalCode – String: code given by the system when user successfully rents a bike and can be used for other purposes
* barcode – String: Unique code of each bike in String format

*Exception:*

None

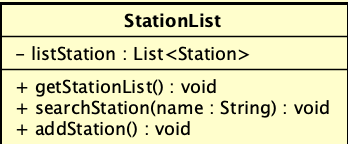
**Method:**

None

**State:**

None

#### Class “StationList”

****

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | listStation | List<Station> | NULL | List of the stations |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getStationList | List<Station> | Get list of the stations |
| 2 | searchStation | Station | Search for a station by its name |
| 3 | addStation | void | Add a new station to the list |

*Parameter:*

* name – String: Name of station
* station – Station: The station you want to add

*Exception:*

None

**Method:**

None

**State:**

None

# Design Considerations

***<Describe issues which need to be addressed or resolved before attempting to devise a complete design solution>***

## Goals and Guidelines

* + 1. Goals

The project was designed with simplicity and reliability in mind. This means that the program would be easily accessible and guaranteed to deliver all the requirements. However, it should not be only plain and boring. The application should bring a pleasant experience for the users and also attract other new customers for the business.

* + 1. Guidelines

## Architectural Strategies

*<Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off.*

*Examples of design decisions might concern (but are not limited to) things like the following:*

*• Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)*

*• Reuse of existing software components to implement various parts/features of the system*

*• Future plans for extending or enhancing the software*

*• User interface paradigms (or system input and output models)*

*• Hardware and/or software interface paradigms*

*• Error detection and recovery*

*• Memory management policies*

*• External databases and/or data storage management and persistence*

*• Distributed data or control over a network*

*• Generalized approaches to control*

*• Concurrency and synchronization*

*• Communication mechanisms*

*• Management of other resources*

>

* In this project, the programming language chosen is Java as it’s taught in the course and also provides good support for the OOP paradigm.
* For the GUI, we use JavaFX as the framework provides an easy interface for UI design and integration with the code controller.
* The database is hosted using MySQL as the DBMS.

## Coupling and Cohesion

*<Evaluate your design and describe which levels of coupling and cohesion that your design is at. Give proofs for your assumptions. Explain if there is any special design or exceptions>*

## Design Principles

*<Does your design follow the SOLID principles for the new requirements/changing requirements? Give proofs for your assumptions. Explain if there is any special design or exceptions>*

## Design Patterns

*<Do you use any design patterns for your design? If yes, describe detailly why you use those design patterns? Describe in detail on the solutions and how to implement each design pattern>*