HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.3

EcoBikeRental (EBR)

Subject: ITSS Software Development

Group 03

Trịnh Thu Hải - 20184255

Nguyễn Huy Hoàng - 20184265

Bùi Thanh Tùng - 20184324

*Hanoi, November, 2021*

Table of Contents

Table of Contents 1

*1* Introduction 7

1.1 Objective 7

1.2 Scope 7

1.3 Glossary 7

1.4 References 10

2 Overall Description 12

2.1 General Overview 12

2.1.1 Usecase diagram 12

2.1.2 Usecase “View Dock Info” 13

2.1.3 Usecase “View Rent Info” 14

2.1.4 Usecase “Rent Bike” 15

2.1.5 Usecase “Return Bike” 16

2.2 Assumptions/Constraints/Risks 17

2.2.1 Assumptions 17

2.2.2 Constraints 17

2.2.3 Risks 17

3 System Architecture and Architecture Design 18

3.1 Architectural Patterns 18

3.2 Interaction Diagrams 18

3.2.1 View Dock Info 18

3.2.2 View Rent Info 19

3.2.3 Rent Bike 20

3.2.4 Return Bike 20

3.3 Analysis Class Diagrams 21

3.3.1 View Dock Info 21

3.3.2 View Rent Info 21

3.3.3 Rent Bike 22

3.3.4 Return Bike 22

3.4 Unified Analysis Class Diagram 22

3.5 Security Software Architecture 23

4 Detailed Design 24

4.1 User Interface Design 24

4.1.1 Screen Configuration Standardization 24

4.1.2 Screen Transition Diagrams 25

4.1.3 Screen Specifications 25

4.1.3.1 Home Station List Screen 25

4.1.3.2 Return Bike Station List Screen 27

4.1.3.3 Station Details Screen 28

4.1.3.4 Rent New Bike Screen 29

4.1.3.5 View Rental Screen 29

4.1.3.6 Invoice Screen 30

4.1.3.7 Splash Screen 31

4.1.3.8 Result Screen 31

4.1.3.9 Pop-up screen 32

4.2 Data Modeling 33

4.2.1 Conceptual Data Modeling 33

4.2.2 Database Design 34

4.2.2.1 Database Management Systems 34

4.2.2.2 Logical Data Model 34

4.2.2.3 Physical Data Model 35

4.3 Non-Database Management System Files 42

4.4 Class Design 43

4.4.1 General Class Diagram 43

4.4.2 Class Diagrams 44

4.4.2.1 Class Diagram for Package “Controller” 44

4.4.2.2 Class Diagram for Package “View-handler” 44

4.4.2.3 Class Diagram for Package “Bike” 45

4.4.2.4 Class Diagram for Package “Payment” 45

4.4.2.5 Class Diagram for Package “Station” 46

4.4.2.6 Class Diagram for Package “Subsystem” 46

4.4.2.7 Class Diagram for Package “Repository” 47

4.4.2.8 Class Diagram for Package “Rental” 47

4.4.2.9 Class Diagram for Package “Common” 48

4.4.2.10 Class Diagram for Package “User” 48

4.4.2.11 Class Diagram for Package “Utils” 48

4.4.3 Class Design 49

4.4.3.1 Class “API” 49

4.4.3.2 Class “ViewsConfigs” 50

4.4.3.3 Class “AppUser” 51

4.4.3.4 Class “AppUserRepo” (extends interface BaseRepo) 52

4.4.3.5 Class “AppUserRepoImpl” (implements AppUserRepo, ResultSetMappable) 53

4.4.3.6 Class “AppUserService” 54

4.4.3.7 Class “AppUserServiceImpl” (implements AppUserService) 55

4.4.3.8 Class “BaseRepo” 55

4.4.3.9 Class “BaseScreenHandler” 56

4.4.3.10 Class “Bike” 58

4.4.3.11 Class “BikeRepo” (extends BaseRepo) 59

4.4.3.12 Class “BikeRepoImpl” ( implements BikeRepo, ResultSetMappable) 60

4.4.3.13 Class “BikeService” 61

4.4.3.14 Class “BikeServiceImpl” (implements BikeService) 62

4.4.3.15 Class “BikeType” 63

4.4.3.16 Class “BikeTypeRepo” (extends BaseRepo) 64

4.4.3.17 Class “BikeTypeRepoImpl” (implements BikeTypeRepo, ResultSetMappable) 65

4.4.3.18 Class “CreditCard” 66

4.4.3.19 Class “CreditCardRepo” (extends BaseRepo) 67

4.4.3.20 Class “CreditCardRepoImpl” (implements CreditCardRepo, ResultSetMappable) 67

4.4.3.21 Class “CreditCardValidator” 68

4.4.3.22 Class “DataSource” 69

4.4.3.23 Class “DateTimeUtils” 71

4.4.3.24 Class “DepositBikeInvoiceDTO” 71

4.4.3.25 Class “DockDetailDTO” 72

4.4.3.26 Class “DockStation” 73

4.4.3.27 Class “DockStationRepo” (extends BaseRepo) 75

4.4.3.28 Class “DockStationRepoImpl” 76

4.4.3.29 Class “DockStationService” 76

4.4.3.30 Class “DockStationServiceImpl” 77

4.4.3.31 Class “FailureMessage” (extends Message) 78

4.4.3.32 Class “FXMLScreenHandler” 79

4.4.3.33 Class “HomeScreenHandler” (extends BaseScreenHandler implements ScreenElementObserver) 80

4.4.3.34 Class “HomeStationListItemHandler”( extends FXMLScreenHandler implements ScreenElementObservable) 82

4.4.3.35 Class “InterbankBoundary” 83

4.4.3.36 Class “InterbankInterface” 84

4.4.3.37 Class “InterbankSubsystem” (implements InterbankInterface) 85

4.4.3.38 Class “InterbankSubsystemController” 86

4.4.3.39 Class “Invoice” 87

4.4.3.40 Class “InvoiceRepo”(extends BaseRepo) 88

4.4.3.41 Class “InvoiceRepoImpl”(implements InvoiceRepo, ResultSetMappable) 89

4.4.3.42 Class “InvoiceScreenHandler” (extends BaseScreenHandler) 90

4.4.3.43 Class “IRentalCostStrategy” 92

4.4.3.44 Class “IRentalCountingTimeStrategy” 93

4.4.3.45 Class “IRentalDepositStrategy” 93

4.4.3.46 Class “Message” 94

4.4.3.47 Class “MyMap” (extends LinkedHashMap) 95

4.4.3.48 Class “PaymentService” 96

4.4.3.49 Class “PaymentServiceImpl” (implements PaymentService) 97

4.4.3.50 Class “PaymentTransaction” 99

4.4.3.51 Class “PaymentTransactionRepo” 100

4.4.3.52 Class “PaymentTransactionRepoImpl” (implements PaymentTransactionRepo, ResultSetMappable<PaymentTransaction>) 100

4.4.3.53 Class “PopupScreenHandler” 101

4.4.3.54 Class “Rental” 102

4.4.3.55 Class “RentalDefaultCostStrategy” 103

4.4.3.56 Class “RentalDefaultCountingTimeStrategy” 104

4.4.3.57 Class “RentalDefaultDepositStrategy” 105

4.4.3.58 Class “RentalDetailDTO” 105

4.4.3.59 Class “RentalRepo” 106

4.4.3.60 Class “RentalRepoImpl” (implements RentalRepo, ResultSetMappable<Rental>) 107

4.4.3.61 Class “RentalService” 108

4.4.3.62 Class “RentalServiceImpl” (implements RentalService) 109

4.4.3.63 Class “RentalStatus” 110

4.4.3.64 Class “RentalStrategy” 111

4.4.3.65 Class “RentalStrategyRepo” 112

4.4.3.66 Class “RentalStrategyRepoImpl” (implements RentalStrategyRepo, ResultSetMappable<RentalStrategy>) 113

4.4.3.67 Class “RentBikeController” 113

4.4.3.68 Class “RentBikeScreenHandler” (extends BaseScreenHandler) 115

4.4.3.69 Class “ResultScreenHandler” (extends BaseScreenHandler) 116

4.4.3.70 Class “ResultSetMappable” 118

4.4.3.71 Class “ReturnBikeController” 118

4.4.3.72 Class “ReturnBikeScreenHandler”(extends BaseScreenHandler implements ScreenElementObserver) 120

4.4.3.73 Class “ReturnBikeStationListItemHandler” (extends FXMLScreenHandler implements ScreenElementObserver) 121

4.4.3.74 Class “ScreenElementObservable” 122

4.4.3.75 Class “ScreenElementObserver” 123

4.4.3.76 Class “SplashScreenHandler” 124

4.4.3.77 Class “SpringContext” 124

4.4.3.78 Class “StationDetailBikeItemHandler” 125

4.4.3.79 Class “StationDetailScreenHandler” 127

4.4.3.80 Class “SuccessMessage” 128

4.4.3.81 Class “Utils” 129

4.4.3.82 Class “ViewDockController” 130

4.4.3.83 Class “ViewRentalController” 131

4.4.3.84 Class “ViewRentScreenHandler” 132

5 Design Considerations 134

5.1 Goals and Guidelines 134

5.1.1 Goals 134

5.1.2 Guidelines 134

5.2 Architectural Strategies 134

5.3 Coupling and Cohesion 134

5.3.1 Coupling 135

5.3.2 Cohesion 135

5.4 Design Principles 137

5.5 Design Patterns 137

**List of Figures**

No table of figures entries found.

**List of Tables**

No table of figures entries found.

# Introduction

## Objective

This document contains the complete design description of the *EcoBikeRental* application. This includes the architectural features of the system down through details of what operations each code module will perform and the database layout. It also shows how the use cases detailed in the SRS will be implemented in the system using this design.

The primary audiences of this document are the software developers.

## Scope

The software’s goal is to stimulate the managing of customers renting bike and returning bike from docking stations. Four kinds of bicycles are standard bike, twin bike, standard e-bike and twin e-bike, which have different rental costs.

When the app is launched, location of nearby docking stations appear on the map. By selecting dock or searching dock by its name/address, customer can view detailed information of a dock and list of available bikes in that dock and their information. To rent a bike, a customer needs to insert the barcode of the bike into the renting screen, or to choose a bike from the dock, in which case the aeforementioned insertion is done automatically.   
Should the customer choose to rent the bike, they have to deposit an amount equal to 40% of the value of the bike. After confirming the transaction, the   
system will automatically deduct the amount in the customer's card or account and save   
the transaction. Then the user can take the bike for a ride. It should be noted that the same customer can only has a maximum of one bike rented at any time.

While renting, customer can always use the app to view information about the bikes that he or she is renting, including bike type, renting time, the amount to be paid up to now, and bike status (e.g., current battery percentage of e-bike). When customer returns a bike, the system will automatically return the deposit and deduct the amount of money corresponding to the rental period and save transactions.

The system handles validation while displaying corresponding information to the customers. If there is an error, customers will be notified immediately.

## Glossary

**A.**

*Actor:* Users or other systems / programs interacting and exchanging information with the main system / program externally.

**B.**

*Barcode:* a machine-readable code in the form of numbers and a pattern of parallel lines of varying widths, printed on lock for identifying bikes

**C.**

**D.**

*Data*: Facts and statistics collected together for reference or analysis that can be stored and used by a computer

*Docking station*: Place where customers go to rent bikes and return bikes

**E.**

**F.**

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

**G.**

**H.  
I.**

*Information*: is a resolution of uncertainty, which is abstract. Therefore, in order to transmit information, it needs to be converted into 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. Every Java developer must have JRE in order to compile and run a Java program.

**K.**

**L.**

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

**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 look like a standard bike, but has an integrated electric motor for assist propulsion and rental fee costs 1.5 times more expensive than the fee of 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 more expensive than the fee of standard bike

**U.**

*Use case*: A sequence of actions a system performs that yields and 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

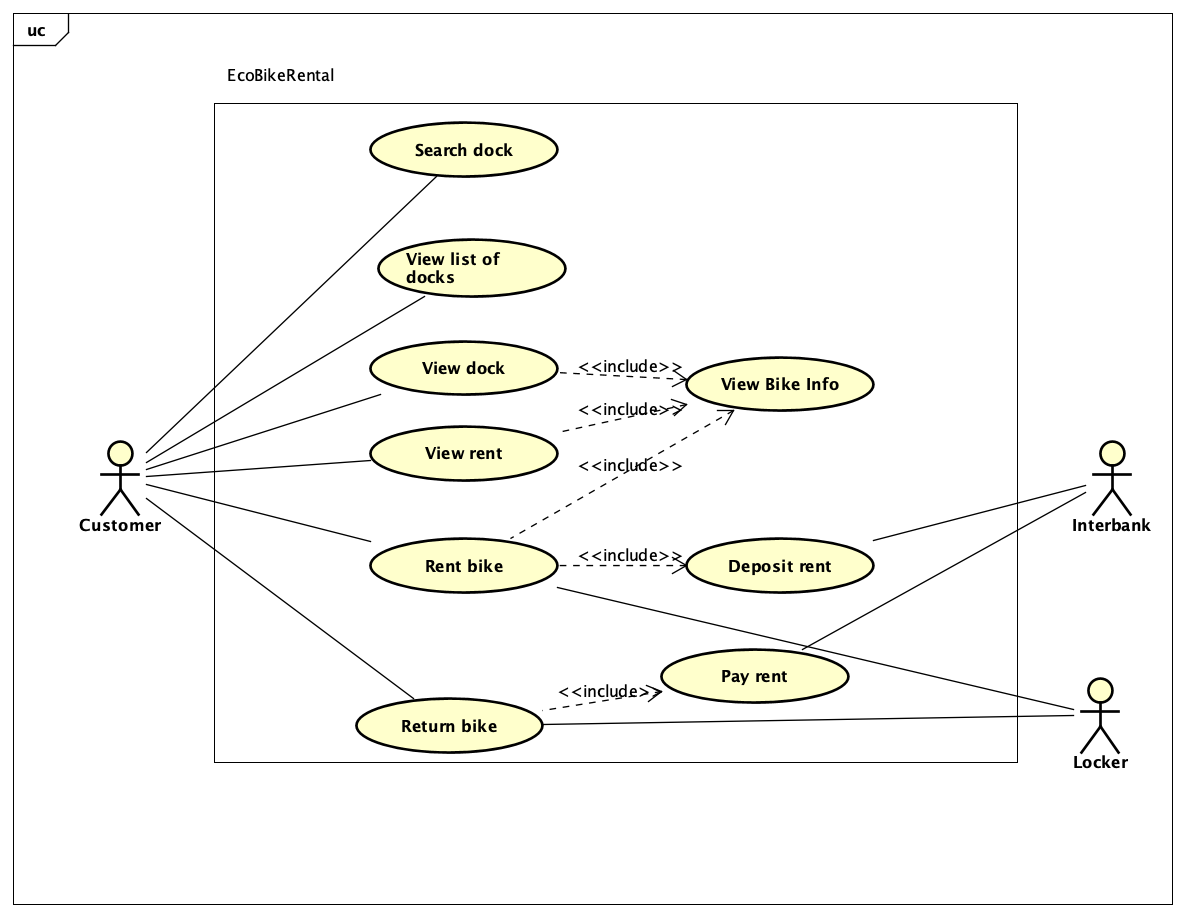
## Trang, N. T. (2019, September). Problem Statement: Eco Bike Rental. Retrieved from: https://www.dropbox.com/sh/2llptvvm9atklen/AADGszPxE-JdkRrsPnaRYeO6a/CapstoneProject?dl=0&preview=EcoBikeRental-ProblemStatement-EN.pdf&subfolder\_nav\_tracking=1

*<Listing the referenced material used in this document, including the one related to the project>*

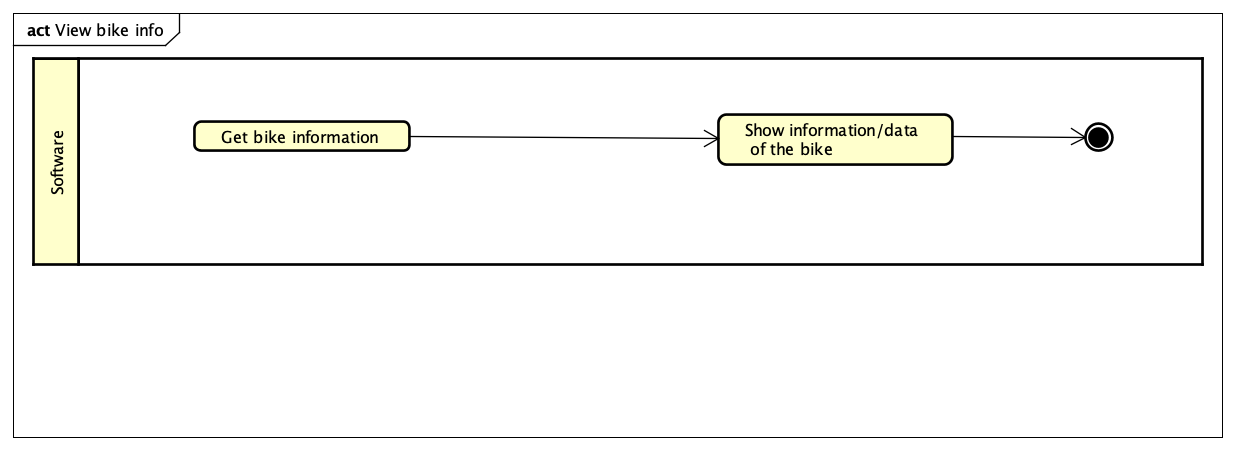
# Overall Description

## General Overview

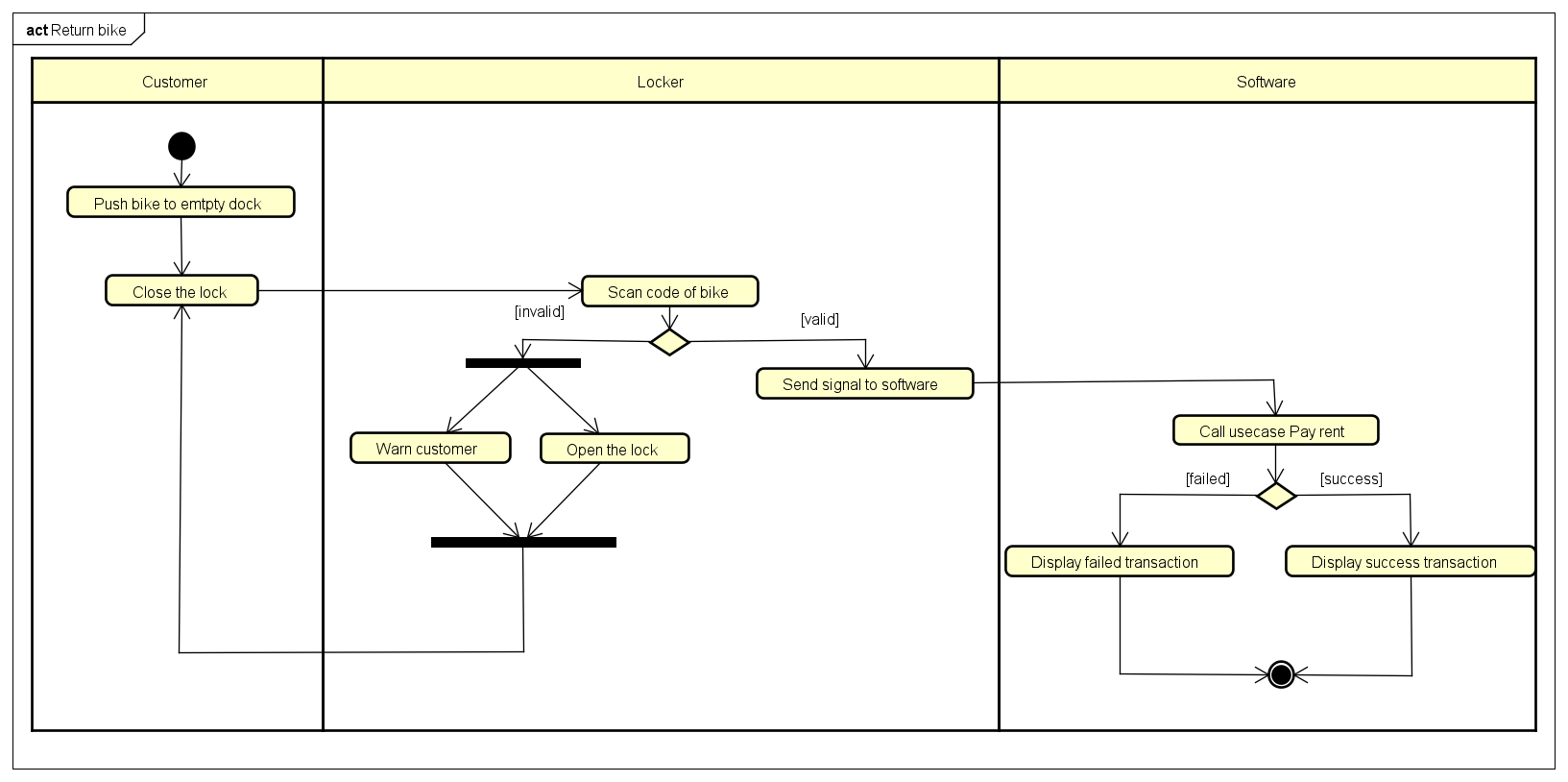
### Usecase diagram



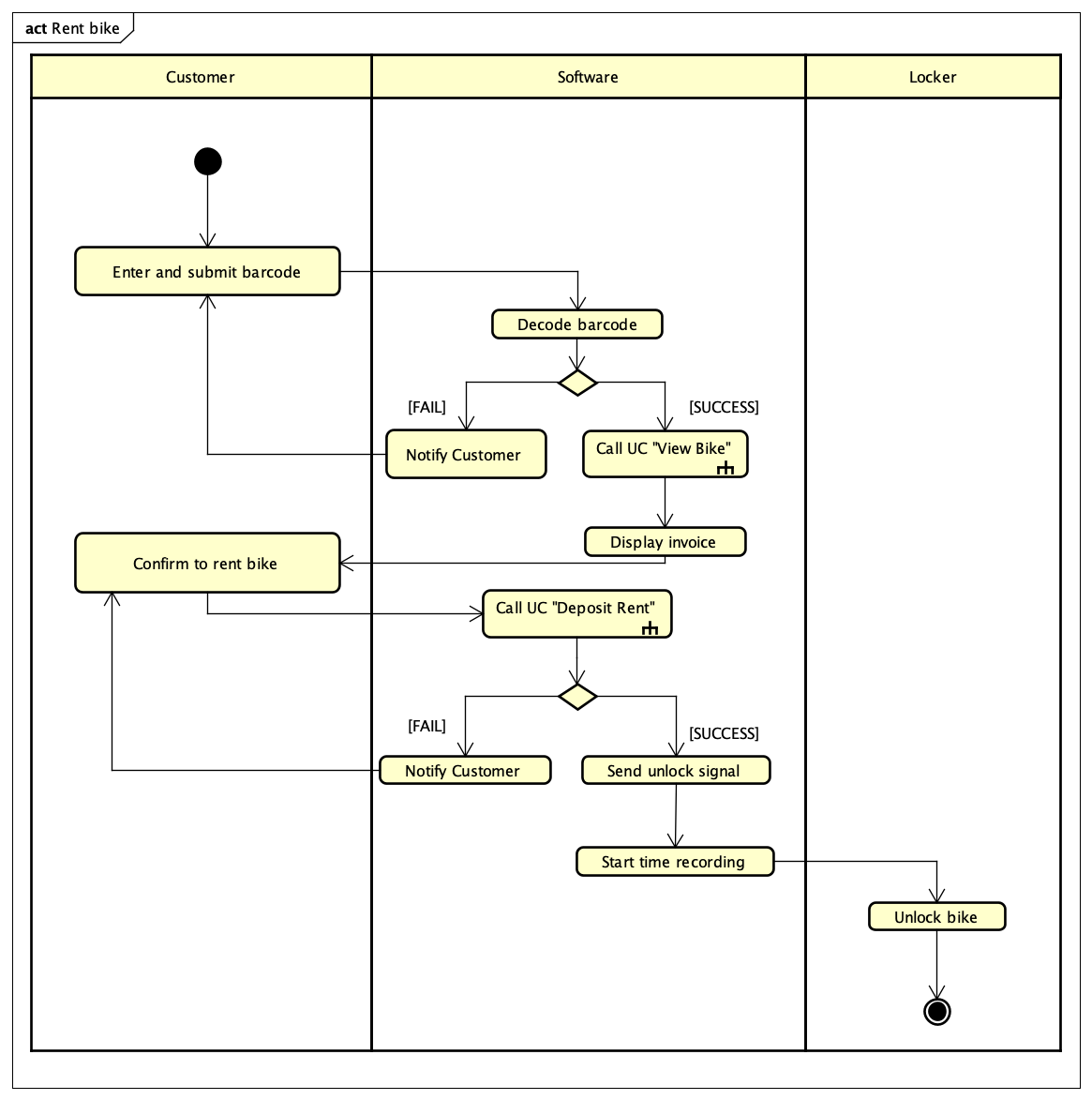
### Usecase “View Bike Info”

****

### Usecase “Return Bike”

****

### Usecase “Rent Bike”

****

### Usecase “Pay Rent”

****

## Assumptions/Constraints/Risks

### Assumptions

*<Describe any assumptions or dependencies regarding the system, software and its use. These may concern such issues as: related software or hardware, operating systems, end-user characteristics, and possible and/or probable changes in functionality>*

### 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*
* *End-user environment*
* *Availability or volatility of resources*
* *Standards compliance*
* *Interoperability requirements*
* *Interface/protocol requirements*
* *Licensing requirements*
* *Data repository and distribution requirements*
* *Security requirements (or other such regulations)*
* *Memory or other capacity limitations*
* *Performance requirements*
* *Network communications*
* *Verification and validation requirements (testing)*
* *Other means of addressing quality goals*
* *Other requirements described in the Requirements Document*

*>*

### Risks

*<Describe any risks associated with the system design and proposed mitigation strategies.>*

# System Architecture and Architecture Design

## Architectural Patterns

The highest cares when designing the project structure is to make role of each component/module as clear as possible, to enhance the code speed and to identify the responsibility of each member during the development of the software. To archive those, each module should contain a solid amount of logic and be independent with each other. Therefore, both horizontal and vertical slices are applied on the project architecture. Horizontal slices, or layering, help improve the abstraction, while vertical slices enhance the agility in coding and make it easier to divide task to members. Thus, we chose multi-tier architecture to implement our project.

To save time of code, we reuse code from the case study project for the views layer and use several libraries. The most important one is Spring core framework. By making use of JavaBeans pattern and Dependency Injection mechanism, Spring framework co-works really efficiently with horizontal slicing to reduce coupling between layers, give dynamic logic in runtime and help abstraction more focused. Additionally, Lombok library is widely used in our projects, removing tons of boiling code related to basic functionalities such as getters, setters, default constructors.

## Interaction Diagrams

### View Dock Info

Diagram

Description automatically generated

### View Rent Info

Diagram

Description automatically generated

### Rent Bike

Table

Description automatically generated

### Return Bike

Diagram, schematic

Description automatically generated

## Analysis Class Diagrams

### View Dock Info

Table

Description automatically generated

### View Rent Info

Diagram

Description automatically generated

### Rent Bike

Diagram

Description automatically generated

### Return Bike

Diagram

Description automatically generated

## 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: Any color possible.

Resolution: 1120 × 640 pixels

**Screen**

Location of standard buttons: At the bottom (vertically) and in the middle (horizontally) of the frame; At the top-right corner of the screen.

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 16px). 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. However, the manual is considered a popup message, as the main screen cannot be operated while the manual screen is shown. After the opening screen, the app will start with splash screen, and then the first screen (home screen) will appear.

Sequences of the system screens:

1. Splash screen (first screen)

2. Home screen – Home station list screen

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

4. Rent new bike screen – input the bike barcode for renting new bike

5. View rental screen – view details of current renting

6. Return bike station list screen – choose a dock station from a list to return bike

7. Invoice screen – view invoice details

8. Result screen – result for payment transaction

9. Pop-up screen – notify the error for users

**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

Diagram

Description automatically generated

### Screen Specifications

#### Home Station List Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Home station list screen | 06/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying searching bar | Initial | Display the search bar to look up for stations based on several fields | |
| Area for displaying the list of stations | Initial | Display the list of stations based on the searching information | |
| My rental button | Click | Display the View Rental screen | |
| View detail button | Click | Display the Station details screen of the corresponding station | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Search bar | 100 | String | Black | Left-justified |
| Name | 30 | String | Black | Left-justified |
| Location | 100 | String | Black | Center-justified |
| No. of available bike | 5 | Numeral | Black | Center-justified |
| Distance | 10 | Numeral | Black | Center-justified |

#### Return Bike Station List Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Return bike station list return screen | 06/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Area for displaying searching bar | Initial | Display the search bar to look up for stations based on several fields | |
| Area for displaying the list of stations | Initial | Display the list of stations based on the searching information | |
| Send bike button | Click | Display the Result screen | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Search bar | 100 | String | Black | Left-justified |
| Name | 30 | String | Black | Left-justified |
| Location | 100 | String | Black | Center-justified |
| No. of empty docking points | 5 | Numeral | Black | Center-justified |
| Distance | 10 | Numeral | Black | Center-justified |

#### Station Details Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Station details screen | 07/11/2021 |  |  | Nguyen Duc Thai |
|  | | **Control** | **Operation** | **Function** | |
| Area for dock information | Initial | Display dock information | |
| Area for bike information | Initial | Display information of bikes in station | |
| Rent button | Click | Direct to Rent new bike screen with bike car code filled in bar code text box | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Station name | 50 | String | Black | Center justified |
| Station address | 50 | String | Black | Center justified |
| Number of empty lock | 5 | Numeral | Black | Center justified |
| Distance | 10 | Numeral | Black | Center justified |
| Number of double bike in dock | 5 | Numeral | Black | Left justified |
| Number of twin e-bike in dock | 5 | Numeral | Black | Left justified |
| Number of standard bike | 5 | Numeral | Black | Left justified |
| Number of standard e-bike | 5 | Numeral | Black | Left justified |
| Bike type | 50 | String | Black | Left justified |
| Bike license plate | 50 | String | Black | Center justified |
| Bike current battery level | 50 | String | Black | Center justified |
| Bike barcode | 50 | String | Black | Left justified |

#### Rent New Bike Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Rent new bike screen | 07/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Bar code text box | Enter from keyboard | Text box to enter bar code of bike | |
| Rent this bike button | Click | Rent the bike with bar code in text box | |
| Rental option box | Click | Options for renting new bike | |
| Reset button | Click | Reset bar code text box | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Barcode | 50 | String | Black | Left justified |

#### View Rental Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | View rental screen | 08/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Area for renting information | Initial | Display renting information | |
| Area for bike image | Initial | Display bikes image | |
| Return button | Click | Direct to Rent dock list return screen | |
| Pause button | Click | Pause the rental time | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Bike type | 50 | String | Black | Center justified |
| Lisence Plate | 50 | String | Black | Center justified |
| Current battery life | 20 | Numeral | Black | Center justified |
| Start time | 50 | Datetime | Black | Center justified |
| Deposit amount | 20 | Numeral | Black | Right justified |
| Rent amount | 20 | Numeral | Black | Right justified |
| Valid time of rent amount | 50 | Datetime | Black | Center justified |

#### Invoice Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Invoice screen | 08/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Area for renting information | Initial | Display renting information | |
| Filter area | Check boxes | Display type of card | |
| Area for card information | Initial | Display card information | |
| Deposit button | Click | Direct to result screen | |
| Back button | Click | Return to previous screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| Bike barcode | 50 | String | Black | Center justified |
| Bike type | 50 | String | Black | Center justified |
| Lisence Plate | 50 | String | Black | Center justified |
| Current battery life | 20 | Numeral | Black | Center justified |
| Dock name | 50 | String | Black | Center justified |
| Deposit | 20 | Numeral | Black | Right justified |

#### Splash Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Splash screen | 07/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Loading gif | Initial | Display the loading state | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item name** | **Number of digits (bytes)** | **Type** | **Field attribute** | **Remark** |
| (NONE) |  |  |  |  |

#### Result Screen

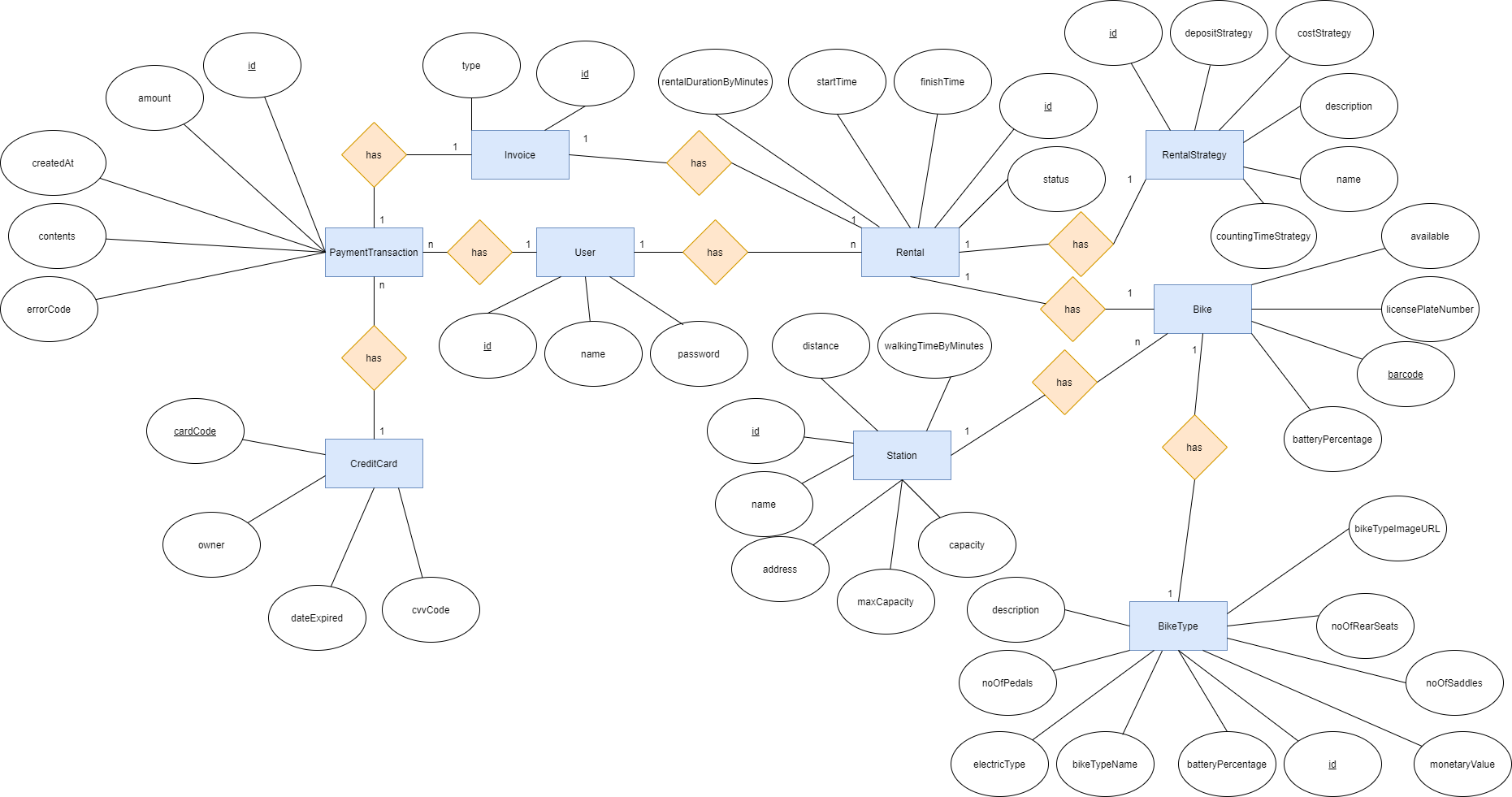
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Result screen | 08/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Area for result information | Initial | Display result of renting | |
| OK button | Click | Direct to view rent screen | |
| Home button | Click | Direct to view dock list screen | |

#### Pop-up screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| Screen specification | Pop-up screen | 08/11/2021 |  |  | Bui Thanh Tung |
|  | | **Control** | **Operation** | **Function** | |
| Area for result information | Initial | Display error | |
|  |  |  | |
|  |  |  | |

## Data Modeling

### Conceptual Data Modeling



* A bike has a distinct barcode, as well as a license plate number, the battery percentage (if it is an electrical bike) and its availability. A bike must belong to one of several bike types, each one with their own id, number of saddles, pedals and rear seats, monetary value, A bike type also contains information on whether or not the type consists of electrical bikes. A possible preview image representing the bike type, as well as a description, is very much appreciated, although not mandatory.
* A station has a distinct ID, a name and address. The station has two attributes: max capacity and capacity which represent the maximum number of bikes it could hold and the current number of bikes inside, respectively. For simulation purpose, a station also contains distance from current location and walking time to the station.
* A user has an ID, the username and password of the account. One user can initiate many rentals, consist of ID, rental duration, start and finish time, status as well as a rental strategy. The rental strategy, with an ID, name, and most crucially deposit and cost strategy, dictates how the deposit, cost and time of the rental would be calculated.
* A user can also initiate many payment transactions, make up of ID, transaction amount, message contents, error code and creation time. Each transaction uses a credit card, composes of distinct card code. Credit cards also retain information on card’s owner, expiration date and cvv code. Finally, the invoice, with invoice’s ID and invoice type (pay or refund), contains information of the transaction and the rental.

### Database Design

#### Database Management Systems

This project utilize relational database for the following reasons:

* A relational system is better suited to represent the relations between entities.
* Relational DBMS is a good choice in case of multiple tables query, which happens within the application process.
* The designers have had some experiences working with relational databse, so the choice was made to minimize errors and to improve performance.

#### Logical Data Model

a) Database design process from E-R diagram:

AppUser(id, username, password)

BikeType(id, bikeTypeName, monetaryValue, electricType, noOfSaddles, noOfPedals, noOfRearSeats, description, bikeTypeImageURL)

Station(id, name, address, distance, maxCapaccity, capacity, walkingTimeByMinutes)

Bike(barcode, *type*, *location*, licensePlateNumber, available, batteryPercentage)

RentalStrategy(id, depositStrategy, costStrategy, description, name, countingTimeStrategy)

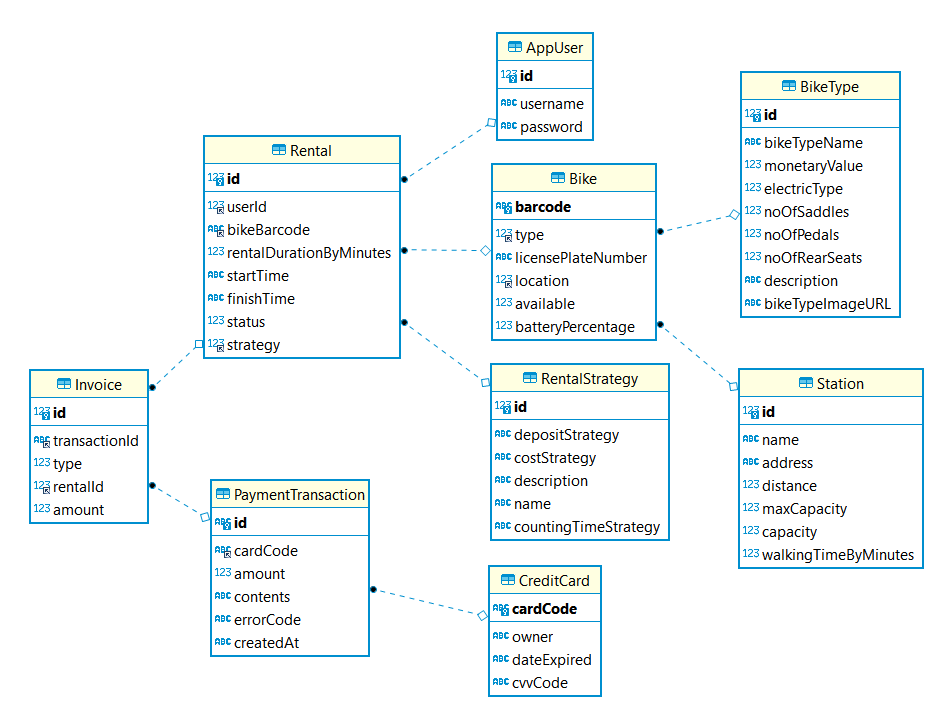
Rental(id, *userId*, *bikeBarcode*, *strategy*¸ rentalDurationByMinutes, startTime, finishTime, status)

CreditCard(cardCode, owner, dateExpited, cvvCode)

PaymentTransaction(id, *cardCode*, amount, contents, errorCode, createdAt)

Invoice(id, *transactionId*, *rentalId*, type)

b) Design result:



#### Physical Data Model

**AppUser**

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

**Bike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | barcode | Varchar(45) | Yes | Barcode of bike |
| 2 |  | X | type | Integer | Yes | ID of bike’s BikeType |
| 3 |  |  | licensePlateNumber | Varchar(45) | Yes | Bike’s license plate number |
| 4 |  | X | location | Integer | No | The station bike is in if not rented |
| 5 |  |  | batteryPercentage | Integer | Yes | Battery percentage of bike, always 0 if bike is not electric |
| 6 |  |  | available | Boolean | Yes | If bike is available to rent |

**BikeType**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | id | Integer | Yes | ID, auto increment |
| 2 |  |  | bikeTypeName | Varchar(45) | Yes | Name of bike type |
| 3 |  |  | monetaryValue | Integer | Yes | Value of bike type in VND |
| 4 |  |  | electricType | Boolean | Yes | Indicate if bike type is electrical |
| 5 |  |  | noOfSaddles | Integer | Yes | Number of saddles on this type |
| 6 |  |  | noOfPedals | Integer | Yes | Number of pedals on this type |
| 7 |  |  | noOfRearSeats | Integer | Yes | Number of rear seats on this type |
| 8 |  |  | description | Varchar(100) | No | A simple description of the type |
| 9 |  |  | bikeTypeImageURL | Varchar(100) | No | Link to the preview image of bike type |

**CreditCard**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | cardCode | Varchar(45) | Yes | Code of card |
| 2 |  |  | owner | Varchar(45) | Yes | Name of card’s owner |
| 3 |  |  | dateExpired | Datetime | Yes | Expiration date of card |
| 4 |  |  | cvvCode | Varchar(45) | Yes | Security code of card |

**Invoice**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | id | Integer | Yes | ID, auto increment |
| 2 |  |  | transactionId | Integer | Yes | ID of corresponding Payment Transaction |
| 3 |  |  | type | Integer | Yes | Type of the Invoice (Pay or Refund) |
| 4 |  |  | rentalId | Integer | Yes | ID of corresponding Rental |
| 5 |  |  | amount | Integer | Yes | Amount of money to pay or refund |

**PaymentTransaction**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | id | Varchar(45) | Yes | ID, auto increment |
| 2 |  | X | cardCode | Varchar(45) | Yes | Code of the card used for the Transaction |
| 3 |  |  | amount | Integer | Yes | Amount paid |
| 4 |  |  | contents | Varchar(100) | Yes | Message of the Transaction |
| 5 |  |  | errorCode | Varchar(100) | Yes | Error code if error happened |
| 6 |  |  | createdAt | Varchar(45) | Yes | Creation time of the Transaction |

**Rental**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | id | Integer | Yes | ID, auto increment |
| 2 |  | X | userId | Integer | Yes | Corresponding User of the Rental |
| 3 |  | X | bikeBarcode | Varchar(45) | Yes | Corresponding Bike of the Rental |
| 4 |  |  | rentalDurationByMinutes | Integer | Yes | Current Rental duration |
| 5 |  |  | startTime | Varchar(45) | Yes | Starting time of the Rental |
| 6 |  |  | finishTime | Varchar(45) | Yes | Finishing time of the Rental |
| 7 |  |  | status | Integer | Yes | Status of the Rental |
| 8 |  | X | strategy | Integer | Yes | Strategy employed by the Rental |

**RentalStrategy**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | id | Integer | Yes | ID, auto increment |
| 2 |  |  | depositStrategy | Varchar(45) | Yes | Strategy for deposit calculation |
| 3 |  |  | costStrategy | Varchar(45) | Yes | Strategy for cost calculation |
| 4 |  |  | description | Varchar(45) | No | Strategy’s description |
| 5 |  |  | name | Varchar(45) | Yes | Startegy’s name |
| 6 |  |  | countingTimeStrategy | Varchar(45) | Yes | Strategy for time calculation |

**Station**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column Name** | **Data Type** | **Mandatory** | **Description** |
| 1 | X |  | id | Integer | Yes | ID, auto increment |
| 2 |  |  | name | Varchar(45) | Yes | Station’s name |
| 3 |  |  | address | Varchar(100) | Yes | Station’s address |
| 4 |  |  | distance | Integer | Yes | Station’s distance to current location |
| 5 |  |  | maxCapacity | Integer | Yes | Max capacity of Station |
| 6 |  |  | capacity | Integer | Yes | Number of Bike currently in Station |
| 7 |  |  | walkingTimeByMinutes | Integer | Yes | Walking time to Station |

**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 "BikeType" (

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

"bikeTypeName" VARCHAR(45) NOT NULL,

"monetaryValue" INTEGER NOT NULL,

"electricType" BOOLEAN NOT NULL,

"noOfSaddles" INTEGER NOT NULL,

"noOfPedals" INTEGER NOT NULL,

"noOfRearSeats" INTEGER NOT NULL,

"description" VARCHAR(100),

"bikeTypeImageURL" VARCHAR(100)

);

CREATE UNIQUE INDEX "BikeType.FK\_BikeType\_bikeTypeName1\_idx" ON "BikeType" ("bikeTypeName");

CREATE TABLE "Station"(

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

"name" VARCHAR(45) NOT NULL,

"address" VARCHAR(100) NOT NULL,

"distance" INTEGER NOT NULL,

"maxCapacity" INTEGER NOT NULL,

"capacity" INTEGER NOT NULL,

"walkingTimeByMinutes" INTEGER NOT NULL

);

CREATE FULLTEXT INDEX "Station.FK\_Station\_name1\_idx" ON "Station" ("name");

CREATE FULLTEXT INDEX "Station.FK\_Station\_address1\_idx" ON "Station" ("address");

CREATE TABLE "Bike"(

"barcode" VARCHAR(45) PRIMARY KEY NOT NULL,

"type" INTEGER NOT NULL,

"licensePlateNumber" VARCHAR(45) NOT NULL,

"location" INTEGER,

"available" BOOLEAN DEFAULT TRUE,

"batteryPercentage" INTEGER DEFAULT 0 CHECK (batteryPercentage >= 0 AND batteryPercentage <= 100),

CONSTRAINT "fk\_Bike\_Station1" FOREIGN KEY ("location") REFERENCES Station("id"),

CONSTRAINT "fk\_Bike\_BikeType1" FOREIGN KEY ("type") REFERENCES BikeType("id")

);

CREATE TABLE "RentalStrategy"(

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

"depositStrategy" VARCHAR(45) NOT NULL,

"costStrategy" VARCHAR(45) NOT NULL,

"description" VARCHAR(45),

"name" VARCHAR(45) NOT NULL,

"countingTimeStrategy" VARCHAR(45) NOT NULL

)

CREATE TABLE "Rental"(

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

"userId" INTEGER NOT NULL,

"bikeBarcode" VARCHAR(45) NOT NULL,

"rentalDurationByMinutes" INTEGER DEFAULT 0,

"startTime" VARCHAR(45) NOT NULL,

"finishTime" VARCHAR(45),

"status" INTEGER DEFAULT 0,

"strategy" INTEGER NOT NULL,

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

CONSTRAINT "fk\_Rental\_Bike1"FOREIGN KEY("bikeBarcode") REFERENCES "Bike"("barcode"),

CONSTRAINT "fk\_Rental\_RentalStrategy1"FOREIGN KEY("strategy") REFERENCES "RentalStrategy"("id")

);

CREATE TABLE "CreditCard"(

"cardCode" VARCHAR(45) PRIMARY KEY NOT NULL,

"owner" VARCHAR(45) NOT NULL,

"dateExpired" DATE NOT NULL,

"cvvCode" VARCHAR(45) NOT NULL

);

CREATE TABLE "PaymentTransaction"(

"id" VARCHAR(45) PRIMARY KEY NOT NULL,

"cardCode" VARCHAR(45) NOT NULL,

"amount" INTEGER NOT NULL,

"contents" VARCHAR(100) NOT NULL,

"errorCode" VARCHAR(100) NOT NULL,

"createdAt" VARCHAR(45) NOT NULL,

CONSTRAINT "fk\_PaymentTransaction\_CreditCard1" FOREIGN KEY ("cardCode") REFERENCES "CreditCard"("cardCode")

);

CREATE TABLE "Invoice"(

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

"transactionId" INTEGER DEFAULT NULL,

"type" INTEGER NOT NULL CHECK (type = 1 OR type = -1),

"rentalId" INTEGER NOT NULL,

"amount" INTEGER NOT NULL,

CONSTRAINT "fk\_Invoice\_Rental1" FOREIGN KEY ("rentalId") REFERENCES "Rental"("id"),

CONSTRAINT "fk\_Invoice\_PaymentTransaction1"FOREIGN KEY("transactionId")REFERENCES "PaymentTransaction"("id")

);

CREATE UNIQUE INDEX "Invoice.FK\_Invoice\_typeRentalId1\_idx" ON "Invoice" ("type", "rentalId");

## Non-Database Management System Files

*<Provide the detailed description of all non-DBMS files if any and include a narrative description of the usage of each file that identifies if the file is used for input, output, or both, and if the file is a temporary file. Also provide an indication of which modules read and write the file and include file structures (refer to the data dictionary). As appropriate, the file structure information should include the following:*

*• Record structures, record keys or indexes, and data elements referenced within the records*

*• Record length (fixed or maximum variable length) and blocking factors*

*• Access method (e.g., index sequential, virtual sequential, random access, etc.)*

*• Estimate of the file size or volume of data within the file, including overhead resulting from file access methods*

*• Definition of the update frequency of the file (If the file is part of an online transaction-based system, provide the estimated number of transactions per unit of time, and the statistical mean, mode, and distribution of those transactions.)*

*• Backup and recovery specifications>*

This project employs a system of backup and recovery database whose fundametal purpose is to test various behaviours of the system. They include the files: AppUser.csv, Bike.csv, BikeType.csv, Rental.csv, Station.csv located in the path src/main/resources/db/backup/

These csv files are imported into the database using the import functionality of dbeaver.

## Class Design

### General Class Diagram

Diagram

Description automatically generated

### Class Diagrams

#### Class Diagram for Package “Controller”

Diagram, schematic

Description automatically generated

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

Diagram

Description automatically generated

#### Class Diagram for Package “Bike”

Diagram, schematic

Description automatically generated

#### Class Diagram for Package “Payment”

Diagram, engineering drawing

Description automatically generated

#### Class Diagram for Package “Station”

Diagram

Description automatically generated

#### Class Diagram for Package “Subsystem”

Diagram

Description automatically generated

#### Class Diagram for Package “Repository”

Diagram

Description automatically generated

#### Class Diagram for Package “Rental”

Diagram

Description automatically generated

#### Class Diagram for Package “Common”

Graphical user interface

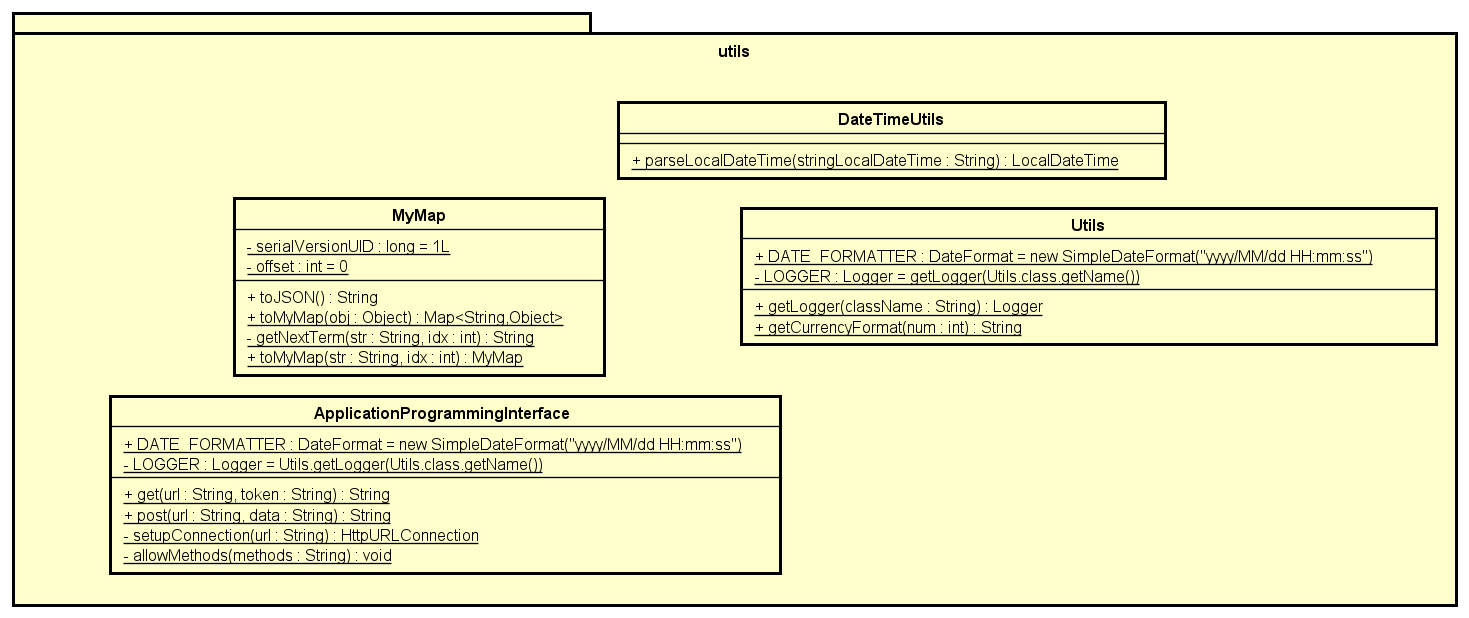
Description automatically generated

#### Class Diagram for Package “User”

Diagram

Description automatically generated

#### Class Diagram for Package “Utils”



### Class Design

#### Class “ApplicationProgrammingInterface”

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | DATE\_FORMATTER | DateFormat |  | Set new format for date |
| 2 | LOGGER | Logger |  | Used for logging |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | get | String | Get API |
| 2 | post | String | Post API |
| 3 | allowMethods | void | Allow methods |

*Parameter:*

* url
* token: mock data for authentication
* data: data for post API

*Exception:*

* IOException
* Exception

**Method**

None

**State**

None

#### Class “ViewsConfigs”

Text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | IMAGE\_PATH | String |  | Path to images |
| 2 | SPLASH\_SCREEN\_PATH | String |  | Path to splash screen fxml |
| 3 | RENT\_BIKE\_SCREEN\_PATH | String |  | Path to rent bike screen fxml |
| 4 | HOME\_DOCK\_LIST\_SCREEN\_PATH | String |  | Path to gome dock list screen fxml |
| 5 | VIEW\_RENTAL\_SCREEN\_PATH | String |  | Path to view rental screen fxml |
| 6 | INVOICE\_SCREEN\_PATH | String |  | Path to invoice screen fxml |
| 7 | DOCK\_DETAIL\_SCREEN\_PATH | String |  | Path to dock detail screen fxml |
| 8 | RETURN\_BIKE\_DOCK\_LIST\_PATH | String |  | Path to return bike dock list screen fxml |
| 9 | RESULT\_SCREEN\_PATH | String |  | Path to result screen fxml |
| 10 | POPUP\_PATH | String |  | Path to popup screen fxml |
| 11 | HOME\_STATION\_ITEM | String |  | Path to station item fxml |
| 12 | RETURN\_BIKE\_STATION\_ITEM | String |  | Path to return bike station item fxml |
| 13 | STATION\_DETAIL\_BIKE\_ITEM | String |  | Path to station detail bike items fxml |

**Operation**

None

*Parameter:*

*NONE*

*Exception:*

* None

**Method**

None

**State**

None

#### Class “AppUser”

Text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | appUserId | Integer | NULL | user’s ID |
| 2 | name | String | NULL | User’s name |
| 3 | password | String | NULL | User’s password |
| 4 | rentalHistory | List<Rental> | NULL | Rental history of current user |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | AppUser |  | Constructor for AppUser class |

*Parameter:*

None

*Exception:*

* None

**Method**

None

**State**

None

#### Class “AppUserRepo” (extends interface BaseRepo)

A picture containing text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | insert | AppUser | Insert user to database |

*Parameter:*

* user: app user

*Exception:*

None

**Method:**

None

**State:**

None

#### Class “AppUserRepoImpl” (implements AppUserRepo, ResultSetMappable)

Text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | insert | AppUser | Insert user to database |
| 2 | getById | AppUser | Get user by id |
| 3 | mapFromResultSet | AppUser | Convert result set to AppUser |

*Parameter:*

* res: result set

*Exception:*

* None

**Method:**

None

**State:**

None

#### Class “AppUserService”

Text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | decodeAuthenticationToken | Integer | Decode the authentication token |
| 2 | getById | AppUser | Get user by id |

*Parameter:*

* authenticationToken
* userId: user’s id

*Exception:*

None

**Method**

None

**State**

None

#### Class “AppUserServiceImpl” (implements AppUserService)

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | appUserRepo | AppUserRepo | Initialized inside constructor | App user repository |

**Operation**

Override interface operations.

*Parameter:*

None

*Exception:*

None

**Method**

None

**State**

None

#### Class “BaseRepo”

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | DATA\_SOURCE | DataSource |  | Data resource |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | getById | Void | get the object from it’s ID |

*Parameter:*

id: Object key

*Exception:*

* DataSourceException: exception for data source.

**Method:**

* NONE

**State:**

NONE

#### Class “BaseScreenHandler”

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | LOGGER | Logger |  | used for logging |
| 2 | authenticationToken | String |  | authentication token |
| 3 | scene | Scene |  | the scene |
| 4 | prev | BaseScreenHandler |  | previous screen |
| 5 | stage | Stage |  | stage |
| 6 | homeScreenHandler | homeScreenHandler |  | home screen handler |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | BaseScreenHandler |  | Constructor |
| 2 | setPreviousScreen | Void | set the previous screen |
| 3 | getPreviousScreen | Void | get the previous screen |
| 4 | init | void | initialize a screen |
| 5 | show | Void | show scene |
| 6 | setScreenTitle | Void | set the title for screen |
| 7 | setHomeScreenHandler | Void | set home screen handler |
| 8 | goBack | Void | show previous screen |
| 9 | goHome | Void | show the home screen |

*Parameter:*

* screenPath: path to the screen
* prev: previous screen
* stage

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “Bike”

A picture containing text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | barcode | String | NULL | barcode of the bike |
| 2 | licensePlateNumber | String | Null | bike lisence plate number |
| 3 | available | Boolean | Null | the availability of renting that bike |
| 4 | batteryPercentage | Integer | Null | battery level of bike |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | Bike |  | contructor |
| 2 | isAvailable | Boolean | check if bike is available for rent |
| 3 | getLocation | DockStation | get the location of the bike |

*Parameter:*

None

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “BikeRepo” (extends BaseRepo)

Text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getAllByLocation | List<Bike> | get the bikes by location |
| 2 | updateLocation | Bike | update the location of a bike |

*Parameter:*

* stationId: dock station ID
* barcode: barcode of the bike

*Exception:*

* NONE

**Method:**

* NONE

**State:**

* NONE

#### Class “BikeRepoImpl” ( implements BikeRepo, ResultSetMappable)

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | bikeTypeRepo | BikeTypeRepo | Injected from IoC | bike type repository |
| 2 | dockStationRepo | String | Null | bike lisence plate number |

**Operation**

Overrides interface operations

*Parameter:*

* NONE

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “BikeService”

A picture containing text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | getBikeIfAvailable | Bike | Get the bike if it is available for rent |
| 2 | getAllByLocation | List<Bike> | get all the bike by location |
| 3 | updateRentedBikeLocation | Bike | update the bike location after being rented |
| 4 | updateReturnedBikeLocation | Bike | update bike location after being returned |

*Parameter:*

* barcode: bike barcode
* stationId: Id of dock station

*Exception:*

* UnavailableBikeException: if the bike is not available for renting
* DataSourceException

**Method:**

* NONE

**State:**

NONE

#### Class “BikeServiceImpl” (implements BikeService)

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | bikeRepo | BikeRepo | Initialized inside constructor | bike repository |
| 2 | bikeTypeRepo | BikeTypeRepo | Initialized inside constructor | bike type repository |

**Operation**

Overrides interface operations.

*Parameter:*

* NONE

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “BikeType”

Text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | bikeTypeId | Integer | Null | bike type id |
| 2 | bikeTypeName | String | Null | bike type name |
| 3 | monetaryValue | Integer | Null | monetary value |
| 4 | electricType | Boolean | Null | whether the bike is electric bike |
| 5 | noOfSaddles | Integer | Null | number of saddles |
| 6 | noOfPedals | Integer | Null | number of pedals |
| 7 | noOfRearSeats | Integer | Null | number of rear seats |
| 8 | description | String | Null | description for bike |
| 9 | bikeTypeImageURL | String | Null | image url for bike type |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | isElectricType | Boolean | check if the bike is electric or not |

*Parameter:*

* bikeTypeId: bike type id

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “BikeTypeRepo” (extends BaseRepo)

Text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | getAll | List<BikeType> | get all bike type |

*Parameter:*

* NONE

*Exception:*

* DataSourceException

**Method:**

* NONE

**State:**

NONE

#### Class “BikeTypeRepoImpl” (implements BikeTypeRepo, ResultSetMappable)

Text

Description automatically generated

**Attribute**

None

**Operation**

Overrides interface operations.

*Parameter:*

*Exception:*

* DataSourceException
* SQLException

**Method:**

* NONE

**State:**

NONE

#### Class “CreditCard”

Text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data Type* | *Default Value* | *Description* |
| 1 | cardCode | String | Null | card code |
| 2 | owner | String | Null | owner of the card |
| 3 | cvvCode | Integer | Null | cvv code of card |
| 4 | dateExpired | String | Null | date expired of card |

**Operation**

None

*Parameter:*

None

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “CreditCardRepo” (extends BaseRepo)

Text

Description automatically generated

**Attribute**

* NONE

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | insert | CreditCard | insert card to database |
| 2 | update | CreditCard | update card in database |

*Parameter:*

* NONE

*Exception:*

* DataSourceException

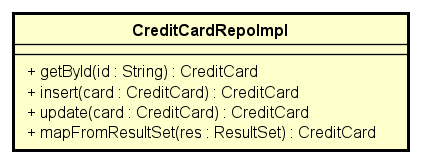
**Method:**

* NONE

**State:**

NONE

#### Class “CreditCardRepoImpl” (implements CreditCardRepo, ResultSetMappable)



**Attribute**

* NONE

**Operation**

Overrides interface operations.

*Parameter:*

* NONE

*Exception:*

* DataSourceException
* SQLException

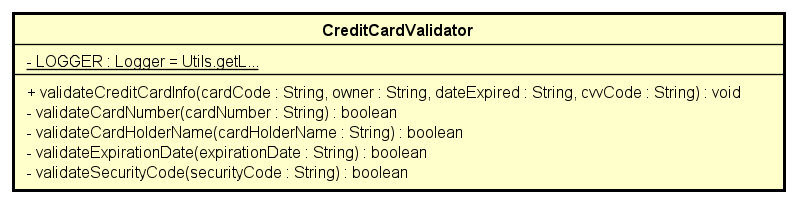
**Method:**

* NONE

**State:**

NONE

#### Class “CreditCardValidator”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | LOGGER | Logger | Null | used for logging |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | validateCreditCardInfo | Void | validate the card information |

Parameter:

* cardCode: code of card
* owner: owner of card
* dateExpired: expiration date of card
* cvvCode: cvv code of card
* cardNumber: card number
* cardHolderName: card holder name
* securityCode: security code of the bike

Exception:

* InvalidCardException: if the input card information is invalid

**Method:**

* validateCardNumber: check card number in format 121319\_groupSTT\_2021
* validateCardHolderName: check card holder name format
* validateExpirationDate: check expiration date in format mm/yy

**State:**

None

#### Class “DataSource”

**A picture containing table

Description automatically generated**

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | connection | Connection | Null | connection to database |
| 2 | transactional | Boolean | Null | whether in transaction or not |
| 3 | INSTANCE | DataSource | Null | singleton instance |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getConnection | Connection | get connection to database |
| 2 | enableTransaction | void | enable transaction |
| 3 | disableTransaction | void | disable transaction |

Parameter:

None

Exception:

- DataSourceException

**Method:**

None

**State:**

None

#### Class “DataSourceConfigs”

Text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | DB\_URL | String |  | database url |
| 2 | DB\_USERNAME | String |  | username for database |
| 3 | DB\_PASSWORD | String |  | password for database |

**Operation**

None

Parameter:

None

Exception:

- DataSourceException

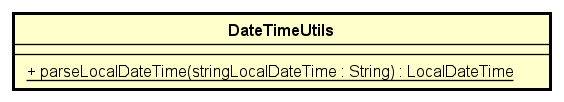
**Method:**

None

**State:**

None

#### Class “DateTimeUtils”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | parseLocalDateTime | LocalDateTime | parse string to date |

Parameter:

None

Exception:

None

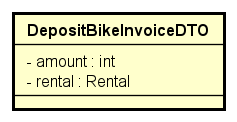
**Method:**

None

**State:**

None

#### Class “DepositBikeInvoiceDTO”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | amount | Integer | Null | amount to deposit |
| 2 | rental | Rental | Null | rental information |

**Operation**

- None

*Parameter:*

* NONE

*Exception:*

* NONE

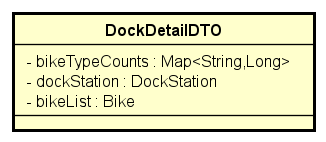
**Method:**

* NONE

**State:**

NONE

#### Class “DockDetailDTO”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | bikeTypeCounts | Map<String, Long> | Null | bike types and quantity |
| 2 | dockStation | DockStation | Null | dock station |
| 3 | bikeList | List<Bike> | Null | bike list |

**Operation**

* NONE

*Parameter:*

* NONE

*Exception:*

* NONE

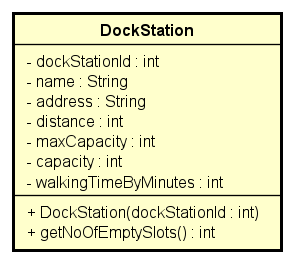
**Method:**

* NONE

**State:**

NONE

#### Class “DockStation”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | dockStationId | Int | Null | dock station id |
| 2 | name | String | Null | name of dock station |
| 3 | address | String | Null | address of dock station |
| 4 | distance | Int | Null | distance to dock station |
| 5 | maxCapacity | Int | Null | max capacity of the dock |
| 6 | capacity | Int | Null | current capacity of the dock |
| 7 | walkingTimeByMinutes | Int | Null | walking time by minutes |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | DockStation |  | constructor |
| 2 | getNoOfEmptySlots | Int | Get the number of empty slots |

*Parameter:*

* NONE

*Exception:*

* NONE

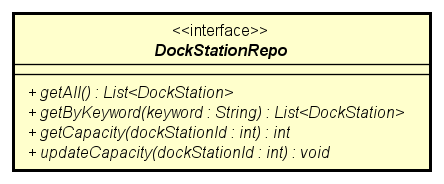
**Method:**

* NONE

**State:**

NONE

#### Class “DockStationRepo” (extends BaseRepo)



**Attribute**

**None**

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getAll | List<DockStation> | get all dock station |
| 2 | getByKeyword | List<DockStation> | get dock station by keyword |
| 3 | getCapacity | Int | get capacity of dock station by id |
| 4 | updateCapacity | void | update capacity of dock station by id |

*Parameter:*

* dockStationId: dock station id

*Exception:*

* DataSourceException

**Method:**

* NONE

**State:**

NONE

#### Class “DockStationRepoImpl”

Text

Description automatically generated

**Attribute**

None

**Operation**

Overrides interface operations.

*Parameter:*

* NONE

*Exception:*

* NONE

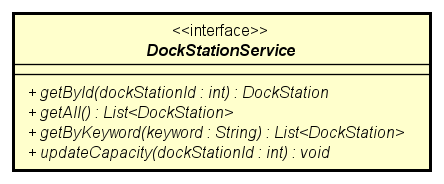
**Method:**

* NONE

**State:**

NONE

#### Class “DockStationService”



**Attribute**

* NONE

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | getById | DockStation | get dock station by id |
| 2 | getAll | List<DockStation> | get all dock station |
| 3 | getByKeyWord | Int | get dock station by keyword |
| 4 | updateCapacity | void | update capacity of dock station by id |

*Parameter:*

* dockStationId: the id of dock station
* keyword: keyword to search for dock station

*Exception:*

* DataSourceException

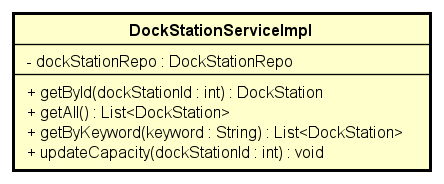
**Method:**

* NONE

**State:**

NONE

#### Class “DockStationServiceImpl”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | dockStationRepo | DockStationRepo | Initialized inside constructor | dock station repository |

**Operation**

Overrides interface operations.

*Parameter:*

* NONE

*Exception:*

* NONE

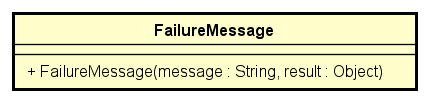
**Method:**

* NONE

**State:**

NONE

#### Class “FailureMessage” (extends Message)



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | FailureMessage |  | constructor |

*Parameter:*

* message
* result

*Exception:*

* NONE

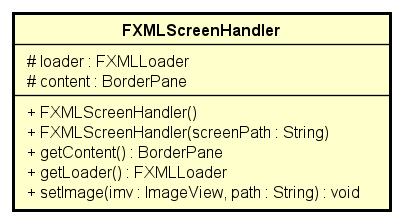
**Method:**

* NONE

**State:**

NONE

#### Class “FXMLScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | loader | FXMLLoader |  | loader |
| 2 | content | BorderPane |  | content |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | getContent | BorderPane | get content |
| 2 | getLoader | FXMLLoader | get loader |
| 3 | setImage | void | set image |

*Parameter:*

* imv: imageView
* path: screen path

*Exception:*

* NONE

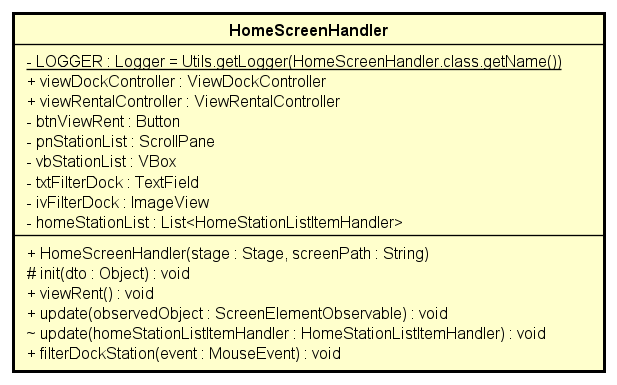
**Method:**

* NONE

**State:**

NONE

#### Class “HomeScreenHandler” (extends BaseScreenHandler implements ScreenElementObserver)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger |  | Used for logging |
| 2 | viewDockController | ViewDockController | Initialized inside constructor | view dock controller |
| 3 | viewRentalController | ViewRentalController | Initialized inside constructor | view rental controller |
| 4 | btnViewRent | Button |  | button view rent |
| 5 | pnStationList | ScrollPane |  | scrollpane station list |
| 6 | vbStationList | VBox |  | VBox station list |
| 7 | txtFilterDock | TextField |  | text field filter dock |
| 8 | ivFilterDock | ImageView |  | image view filter dock |
| 9 | homeStationList | List<HomeStationListItemHandler> |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | viewRent | Void | view rent |
| 2 | update | Void | update home station list handler |
| 3 | filterDockStation | void | filter dock station |

*Parameter:*

* observedObject
* homeStationListItemHandler: handler for home station list item

*Exception:*

* DataSourceException
* IOException

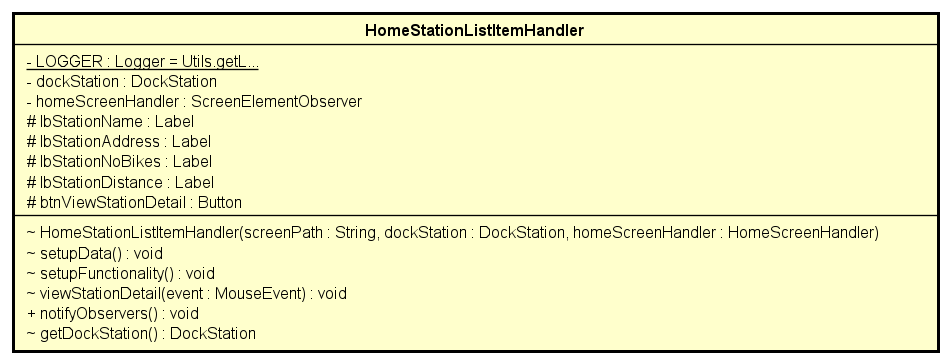
**Method:**

* NONE

**State:**

NONE

#### Class “HomeStationListItemHandler”( extends FXMLScreenHandler implements ScreenElementObservable)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger |  | Used for logging |
| 2 | dockStation | DockStation | Null | dock station |
| 3 | homeScreenHandler | ScreenElementObserver |  | home screen handler |
| 4 | lbStationName | Label |  | label station name |
| 5 | lbStationAddress | Label |  | label station address |
| 6 | lbStationNoBikes | Label |  | label station number of bike |
| 7 | lbStationDistance | Label |  | label station distance |
| 8 | btnViewStationDetail | Button |  | button view station detail |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | setupData | void | setup data |
| 2 | setupFunctionality | void | setup functionality |
| 3 | viewStationDetail | void | view station detail |
| 4 | getDockStation | DockStation | get dock station |

*Parameter:*

* screenPath
* dockStation

*Exception:*

* IOException

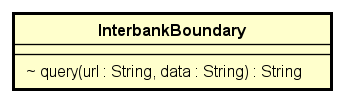
**Method:**

* NONE

**State:**

NONE

#### Class “InterbankBoundary”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | Query | String | send query |

*Parameter:*

* url
* data: data to send

*Exception:*

* UnrecognizedException

**Method:**

* NONE

**State:**

NONE

#### Class “InterbankConfigs”

Text

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | PROCESS\_TRANSACTION\_URL | String |  | transaction url |
| 2 | PUBLIC\_KEY | String |  | public key |
| 3 | SECRET\_KEY | String |  | secret key |
| 4 | PAY\_COMMAND | String | pay | command |
| 5 | REFUND\_COMMAND | String | refund | command |
| 6 | VERSION | String |  | version of interbank |

**Operation**

None

Parameter:

None

Exception:

- DataSourceException

**Method:**

None

**State:**

None

#### Class “InterbankDataConverter”

Table

Description automatically generated

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger |  | Used for logging |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | convertToPayload | String | Compile data into interbank standard payload |
| 2 | convertToResponse | PaymentTransaction | Make payment transaction |

*Parameter:*

* card: the credit card used for payment/refund
* amount: the amount to pay/refund
* contents: the transaction contents
* interbankCommand: command for interbank
* responseText: JSON formatted returned by Interbank server

*Exception:*

* PaymentException
* UnrecognizedException

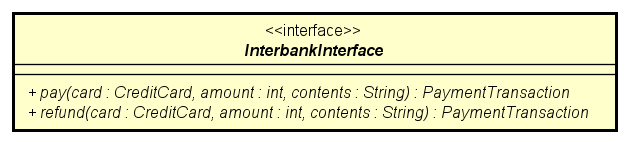
**Method:**

* convertErrorCodeToException: Convert Interbank errorCode to readable system-native Exception
* generateData: Convert a map to JSON-formatted String
* getToday: Return a string that represents the current time in the format of yyyy-MM-dd HH:mm:ss.
* md5: Return a string that represents the cipher text encrypted by md5 algorithm.

**State:**

NONE

#### Class “InterbankInterface”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 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
* UnrecognizedException

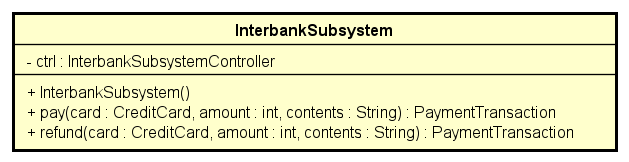
**Method:**

* NONE

**State:**

NONE

#### Class “InterbankSubsystem” (implements InterbankInterface)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | ctrl | InterbankSubsystemController | Initialized inside constructor | interbank subsystem controller |

**Operation**

Overrides interface operations.

*Parameter:*

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

*Exception:*

* PaymentException
* UnrecognizedException

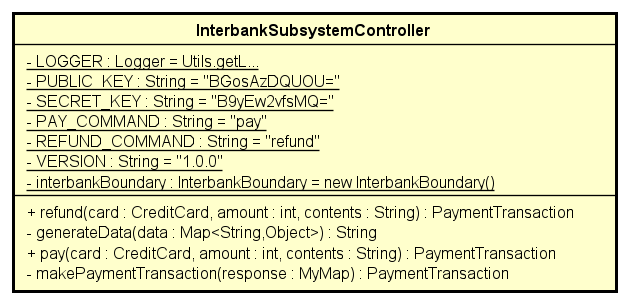
**Method:**

* NONE

**State:**

NONE

#### Class “InterbankSubsystemController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger |  | Used for logging |
| 2 | PUBLIC\_KEY | String |  | public key |
| 3 | SECRET\_KEY | String |  | secret key |
| 4 | PAY\_COMMAND | String |  | pay command |
| 5 | REFUND\_COMMAND | String |  | refund command |
| 6 | VERSION | String |  | version |
| 7 | interbankBoundary | InterbankBoundary |  | interbank boundary |

**Operation**

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

*Parameter:*

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

*Exception:*

* PaymentException
* UnrecognizedException

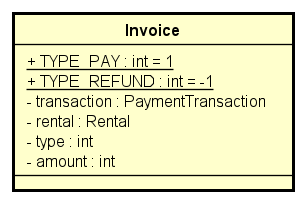
**Method:**

* generateData: generate data
* makePaymentTransaction: create payment transaction

**State:**

NONE

#### Class “Invoice”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | TYPE\_PAY | Int | 1 | type of pay |
| 2 | TYPE\_REFUND | Int | 1 | type of refund |
| 3 | transaction | PaymentTransaction | Null | payment transaction |
| 4 | rental | Rental | Null | rental information |
| 5 | type | Int | Null | type of invoice |
| 6 | amount | Int | Null | amount to pay/refund |

**Operation**

None

*Parameter:*

None

*Exception:*

* NONE

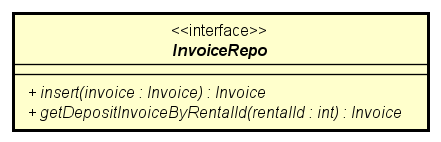
**Method:**

* NONE

**State:**

NONE

#### Class “InvoiceRepo”(extends BaseRepo)



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | insert | Invoice | insert invoice to database |
| 2 | getDepositInvoiceByRentalId | Invoice | get invoice by rental id |

*Parameter:*

* invoice
* rentalId: rental ID

*Exception:*

* DataSourceException

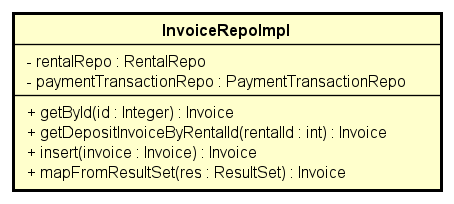
**Method:**

* NONE

**State:**

NONE

#### Class “InvoiceRepoImpl”(implements InvoiceRepo, ResultSetMappable)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | rentalRepo | RentalRepo | Initialized inside constructor | rental repository |
| 2 | paymentTransactionRepo | PaymentTransactionRepo | Initialized inside constructor | payment transaction repository |

**Operation**

Overrides interface operations.

*Parameter:*

* invoice
* rentalId: rental ID

*Exception:*

* NONE

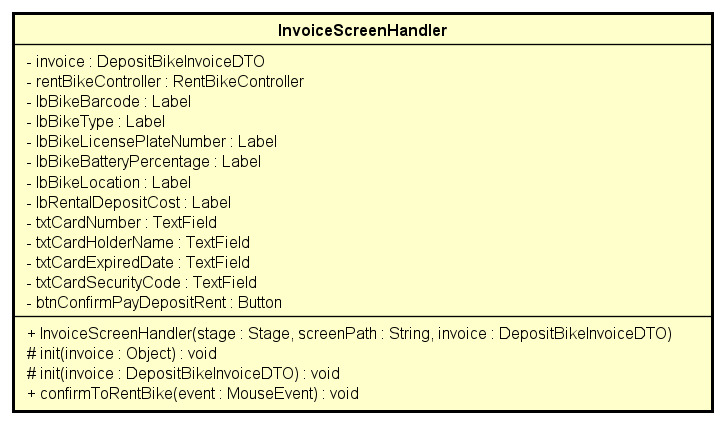
**Method:**

* NONE

**State:**

NONE

#### Class “InvoiceScreenHandler” (extends BaseScreenHandler)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | invoice | DepositBikeInvoiceDTO | Null | invoice DTO |
| 2 | rentBikeController | RentBikeController | Initialized inside constructor | rent bike controller |
| 3 | lbBikeBarcode | Label |  | label bike barcode |
| 4 | lbBikeType | Label |  | label bike type |
| 5 | lbBikeLicensePlateNumber | Label |  | label bike license plate |
| 6 | lbBikeBatteryPercentage | Label |  | label bike battery level |
| 7 | lbBikeLocation | Label |  | label bike location |
| 8 | lbRentalDepositCost | Label |  | label rental deposit cost |
| 9 | txtCardNumber | TextField |  | card number field |
| 10 | txtCardHolderName | TextField |  | card holder name field |
| 11 | txtCardExpirationDate | TextField |  | card expiration date field |
| 12 | txtCardSecurityCode | TextField |  | card security code field |
| 13 | btnConfirmPayDepositRent | Button |  | button confirm pay deposit rent |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | confirmToRentBike | void | confirm to rent bike |

*Parameter:*

* stage
* screenPath
* invoice

*Exception:*

* IOException

**Method:**

* NONE

**State:**

NONE

#### Class “IRentalCostStrategy”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | calculateRentalCost | Int | calculate rental cost |

*Parameter:*

* rental: rental information

*Exception:*

* NONE

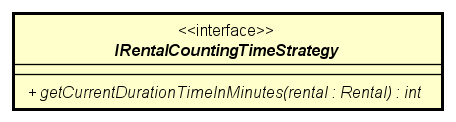
**Method:**

* NONE

**State:**

NONE

#### Class “IRentalCountingTimeStrategy”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | getCurrentDurationTimeInMinutes | Int | get duration time of current rental |

*Parameter:*

* rental: rental information

*Exception:*

* NONE

**Method:**

* NONE

**State:**

NONE

#### Class “IRentalDepositStrategy”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | calculateRentalDeposit | Int | calculate rental deposit amount |

*Parameter:*

* rental: rental information

*Exception:*

* NONE

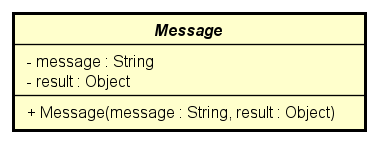
**Method:**

* NONE

**State:**

NONE

#### Class “Message”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | message | String | Null | message |
| 2 | result | Object | Null | result |

**Operation**

None

*Parameter:*

None

*Exception:*

* NONE

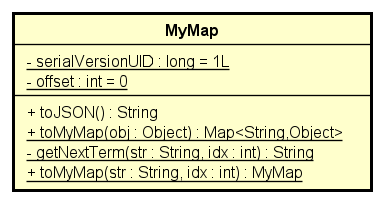
**Method:**

* NONE

**State:**

NONE

#### Class “MyMap” (extends LinkedHashMap)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data Type | Default Value | Description |
| 1 | serialVersionUID | Long | 1L |  |
| 2 | offset | Int | 0 |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | toJSON | String | return a string that represents JSON object |
| 2 | toMyMap | Map<String,Object> | return a map that represents the mapping among attributes name and their values of an object |
| 3 | toMyMap | MyMap | return a mymap that represents the interested substring in a string |

*Parameter:*

* str: string
* obj: Object

*Exception:*

* IllegalAccessException
* IllegalArgumentException

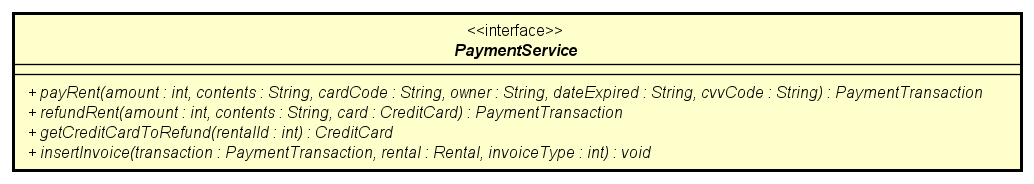
**Method:**

* getNextTerm: Return a string that represents the term in between 2 double quote

**State:**

NONE

#### Class “PaymentService”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return Type* | *Description* |
| 1 | payRent | PaymentTransaction | pay rent then return payment transaction |
| 2 | refund | PaymentTransaction | refund then return payment transaction |
| 3 | getCreditCardToRefund | CreditCard | get credit card to refund to |
| 4 | insertInvoice | void | insert invoice to database |

*Parameter:*

* amount: amount to pay/rent
* contents: content to pay/refund
* cardCode: card code
* owner: owner of card
* dateExpired: date expiration of card
* cvvCode: security code of card
* rentalId: rental ID
* transaction: payment transaction
* rental: rental information
* invoiceType: type of invoice

*Exception:*

* DataSourceException

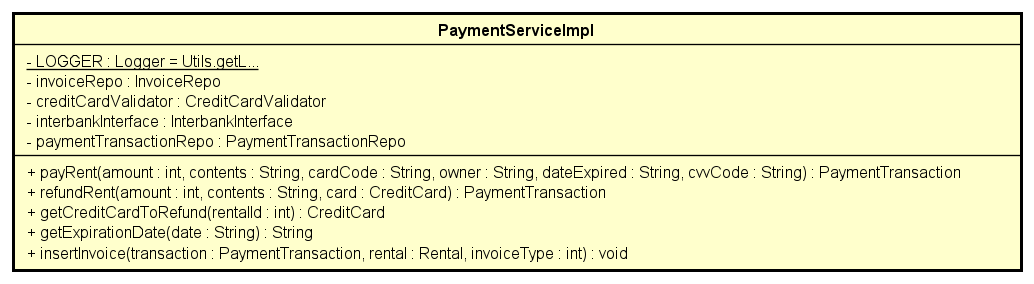
**Method:**

* NONE

**State:**

NONE

#### Class “PaymentServiceImpl” (implements PaymentService)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | invoiceRepo | InvoiceRepo | Initialized inside constructor | invoice repository |
| 3 | creditCardValidator | CreditCardValidator | Initialized inside constructor | validator of credit card |
| 4 | interbankInterface | InterbankInterface | Initialized inside constructor | interbank interface |
| 5 | paymentTransactionRepo | PaymentTransactionRepo | Initialized inside constructor | payment transaction repository |

**Operation**

Overrides interface operations.

*Parameter:*

* amount: amount to pay/rent
* contents: content to pay/refund
* cardCode: card code
* owner: owner of card
* dateExpired: date expiration of card
* cvvCode: security code of card
* rentalId: rental ID
* transaction: payment transaction
* rental: rental information
* invoiceType: type of invoice

*Exception:*

* PaymentException
* UnrecognizedException
* DataSourceException
* InvalidCardException

**Method:**

* getExpirationDate

**State:**

NONE

#### Class “PaymentTransaction”



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | transactionId | String | Null | Transaction’s id |
| 2 | card | CreditCard | Null | Info of credit card |
| 3 | errorCode | String | Null | Error code of transaction |
| 4 | contents | String | Null | Contents of transaction |
| 5 | amount | Int | Null | Amount of transaction |
| 6 | createdAt | String | Null | Time transaction is created |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

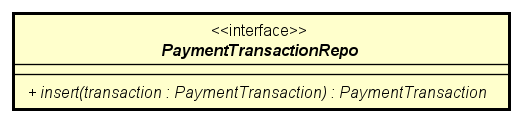
**Method**

* None

**State**

* None

#### Class “PaymentTransactionRepo”



**Attribute**

* None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | Insert | PaymentTransaction | Insert payment transaction to database |

*Parameter:*

* transaction: the transaction is being inserted

*Exception:*

* None

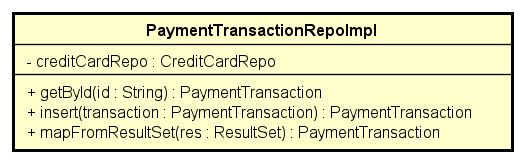
**Method**

* None

**State**

* None

#### Class “PaymentTransactionRepoImpl” (implements PaymentTransactionRepo, ResultSetMappable<PaymentTransaction>)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | creditCardRepo | CreditCardRepo | Initialized inside constructor | Credit card’s repository |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getById | PaymentTransaction | Get the payment transaction by id |
| 2 | mapFromResultSet | PaymentTransaction | Convert result set to payment transation |

*Parameter:*

* id: id of payment transaction
* transaction: inserted transaction
* res: result set from database

*Exception:*

* SqlException: if responded with no connection to database
* DatasourceException: if responded with no database

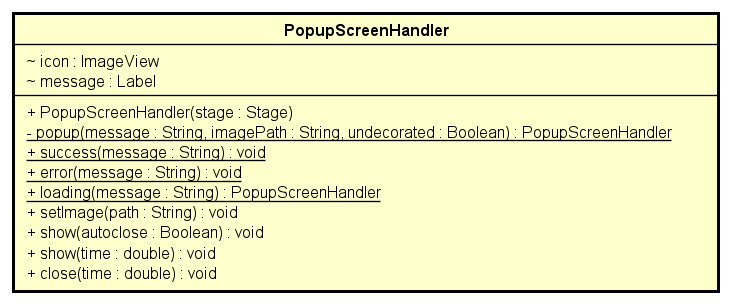
**Method**

* None

**State**

* None

#### Class “PopupScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | Icon | ImageView | Null | Popup’s icon |
| 2 | Message | Label | Null | Message of pop up |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | success | Void | Return success message |
| 2 | error | Void | Return error message |
| 3 | loading | PopupScreenHandler | Return loading icon |
| 4 | setImage | Void | Set image for icon |
| 5 | show | Void | Show pop up |
| 6 | close | Void | Close pop up |

*Parameter:*

* message: contents of the message of pop up
* path: path to display icon
* time: time to do the action

*Exception:*

* IOException: if input or output is wrong format

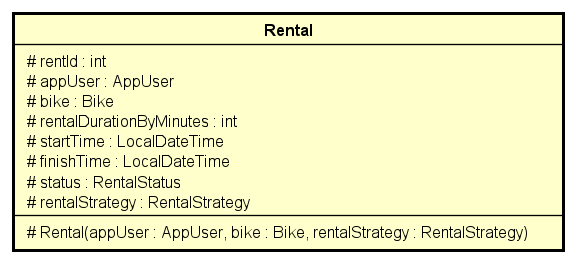
**Method**

* None

**State**

* None

#### Class “Rental”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | rentId | Int | Null | Rental’s id |
| 2 | appUser | AppUser | Null | App user |
| 3 | bike | Bike | Null | Renting bike |
| 4 | rentalDurationByMinutes | Int | Null | Rental duration |
| 5 | startTime | LocalDateTime | Null | Time start renting |
| 6 | finishTime | LocalDateTime | Null | Time finish renting |
| 7 | Status | RentalStatus | Null | Status of rental |
| 8 | rentalStrategy | RentalStrategy | Null | Rental strategy |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

**Method**

* None

**State**

* None

#### Class “RentalCostStrategyFactory”

Table

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | create | IRentalCostStrategy | picking the correct cost strategy for Rental |

Parameter:

None

Exception:

None

**Method:**

None

**State:**

None

#### Class “RentalCountingTimeStrategyFactory”

A picture containing text

Description automatically generated

**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | create | IRentalCostStrategy | picking the correct counting time strategy for Rental |

Parameter:

None

Exception:

None

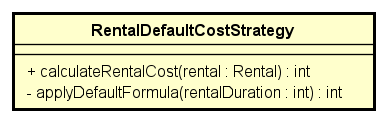
**Method:**

None

**State:**

None

#### Class “RentalDefaultCostStrategy”



**Attribute**

* None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | applyDefaultFormula | Int | Rental cost |

*Parameter:*

* rentalDuration: duration of renting

*Exception:*

* None

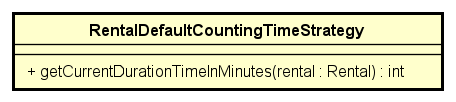
**Method**

* None

**State**

* None

#### Class “RentalDefaultCountingTimeStrategy”



**Attribute**

* None

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

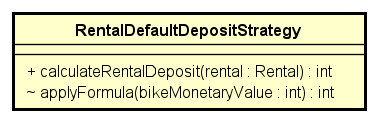
**Method**

* None

**State**

* None

#### Class “RentalDefaultDepositStrategy”



**Attribute**

* None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | applyFormula | Int | Rental cost |

*Parameter:*

* bikeMonetaryValue: monetary of bike

*Exception:*

* None

**Method**

* None

**State**

* None

#### Class “RentalDepositStrategyFactory”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return Type | Description |
| 1 | create | IRentalCostStrategy | picking the correct deposit strategy for Rental |

Parameter:

None

Exception:

None

**Method:**

None

**State:**

None

#### Class “RentalDetailDTO”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | depositAmount | Int | Null | Deposit amount |
| 2 | costAmount | Int | Null | Cost amount |
| 3 | Rental | Rental | Null | Rental |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

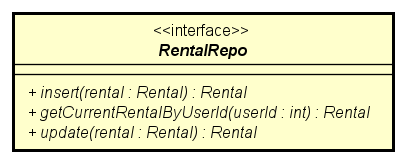
**Method**

* None

**State**

* None

#### Class “RentalRepo”



**Attribute**

* None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | insert | PaymentTransaction | Insert rental to database |
| 2 | getCurrentRentalByUserId | Rental | Get current rental by id |
| 3 | update | Rental | Update rental to database |

*Parameter:*

* rental: Rental
* userId: user’s id

*Exception:*

* DatasourceException: if responded with no database

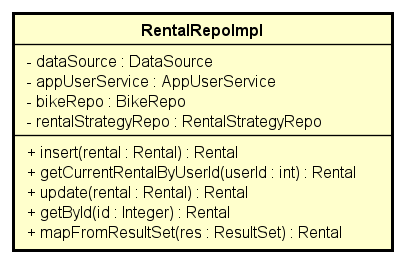
**Method**

* None

**State**

* None

#### Class “RentalRepoImpl” (implements RentalRepo, ResultSetMappable<Rental>)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | dataSource | String | Injected from IoC | Data source |
| 2 | appUserService | AppUserService | Injected from IoC | App user service |
| 3 | bikeRepo | BikeRepo | Injected from IoC | Bike repository |
| 4 | rentalStrategyRepo | RentalStrategyRepo | Injected from IoC | Rental strategy repository |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

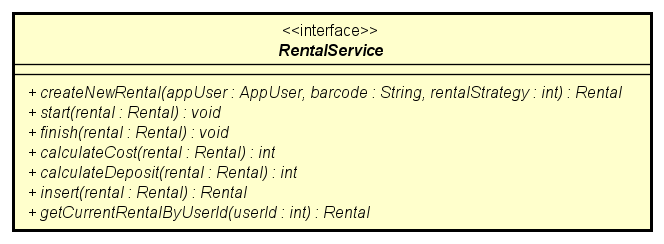
**Method**

* None

**State**

* None

#### Class “RentalService”



**Attribute**

* None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | start | Void | Start renting time |
| 2 | finish | Void | Finish renting time |
| 3 | calculateCost | Void | Calculate cost of renting |
| 4 | calculateDeposit | Void | Calculate deposit of renting |
| 5 | insert | Rental | Insert rental to database |
| 6 | getCurrentRentalByUserId | Rental | Get current rental by user id |

*Parameter:*

* rental: Rental
* userId: user’s id

*Exception:*

* DatasourceException: if responded with no database
* UnavailableBikeException: if responded with no unavailable bike

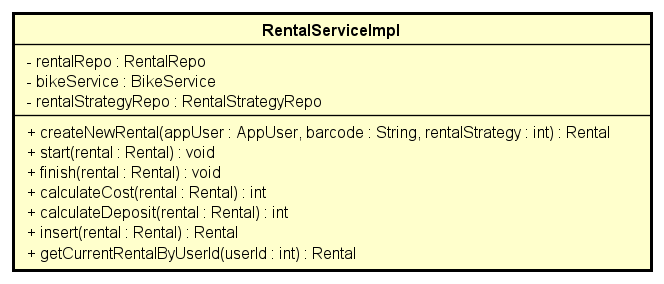
**Method**

* None

**State**

* None

#### Class “RentalServiceImpl” (implements RentalService)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | rentalRepo | RentalRepo | Initialized inside constructor | Rental repository |
| 2 | bikeService | BikeService | Initialized inside constructor | Bike service |
| 3 | rentalStrategyRepo | RentalStrategyRepo | Initialized inside constructor | Rental strategy repository |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

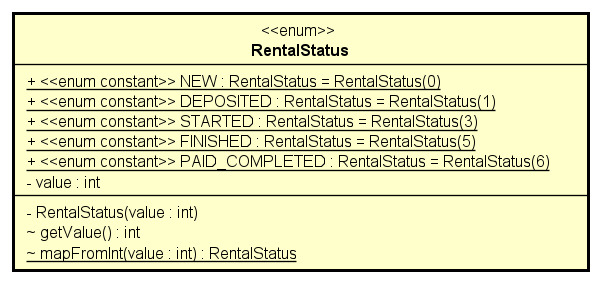
**Method**

* None

**State**

* None

#### Class “RentalStatus”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | value | int | null | value of rental |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getValue | int | Get value of rental |
| 2 | mapFromInt | RentalStatus | Map from value to rental status |

*Parameter:*

* value: value of rental

*Exception:*

* None

**Method**

* None

**State**

* None

#### Class “RentalStrategy”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | id | Int | Null | Rental strategy’s id |
| 2 | name | String | Null | Name of rental strategy |
| 3 | description | String | Null | Description |
| 4 | costStrategy | IRentalCostStrategy | Null | Cost Strategy |
| 5 | depositStrategy | IRentalDepositStrategy | Null | Deposit Strategy |
| 6 | countingTimeStrategy | IRentalCountingTimeStrategy | Null | Counting Time Strategy |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

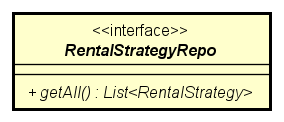
**Method**

* None

**State**

* None

#### Class “RentalStrategyRepo”



**Attribute**

* None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getAll | List<RentalStrategy> | Get all rental strategy |

*Parameter:*

* None

*Exception:*

* DatasourceException: if responded with no database

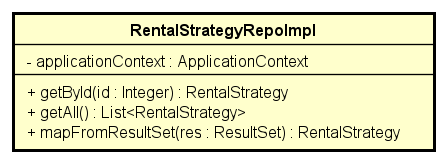
**Method**

* None

**State**

* None

#### Class “RentalStrategyRepoImpl” (implements RentalStrategyRepo, ResultSetMappable<RentalStrategy>)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | applicationContext | ApplicationContext | Injected from IoC | Application context |

**Operation**

* None

*Parameter:*

* None

*Exception:*

* None

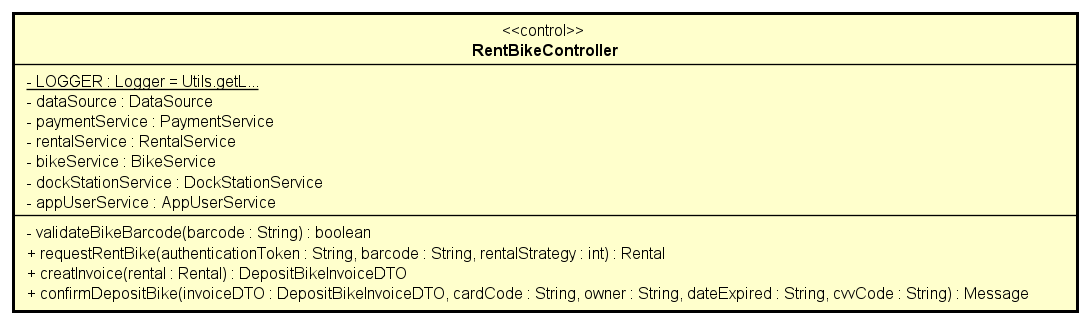
**Method**

* None

**State**

* None

#### Class “RentBikeController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | null | Use for logging |
| 2 | dataSource | DataSource | Initialized inside constructor | Data source |
| 3 | paymentService | PaymentService | Initialized inside constructor | Payment service |
| 4 | rentalService | RentalService | Initialized inside constructor | Rental service |
| 5 | bikeService | BikeService | Initialized inside constructor | Bike service |
| 6 | dockStationService | DockStationService | Initialized inside constructor | Dock station service |
| 7 | appUserService | AppUserService | Initialized inside constructor | App user service |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | requestRentBike | Rental | request to rent bike |
| 2 | createInvoice | DepositBikeInvoiceDTO | create invoice for bike |
| 3 | confirmToDeposit | Message | confirm to deposit for bike |

*Parameter:*

* rental: rental;
* invoiceDTO: invoice DTO
* authenticationToken: authentication token
* barcode: barcode of bike
* rentalStrategy: rental strategy
* cardCode: card code
* owner: owner of card
* dateExpired: date expiration of card
* cvvCode: security code of card

*Exception:*

* SqlException: if responded with no connection to database
* DatasourceException: if responded with no database

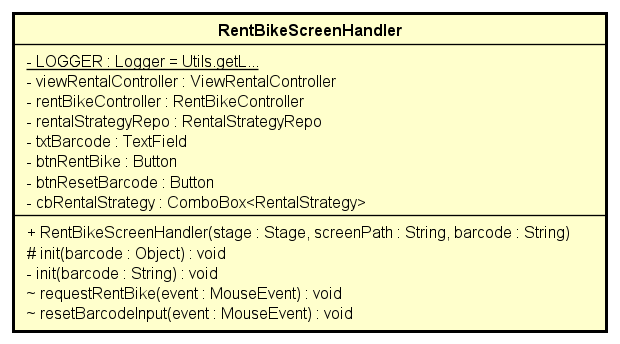
**Method**

* validateBikeBarcode: validate the barcode of bike

**State**

None

#### Class “RentBikeScreenHandler” (extends BaseScreenHandler)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | viewRentalController | ViewRentalController | Initialized inside constructor | view rental controller |
| 3 | rentBikeController | RentBikeController | Initialized inside constructor | rent bike controller |
| 4 | txtBarcode | TextField | null | text of barcode |
| 5 | btnRentBike | Button | null | button rent bike |
| 6 | btnResetBarcode | Button | null | button reset barcode |
| 7 | cbRentalStrategy | ComboBox<RentalStrategy> | null | combobox of rental strategy |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | init | void | Initialize for rent bike screen |
| 2 | requestToRentBike | void | request to rent bike |
| 3 | resetBarcodeInput | void | reset barcode of bike |

*Parameter:*

* barcode: barcode of bike
* event: mouse event

*Exception:*

* DatasourceException: if responded with no database
* IOException: if input output in wrong format

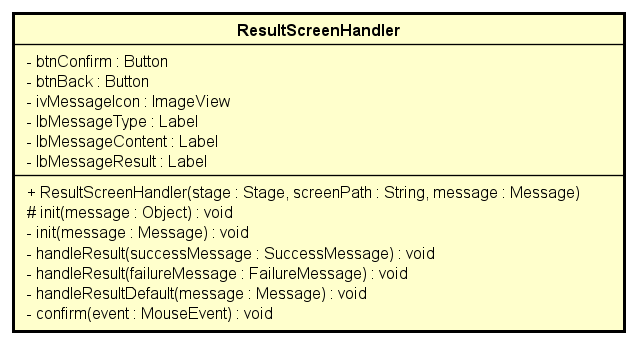
**Method**

None

**State**

None

#### Class “ResultScreenHandler” (extends BaseScreenHandler)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | btnConfirm | Button | null | button confirm result |
| 2 | btnBack | Button | null | button back to previous screen |
| 3 | ivMessageIcon | ImageView | null | image of message |
| 4 | lbMessageType | Label | null | message type |
| 5 | lbMessageContent | Label | null | message content |
| 6 | lbMessageResult | Label | null | message result |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | init | void | Initialize for result screen |
| 2 | handleResult | void | Choose which kind of result to display |
| 3 | confirm | void | Confirm the result |

*Parameter:*

* message: message
* failureMessage: failure message contents
* successMessage: success message contents
* event: mouse event

*Exception:*

* IOException: if input output in wrong format

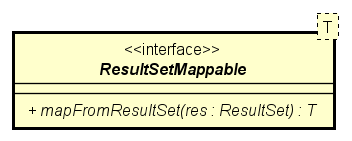
**Method**

None

**State**

None

#### Class “ResultSetMappable”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | mapFromResultSet | T | Convert from result set to object T |

*Parameter:*

* res: result set

*Exception:*

* DatasourceException: if responded with no database
* SQLException: if responded with no connection to database

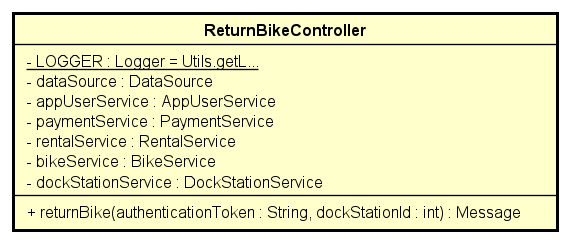
**Method**

None

**State**

None

#### Class “ReturnBikeController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | null | Use for logging |
| 2 | dataSource | DataSource | Initialized inside constructor | Data source |
| 3 | paymentService | PaymentService | Initialized inside constructor | Payment service |
| 4 | rentalService | RentalService | Initialized inside constructor | Rental service |
| 5 | bikeService | BikeService | Initialized inside constructor | Bike service |
| 6 | dockStationService | DockStationService | Initialized inside constructor | Dock station service |
| 7 | appUserService | AppUserService | Initialized inside constructor | App user service |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | returnBike | Message | return bike to dock station and pay the cost |

*Parameter:*

* authenticationToken: authentication token

*Exception:*

* SqlException: if responded with no connection to database
* DatasourceException: if responded with no database

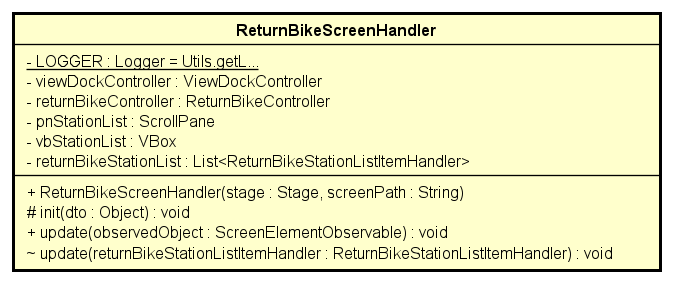
**Method**

None

**State**

None

#### Class “ReturnBikeScreenHandler”(extends BaseScreenHandler implements ScreenElementObserver)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | viewDockController | ViewDockController | Initialized inside constructor | view dock controller |
| 3 | returnBikeController | ReturnBikeController | Initialized inside constructor | return bike controller |
| 4 | pnStationList | ScrollPane | null | list of station |
| 5 | vbStationList | vbox | null | list of station |
| 6 | returnBikeStationList | List<ReturnBikeStationListItemHandler | null | return bike station list |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | update | void | Update return bike station list |

*Parameter:*

* returnBikeStationListItemHandler: return bike station list item handler

*Exception:*

* DatasourceException: if responded with no database
* IOException: if input output in wrong format

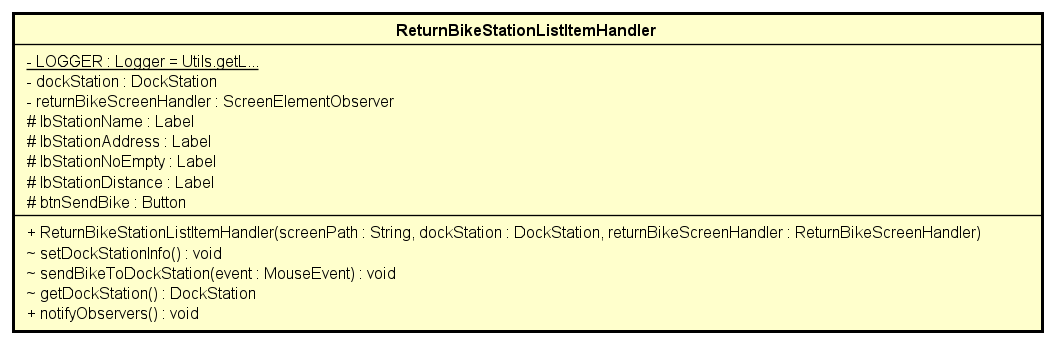
**Method**

None

**State**

None

#### Class “ReturnBikeStationListItemHandler” (extends FXMLScreenHandler implements ScreenElementObserver)



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | returnBikeScreenHandler | ReturnBikeScreenHandler | Initialized inside constructor | return bike screen handler |
| 3 | lbStationName | Label | null | Station name |
| 4 | lbStationAddress | Label | null | Station address |
| 5 | lbStaionNoEmpty | Label | null | Number of empty station |
| 6 | lbStationDistance | Label | null | Station distance |
| 7 | btnSendBike | Button | null | send bike button |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | setDockInfo | void | Set up the dock information |
| 2 | sendBikeToDockStation | void | Send bike to dock station |
| 3 | getDockStation | DockStation | Get dock station |

*Parameter:*

* returnBikeStationListItemHandler: return bike station list item handler

*Exception:*

* IOException: if input output in wrong format

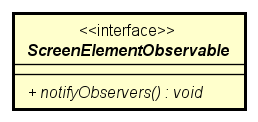
**Method**

None

**State**

None

#### Class “ScreenElementObservable”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | notifyObservers | void | Convert from result set to object T |

*Parameter:*

None

*Exception:*

* Exception

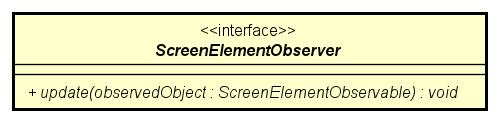
**Method**

None

**State**

None

#### Class “ScreenElementObserver”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | update | void | update when observed object change |

*Parameter:*

None

*Exception:*

* Exception

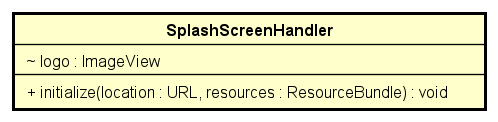
**Method**

None

**State**

None

#### Class “SplashScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | logo | ImageView | null | Logo of application |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | initialize | void | Initialize for splash screen |

*Parameter:*

* location: path to logo
* resources: resource bundle

*Exception:*

None

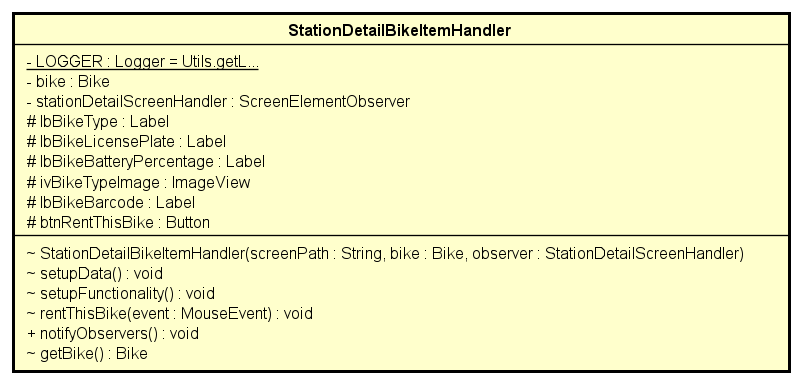
**Method**

None

**State**

None

#### Class “StationDetailBikeItemHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | stationDetailScreenHandler | ScreenElementObserver | Initialized inside constructor | station detail screen handler |
| 3 | lbBikeType | Label | null | bike type |
| 4 | lbBikeLicensePlate | Label | null | license plate |
| 5 | lbBikeBatteryPercentage | Label | null | bike battery percentage |
| 6 | ivBikeTypeImage | ImageView | null | bike type image |
| 7 | ibBikeBarcode | Label | null | bike barcode |
| 8 | btnRentThisBike | Button | null | button rent this bike |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | setupDate | void | set up data |
| 2 | setupFunctionality | void | set up functionality |
| 3 | rentThisBike | void | rent bike |
| 4 | getBike | Bike | get bike |

*Parameter:*

* event: mouse event

*Exception:*

* IOException: if input output in wrong format

**Method**

None

**State**

None

#### Class “StationDetailScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | viewDockController | ViewDockController | Initialized inside constructor | view dock controller |
| 3 | pnBikeList | ScrollPane | null | bike list scroll pane |
| 4 | vbBikeList | VBox | null | bike list vbox |
| 5 | lbDockName | Label | null | dock name |
| 6 | lbDockEmpty | Label | null | dock empty |
| 7 | lbDockDistance | Label | null | dock distance |
| 8 | lbDockWalkingTimeByMinutes | Label | null | dock walking time by minutes |
| 9 | bikeTypeList | VBox | null | bike type list |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | init | void | Initialize for station detail screen |
| 2 | update | void | Update station detail list item |

*Parameter:*

* dockStationId: dock station’s id
* stationDetailBikeItemHandler: station detail bike item handler

*Exception:*

* IOException: if input output in wrong format

**Method**

None

**State**

None

#### Class “SuccessMessage”



**Attribute**

None

**Operation**

None

*Parameter:*

None

*Exception:*

None

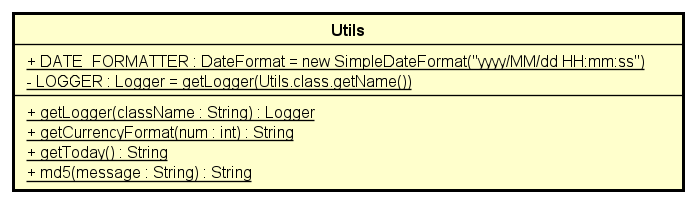
**Method**

None

**State**

None

#### Class “Utils”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getLogger | Logger | get logger |
| 2 | getCurrencyFormat | String | get currency format |
| 3 | getToday | String | get today |
| 4 | md5 | String | get hash code md5 |

*Parameter:*

* className: name of class
* num: money
* message: message to hash

*Exception:*

None

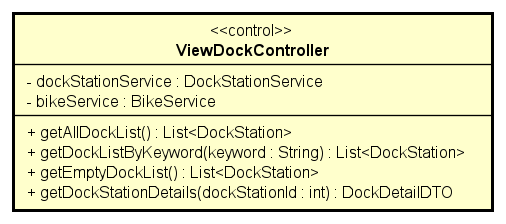
**Method**

None

**State**

None

#### Class “ViewDockController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | dockStationService | DockStationService | Initialized inside constructor | dock station service |
| 2 | bikeService | BikeService | Initialized inside constructor | bike service |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getAllDockList | List<DockStation> | get all dock station to list |
| 2 | getDockListByKeyword | List<DockStation> | get all dock station by keyword |
| 3 | getEmptyDockList | List<DockStation> | get empty dock to list |
| 4 | getDockStationDetails | DockDetatilDTO | get dock station details |

*Parameter:*

* keyword: keyword
* dockStationId: dock station’s id

*Exception:*

* DatasourceException: if responded with no database

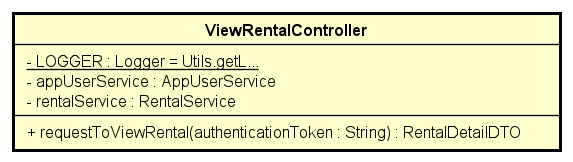
**Method**

None

**State**

None

#### Class “ViewRentalController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | rentalService | RentalService | Initialized inside constructor | Rental service |
| 3 | appUserService | AppUserService | Initialized inside constructor | App user service |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | requestToViewRental | RentalDetailDTO | request to view rental |

*Parameter:*

* authenticationToken: authentication token

*Exception:*

* NotFoundUserRentalException: if responded with no user’s rental
* InternalServerException: if responded with no server

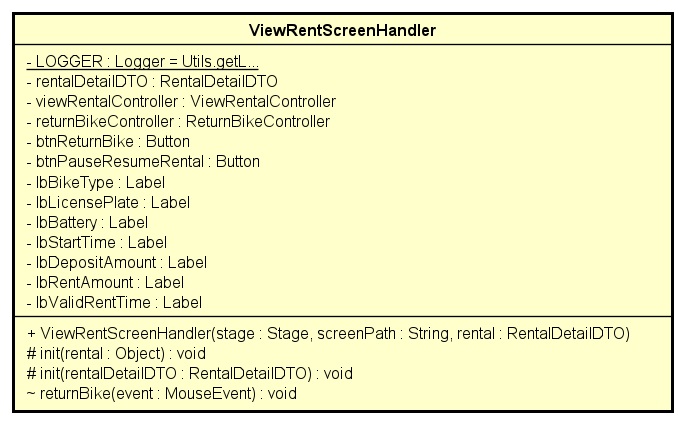
**Method**

None

**State**

None

#### Class “ViewRentScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | LOGGER | Logger | Initialized inside constructor | Use for logging |
| 2 | rentalDetailDTO | RentalDetailDTO | Initialized inside constructor | rental detail dto |
| 3 | viewRentalController | ViewRentalController | Initialized inside constructor | view rental controller |
| 4 | returnBikeController | ReturnBikeController | Initialized inside constructor | return bike controller |
| 5 | btnReturnBike | Button | null | return bike button |
| 6 | btnPauseResumeRental | Button | null | pause resume rental button |
| 7 | lbBikeType | Label | null | bike type |
| 8 | lbBikeLicensePlate | Label | null | license plate |
| 9 | lbBattery | Label | null | bike battery percentage |
| 10 | lbStartTime | Label | null | start time |
| 11 | lbDepositAmount | Label | null | deposit amount |
| 12 | lbRentAmount | Label | null | rent amount |
| 13 | lbValidRentTime | Label | null | valid rent time |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | init | void | Initialize for view rent screen |
| 2 | returnBike | void | Return bike and calculate cost |

*Parameter:*

* rentalDetailDTO: rental detail dto
* event: mouse event

*Exception:*

* IOException: if input output in wrong format

**Method**

None

**State**

None

# Design Considerations

## Goals and Guidelines

### Goals

The system was designed with simplicity and consistency, through transactions, in mind. It would mean less buttons, less cluttering, the UI should feel fresh and easy to navigate, the displayed texts mostly in bare form. It should not, however, be bareboned and emotionless. Here and there the user can spot a touch of bright colors, as if to invoke liveliness. The primary color, in these parts, is green, embodying the spirit of Eco Park, as well as representing cleanness and vitality. Preview images are hand-picked, with an upward view into the cloudless sky to further cement this ideology.

### Guidelines

*Clean code is put in the highest priority.*

* *All the variables are named in order to be fully readable and self-documented.*
* *Classes and interfaces symbol are constructed in the standard to be easily recognized and understood.*
* *Each logical unit is split into methods, ensuring the unit test is utilized and functionally cohesive.*

## Architectural Strategies

The highest cares when designing the project structure is to make role of each component/module as clear as possible, to enhance the code speed and to identify the responsibility of each member during the development of the software. To archive those, each module should contain a solid amount of logic and be independent with each other. Therefore, both horizontal and vertical slices are applied on the project architecture. Horizontal slices, or layering, help improve the abstraction, while vertical slices enhance the agility in coding and make it easier to divide task to members. Thus, we chose multi-tier architecture to implement our project.

To save time of code, we reuse code from the case study project for the views layer and use several libraries. Lombok library is widely used in our projects, removing tons of boiling code related to basic functionalities such as getters, setters, default constructors.

## Coupling and Cohesion

*Coupling and Cohesion is carefully taken during programming phase. Here we examine multiple key classes:*

### Coupling

|  |  |  |  |
| --- | --- | --- | --- |
| **Class name** | **Coupling type** | **Description** | **Improvement** |
| *com.isd.ict.capstoneproject.*  **Group3Application** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.controller*  **RentBikeController** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.controller*  **ReturnBikeController** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.controller*  **ViewDockController** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.controller*  **ViewRentalController** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.bike*  **BikeServiceImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.bike*  **BikeRepoImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.station*  **DockStationServiceImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.station*  **DockStationRepoImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.rental*  **RentalServiceImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.rental*  **RentalRepoImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.payment*  **PaymentServiceImpl** | Stamp Coupling | Modules communicate by entity but only using the id of the object. | For simplitcity, no changes are made |
| *com.isd.ict.capstoneproject.rental*  **RentalStrategyRepoImpl** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.rental*  **InterbankSubsystemController** | Data coupling | Modules communicate by passing only data. | None |
| *Com.isd.ict.capstoneproject.subsystem.interbank*  **InterbankBoundary** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.rental*  **RentalDefaultCostStrategy**  **RentalDefaultCountingStrategy**  **RentalDefaultCostStrategy** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.views.handler*  **HomeScreenHandler** | Stamp Coupling | Modules communicate by entity but only using the id of the object. | For simplitcity, no changes are made |
| *com.isd.ict.capstoneproject.views.handler*  **DockDetailedScreenHandler** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.views.handler*  **InvoiceScreenHandler** | Data coupling | Modules communicate by passing only data. | None |
| *com.isd.ict.capstoneproject.rental*  **RentalStrategyFactory** | Control coupling | Type of strategy is passed to pick the certain strategy | None |

### Cohesion

|  |  |  |  |
| --- | --- | --- | --- |
| **Class name** | **Coupling type** | **Description** | **Improvement** |
| *com.isd.ict.capstoneproject.*  **Group3Application** | Sequential cohesion | All element follow the template of parent class | None |
| *com.isd.ict.capstoneproject.controller*  **RentBikeController** | Sequential cohesion | Most output operations are used for input. | None |
| *com.isd.ict.capstoneproject.controller*  **ReturnBikeController** | Sequential cohesion | Most output operations are used for input. | None |
| *com.isd.ict.capstoneproject.controller*  **ViewDockController** | Temporal cohesion | All element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.controller*  **ViewRentalController** | Functional cohesion | Fully cohesive due to taking only one reponsibility | None |
| *com.isd.ict.capstoneproject.bike*  **BikeServiceImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.bike*  **BikeRepoImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.station*  **DockStationServiceImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.station*  **DockStationRepoImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.rental*  **RentalServiceImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.rental*  **RentalRepoImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.payment*  **PaymentServiceImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | For simplitcity, no changes are made |
| *com.isd.ict.capstoneproject.rental*  **RentalStrategyRepoImpl** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.rental*  **InterbankSubsystemController** | Sequential cohesion  Temporal cohesion | Most output operations are used for input. Besides, some element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.rental*  **InterbankBoundary** | Functional cohesion | Fully cohesive due to taking only one reponsibility | None |
| *com.isd.ict.capstoneproject.rental*  **RentalDefaultCostStrategy**  **RentalDefaultCountingStrategy**  **RentalDefaultCostStrategy** | Functional cohesion | Fully cohesive due to taking only one reponsibility | None |
| *com.isd.ict.capstoneproject.views.handler*  **HomeScreenHandler** | Temporal cohesion | All element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.views.handler*  **DockDetailedScreenHandler** | Temporal cohesion | All element is called following the timing of the main application flow | None |
| *com.isd.ict.capstoneproject.views.handler*  **InvoiceScreenHandler** | Temporal cohesion | All element is called following the timing of the main application flow | None |
| *All the entity classes, including Bike, DockStation, PaymentTransaction, Rental, CreditCard, Invoice, User* | Fuctional cohesion | Fully cohesive due to being pure entity class | None |

## Design Principles

Most classes in the projects have followed the SOLID principles because couple design patterns are applied along with horizontal/vertical slices and support libraries. This unburdens a huge load of refactoring and modifications when it comes to extend the project business scope.

Here are the proofs:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | New Requirement | Violated Principle | Detailed | Improvement |
| 1 | Add a new kinda bike | None | The current design fits this case well | It’s just trivial to add new type of bike because BikeType is already an entity. To adapt to this change, we add one new record to table BikeType in database |
| 2 | Changing the rental pricing | None | The current design fits this case well. | To apply this new requirement, simply add a new class, named Rental24hCostStrategy, implement IRentalCostStrategy to execute this new requirement. Then add a new record to table RentalStrategy with value in costStrategy column as rental24hCostStrategy (followed Spring convention) |
| 3 | Add new features: Allow customer to pause the time counter of rental | Principle O (open-closed) will be violated | Rental need more statuses to track. RentalService need more methods to allow controller layer to invoke the features. Counting Time needs extra class to implement the algorithms. Also, the existed RentalStrategy need updates for this new features. | Couple changes introduced:   * A new class, named RentalPausableCountingTimeStrategy, impelement IRentalCountingStrategy to execute the logic of this feature. * To support the algorithms, extra enum value is added to Rental Status to track the state of the rental * RentalService will need to add 2 more functionality to pause and resume the rental * Update all existed record in RentalStrategy table |

## Design Patterns

*A considerable amount of design patterns is used throughout the project structures, along with the n-tier architecture, they improve abstraction, flexibility and testability.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Pattern Name | Advantages | Use case | Implementation |
| 1 | Singleton | Save time and memory when instantiating a lot of non-entity object | Controller layer and DataSource component | A private static instance is stored inside the Singleton class  Constructors are kept privately also, the global instance are return by a static getInstance method. |
| 2 | Factory | Give a flexible mechanism to create/recreate suitable strategy for rental | For picking the right strategy for Rental both when create new Rental and reload Rental from Database. | The factory class has a create function, taking a string parameter to pick the right strategy. |
| 3 | Strategy | + Flexible when choosing methodology to calculate Rental value  + Extensible for upcoming requirements. | Rental Busines Logic: Counting Time, Calculate Deposit and Cost | Rental Strategy object as an attribute of Rental, here contains all related strategies.  Each strategy is abstract by an interface.  Each implementation is then save to database to later invoke, remove the control coupling |
| 4 | Facade | +  Help solidify the logic of each entity into their own packages, strengthen encapsulation for each packages.  +  Make controller layer thinner and more focused on orchestration | Service interfaces of each entity modules | Make a delegate interface of each modules, list out all the logic functionalities related to the entity into that interface, called Service interface.  The implementation of this interface will deploy all the logical codes, also communicate to persistence layer. |
| 5 | Repository | + Remove the logic of storing data off the business logic.  + This patterns also provide the reusibility. | Repo interfaces of each entity. | Make a interface of each entity, list out all activity of CRUD of the corresponding entity. |
| 6 | Data Transfer Object | + Wrapper  to transfer  information between  layer, provide encapsulation and prevent stamp coupling | Communication between Presentation layer and Controller layer | Create typical wrapper class for each typical method |
| 7 | Observer | + Solve the problem in communication between several screen and its contained children. | Screens which contains a list of items. Such as Home Screen, Dock Details Screen and Dock To Return Bike Screen | 2 interface are made, ScreenElementObserver and ScreenElementObservable, implemented by corresponding element handler class. |
| 8 | Template Method | + Remove duplicated code and increase cohesion | Constructors of Screen Handler classes have to take the data then allocate the functionality attached to each element and display the data on the right location. They are followed the same order of tasks. | Compose an abstract method in the base screen handler class that defined all the step and error handling, with serveral supplementary step methods   Override it later in each sub-class |