HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

EcoBikeRental

Subject: ITSS Software Development

Group 8:

Trần Thị Hằng - 20176748

Dương Thị Huê - 20176772

Đỗ Minh Thông - 20176881

Phạm Nhật Linh – 20184285

Hanoi, November 2020

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

1 Introduction

1.1 Objective

This SDD is written for the purpose of giving the audience a clear view about the design of the software. The document's intended audience is stakeholder and software designer/developer.

1.2 Scope

Product name: **EcoBikeRental Software (Eco-Bike-Rental is how we read)**

Explain:

> The software is for users to rent and return bikes automatically. EcoBikeRental is a 24/7 platform-independent system which allows novice users to user without any training. User must have an account to use the system. The software allows user to enter barcode or directly choose bike to rent and choose any bike station to return bike, use credit card for payment, and show detailed information of station and bike.

Application:

> Nowadays, the need for using bike is higher than ever. Using bike is not only environmentally friendly, but also a very effective way of exercising. The main drawback of this need is that not everyone has a bike, or has any intention of buying one. How about renting public bike for a relative cheap price? Introduce to our software. EcoBikeRental provides a quick and convenience way to rent bike. It helps to reduce employees, saves money and time, and very convenience. It is very easy to use. With a lot of stations and bikes available, it satisfies the need of bike rental service especially in Eco Park Township.

1.3 Glossary

We assume that the reader of this document has relatively good base knowledge about computer/software in general. Still, the document will be written in general-audience-friendly way that most reader can understand. Scholarly terms, if any, in this document will be briefly explained after it has been used.

1.4 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

2 Overall Description

2.1 General Overview

About the system, we have some characteristics that make the apps resemble e-commercial website/software: An interface for interacting with user; user request by clicking on the interface and then the request is processed by system controller; we have a database (remote) to store any kind of data; any data-related request or change will be queried in the database; the change in database is then reflected in the UI (user interface). As you can see, there are three main components in the system: the UI, controller and data model. We choose this software to be a desktop application. We choose the three-layer architecture to be our design approach. The design architecture helps to separate different components and better organize the codebase.

Here is general use-case diagram to help you understand the core of our design:

Fig 1. General Use Case Diagram

2.2 Assumptions/Constraints/Risks

2.2.1 Assumptions

User that use the software should have a good connection to the Internet. Also, our software is a desktop application, so the user also must have a laptop/desktop with an OS (we recommended 64 Bit Microsoft Windows 8 or later; macOS 10.13 or later; or any Linux distribution that supports running application) to run the apps. About the system requirement, we would say 2 GB RAM minimum, 8 GB RAM recommended; for storage 2.5 GB and another 1 GB for caches minimum, solid-state drive with at least 5 GB of free space recommended; require latest version of JRE; 1024×768 minimum screen resolution, 1920×1080 is a recommended screen resolution.

2.2.2 Constraints

- Less than 2GB RAM; JRE version<8; Low storage may cause the software to run incorrectly, or cannot start running at all.
- Implicitly stated, ideally, the response time for any tasks, with a moderate load, within the system is 1 second. But in case of peak load, a response time in the interval of 2 seconds is admissible.
- Weak or no internet connection may cause the software to run improperly.

2.2.3 Risks

- The database may have fault data if the user closes the application when not returning bike.

3 System Architecture and Architecture Design

Architecture Design steps:

1. Find out software components -> use cases
2. Find out Interaction between use cases
3. Find out Relationship between use cases
4. Draw UML Diagram includes: interaction diagram and analysis class diagram

3.1 *Architectural Patterns*

In our project, we use 3-tier architectural pattern. There are many benefits of separating an application into tiers and the most important thing is it allows us to update a specific part of an application independently of the other parts

3.2 Interaction Diagrams

3.2.1. Communication Diagrams

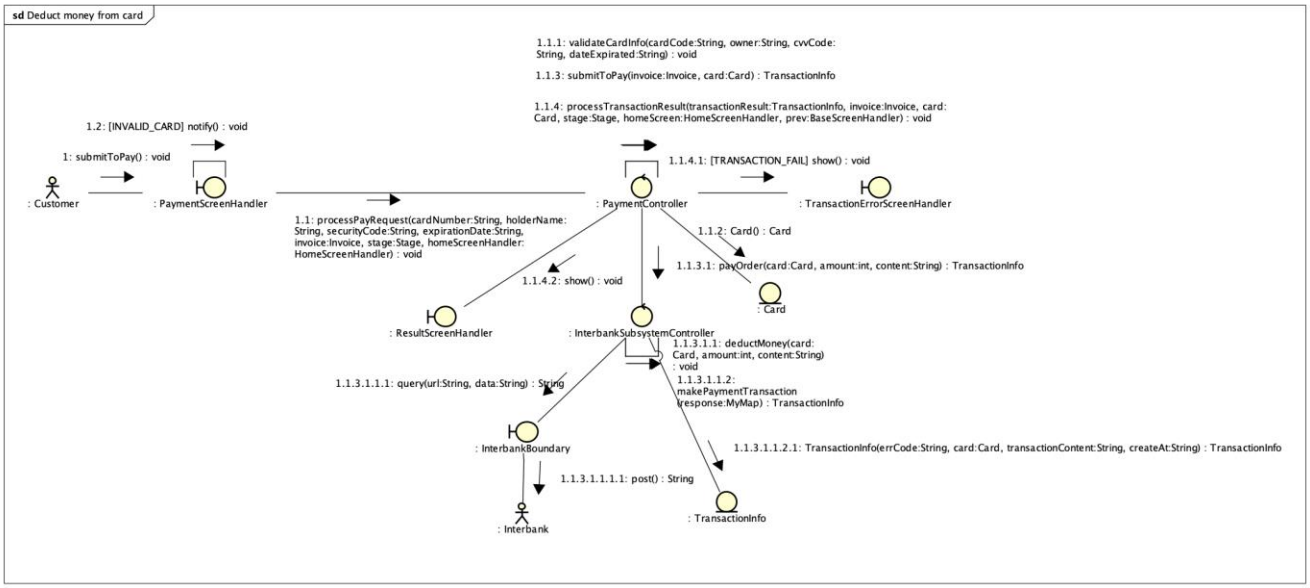


Fig 2. Communication Diagram for Deduct money from card

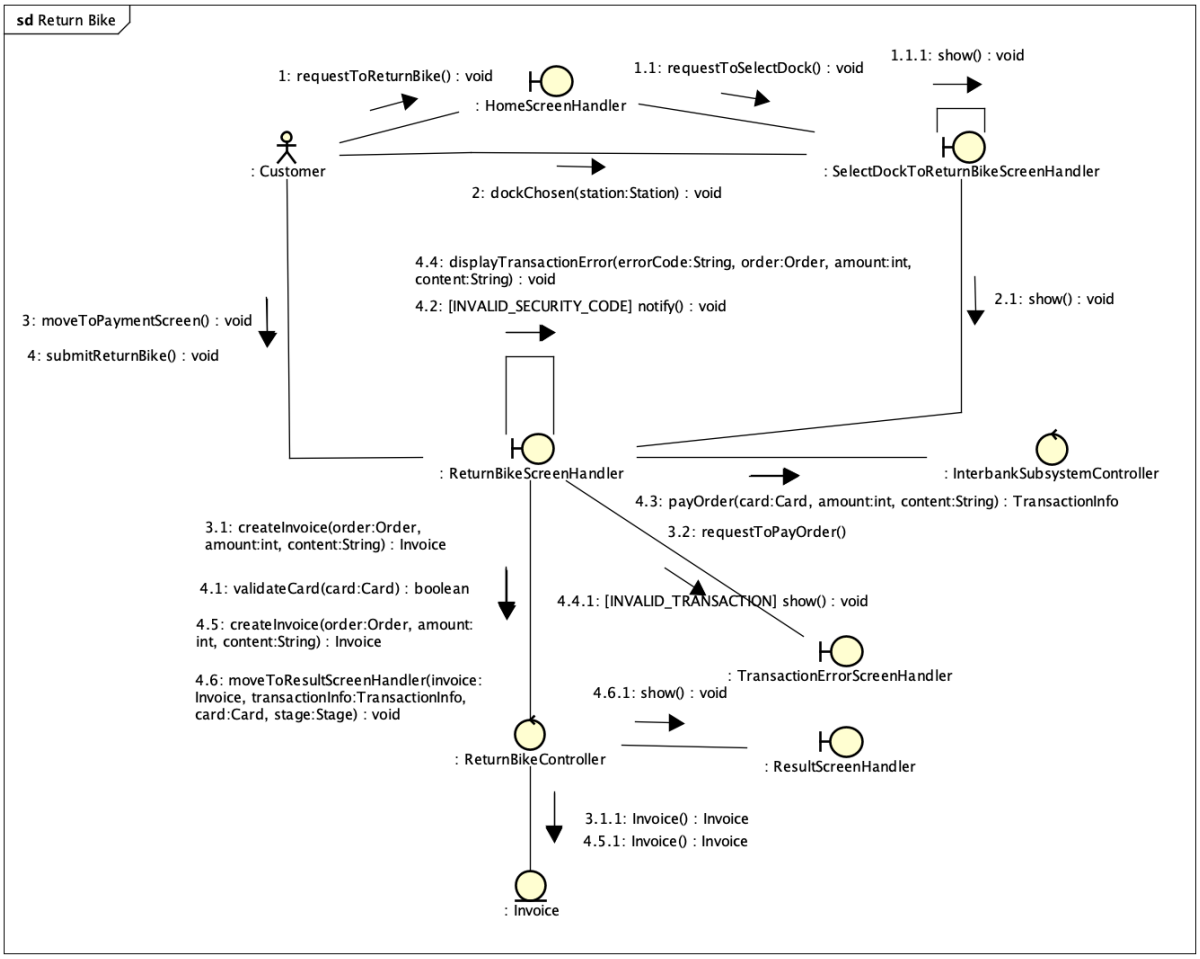


Fig 3. Communication Diagram for Return Bike

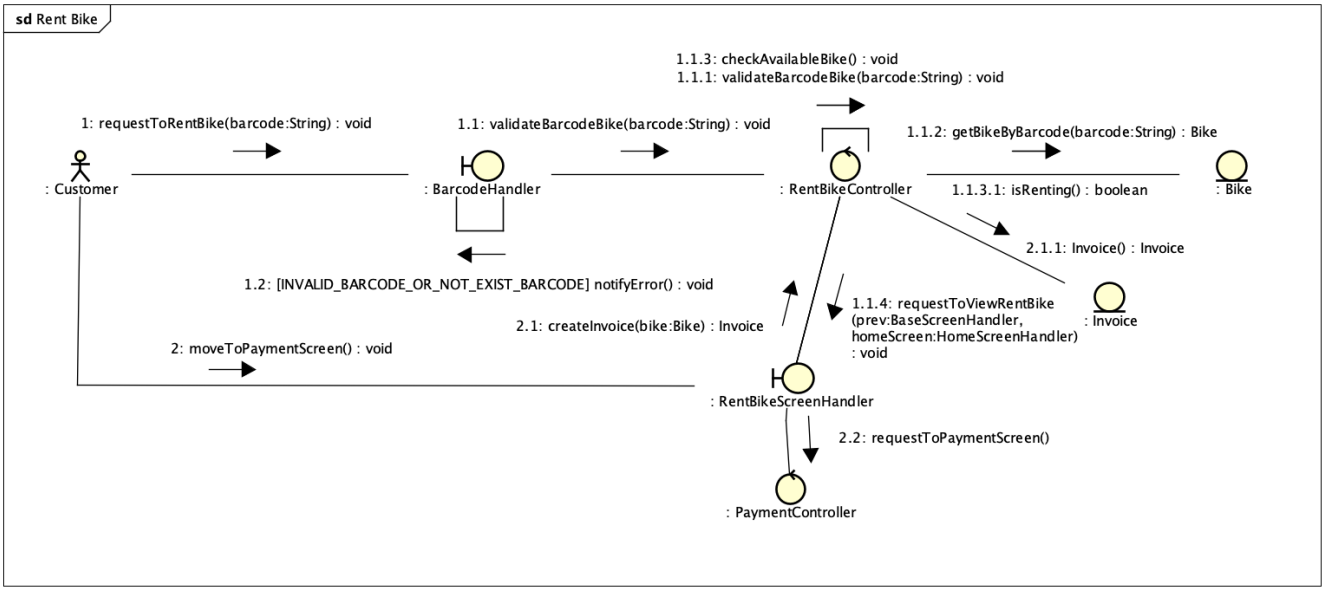


Fig 4. Communication Diagram for Rent Bike

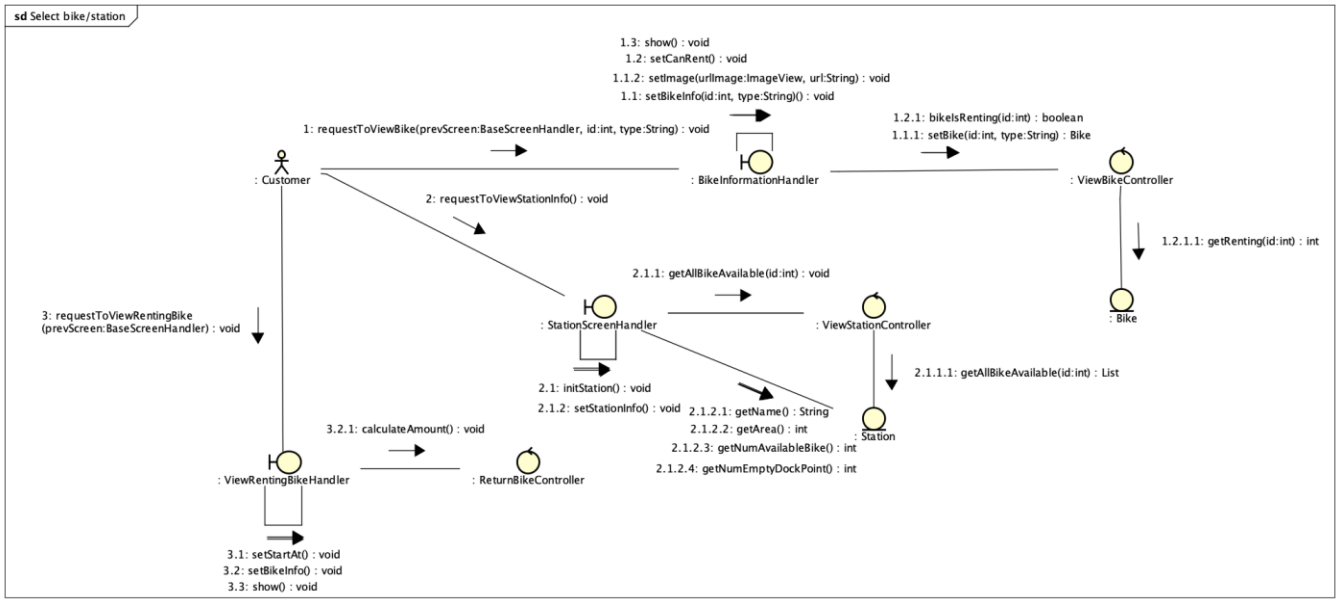


Fig 5. Communication Diagram for View Bike or Station information

3.2.2. Sequence Diagrams

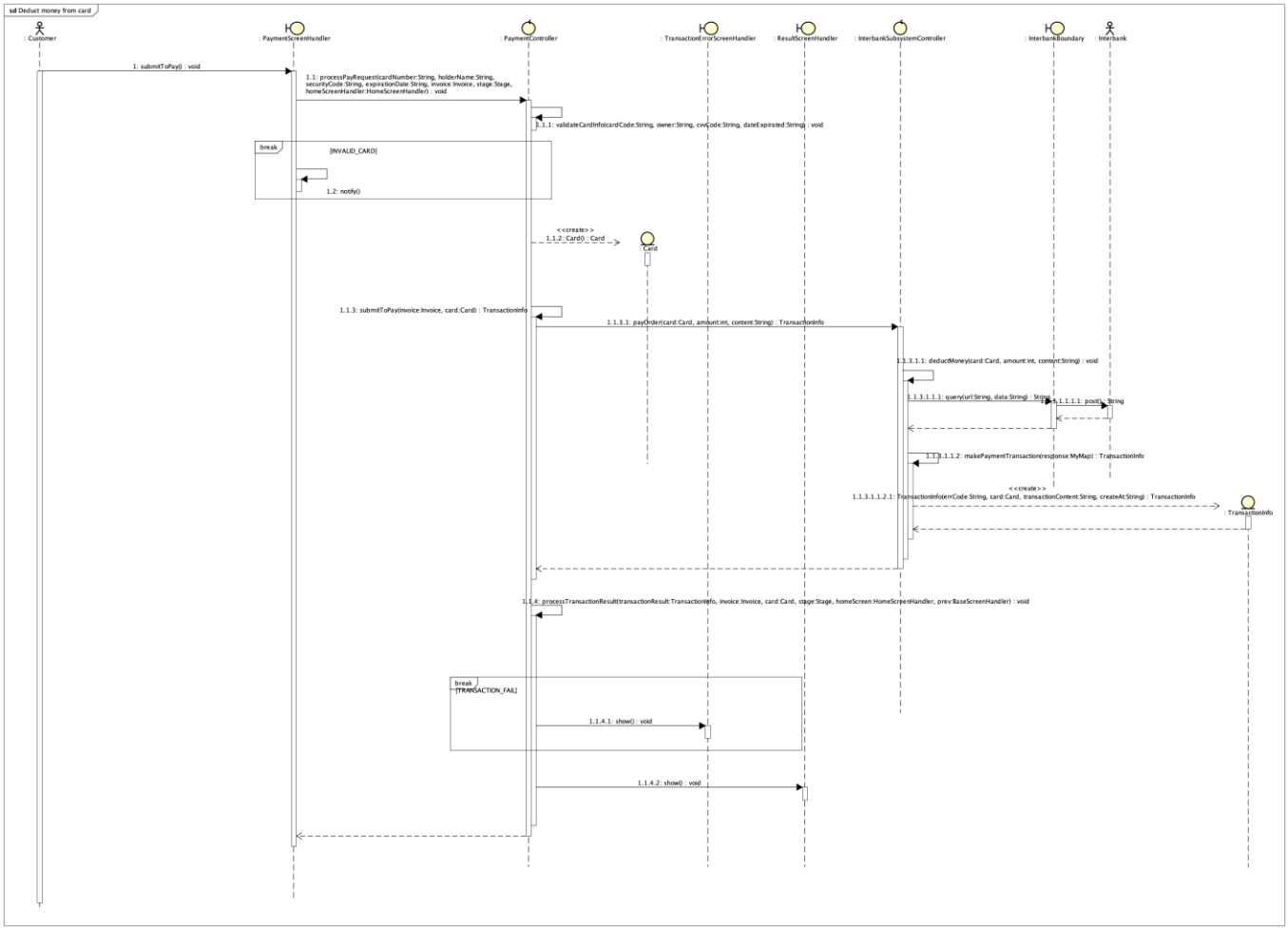


Fig 6. Sequence Diagram for Deduct money from card

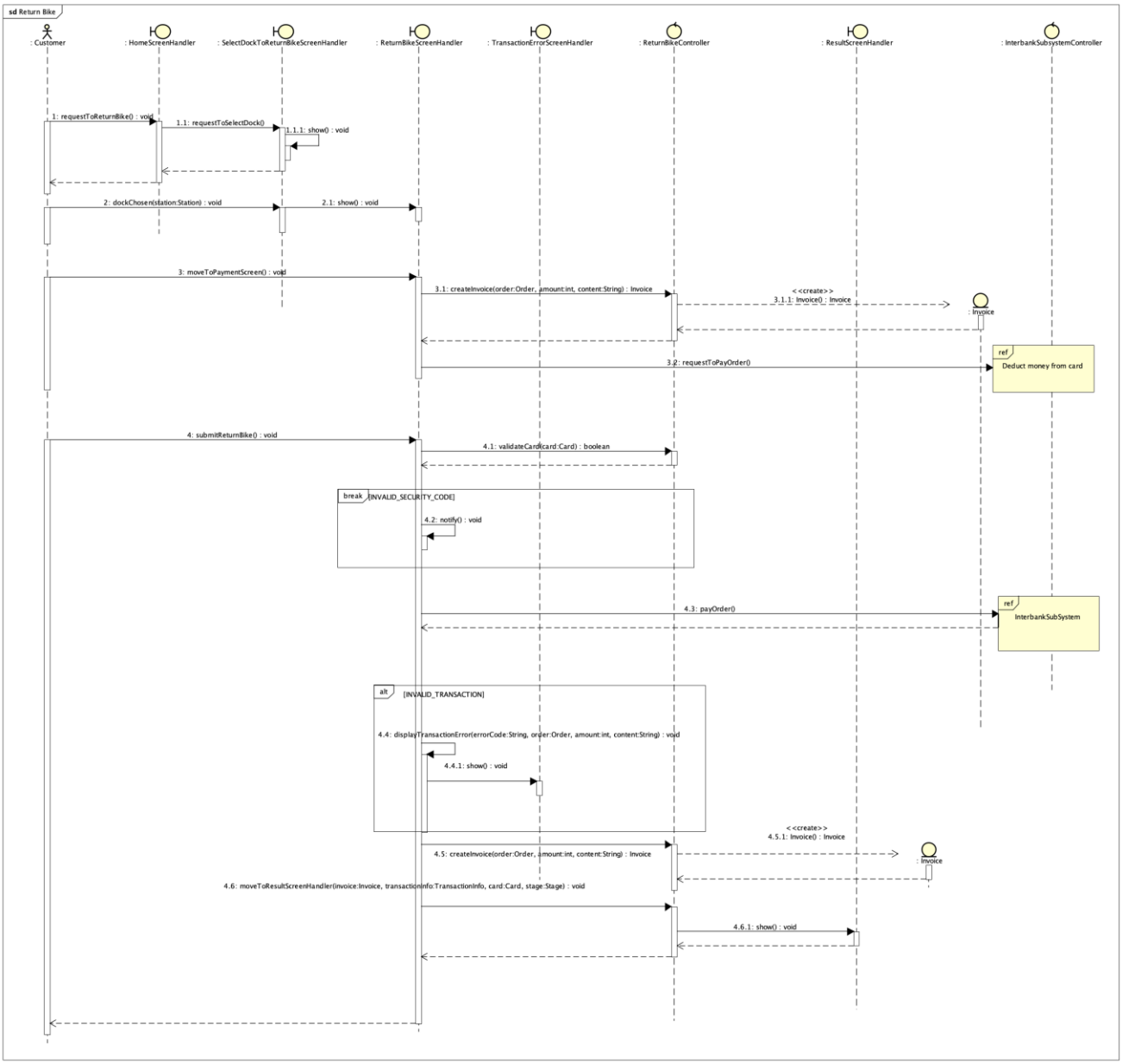


Fig 7. Sequence Diagram for Return bike

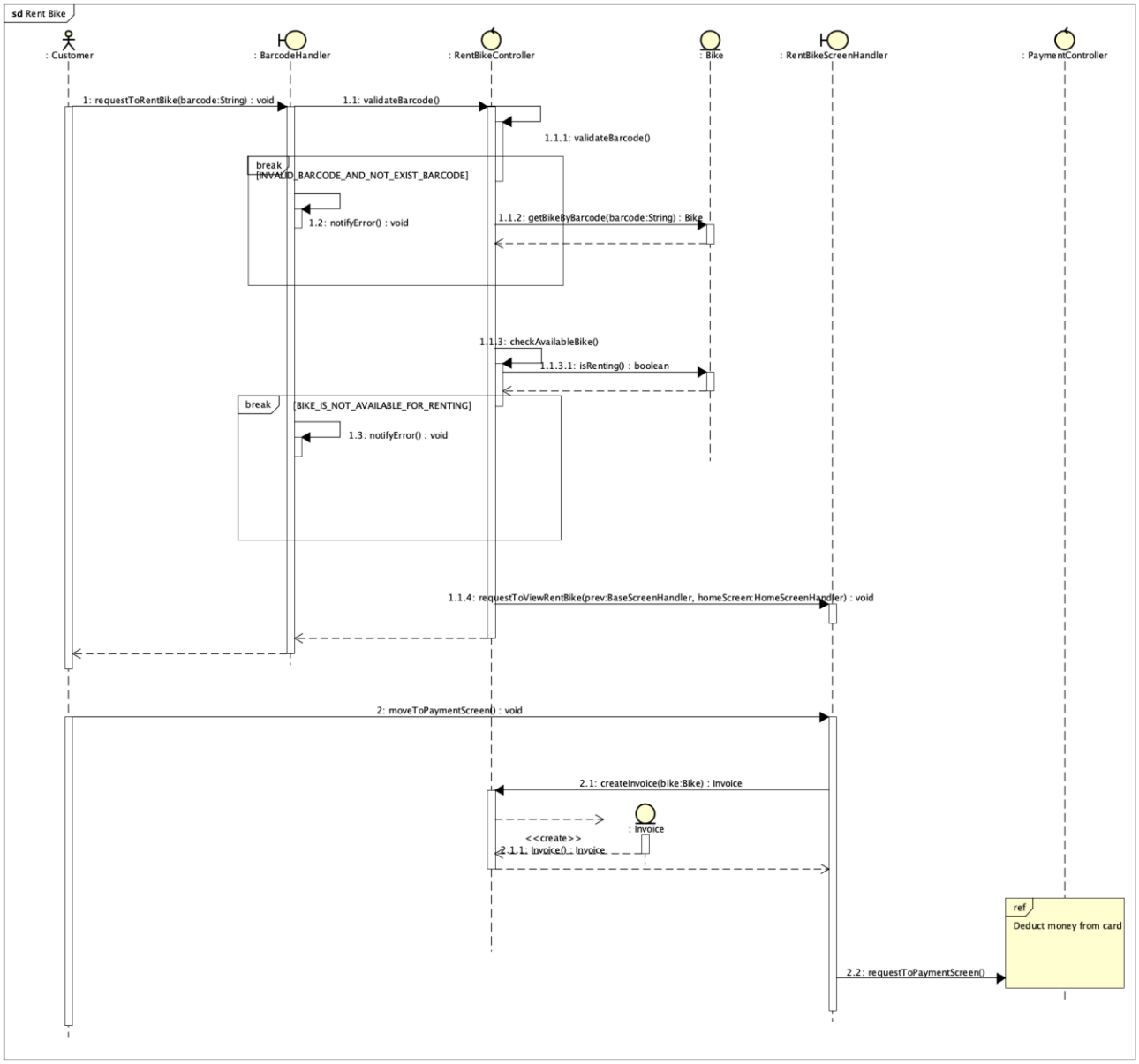


Fig 8. Sequence Diagram for Rent bike

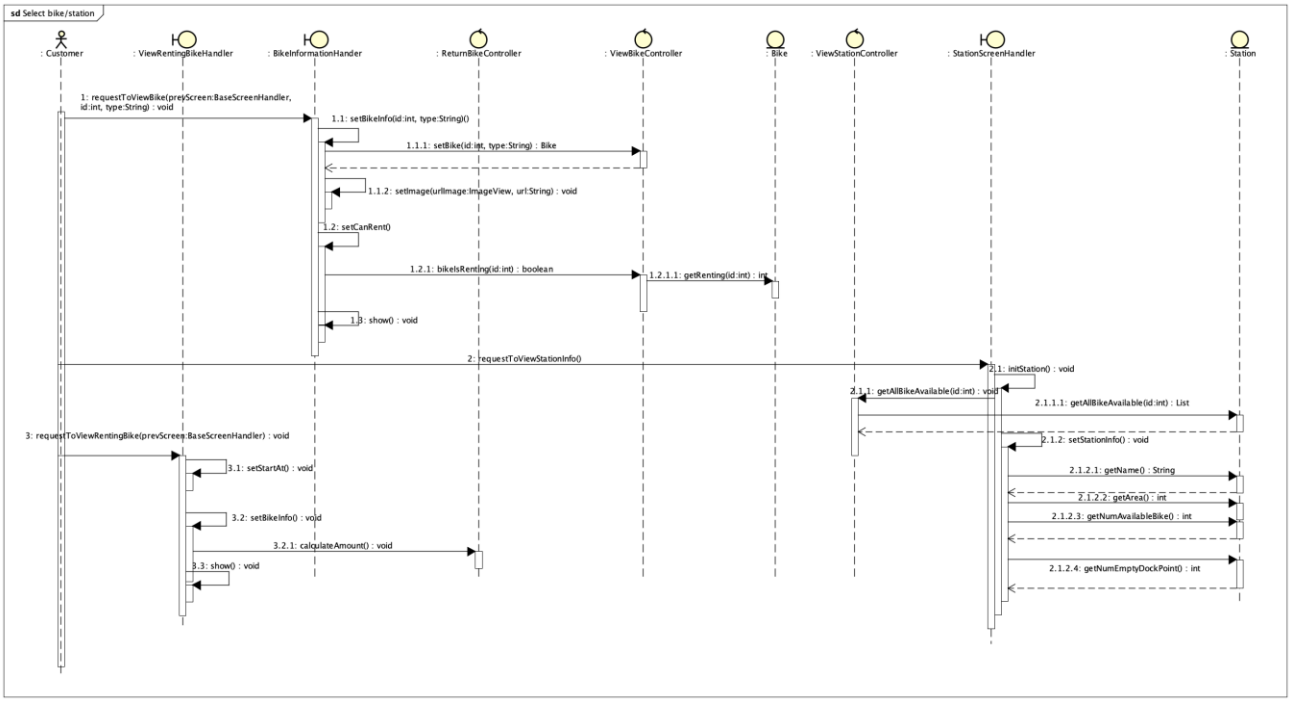


Fig 9. Sequence Diagram for View Bike or Station information

3.3 Analysis Class Diagrams

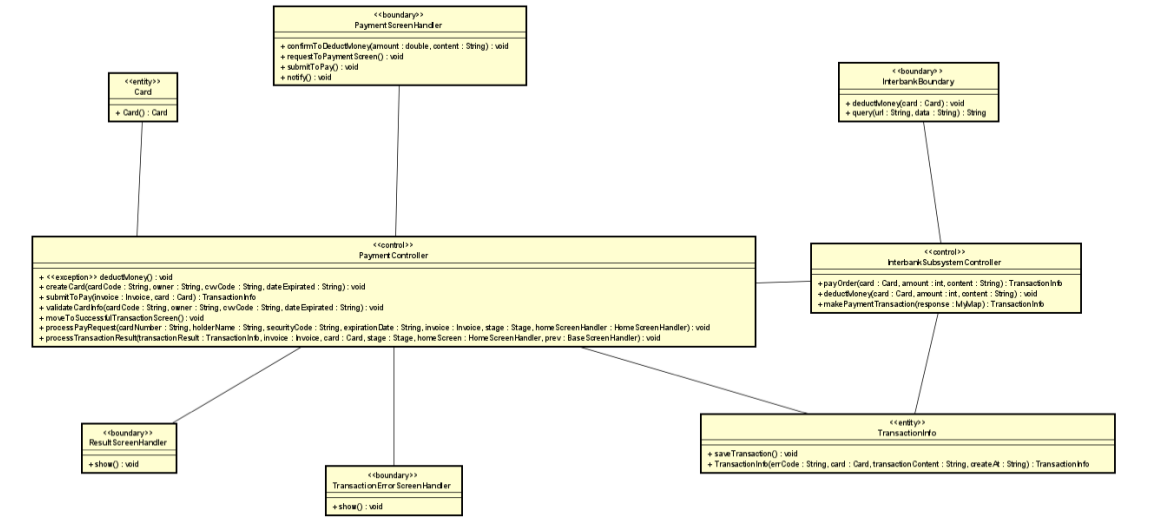


Fig 10. Analysis Class Diagram for Deduct money from card

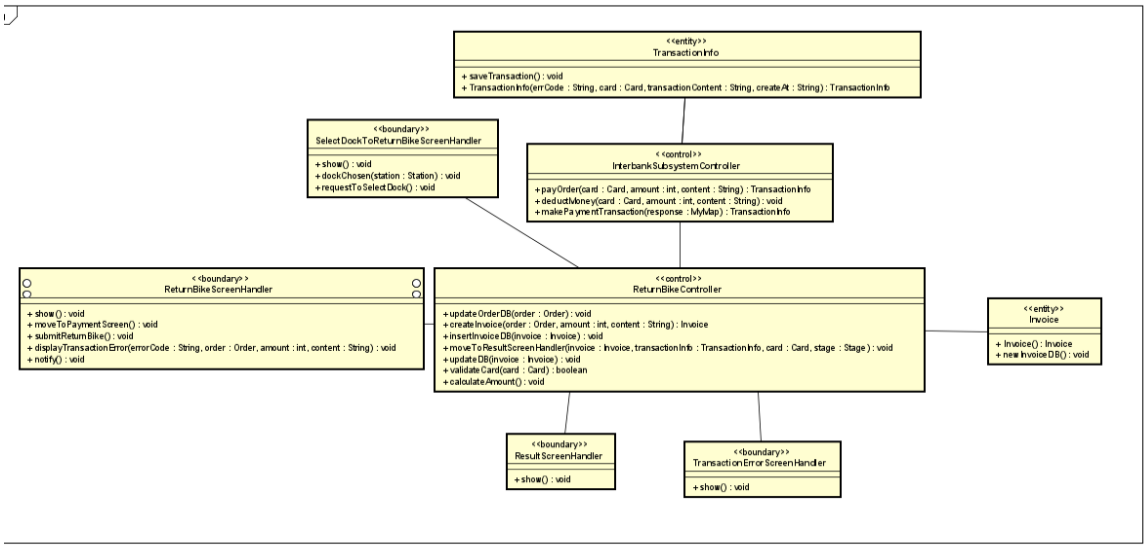


Fig 11. Analysis Class Diagram for Return bike

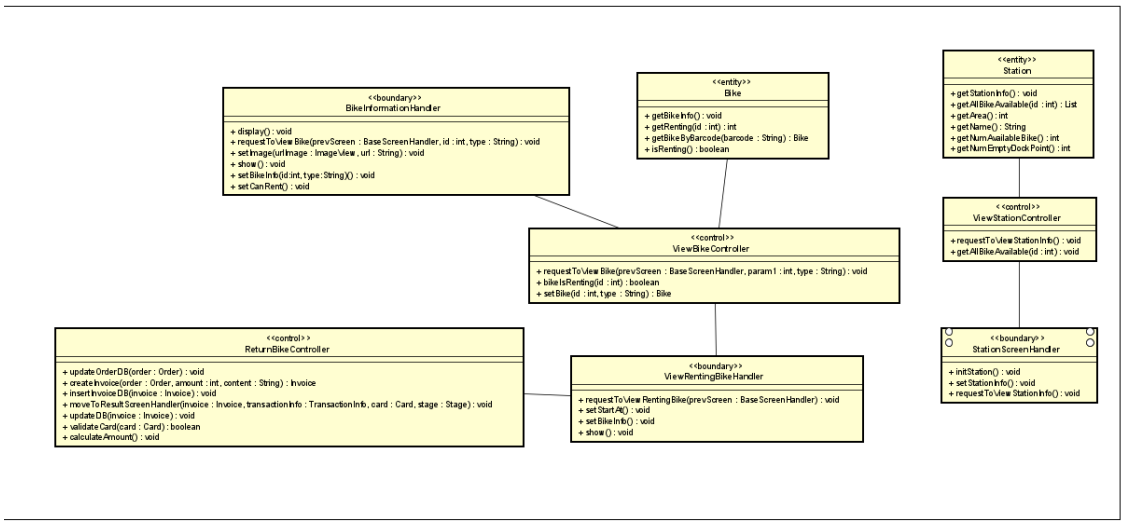


Fig 12. Analysis Class Diagram for View Bike or Station information

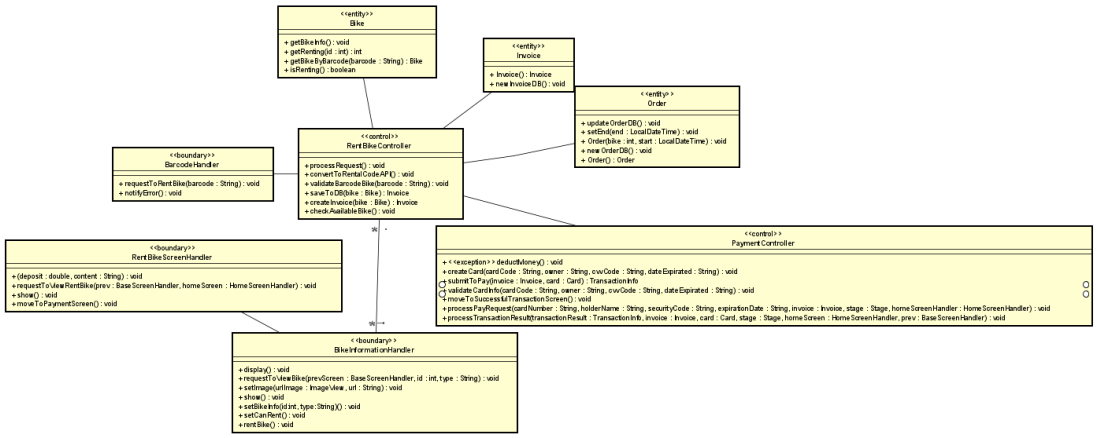


Fig 13. Analysis Class Diagram for Rent bike

3.4 Unified Analysis Class Diagram

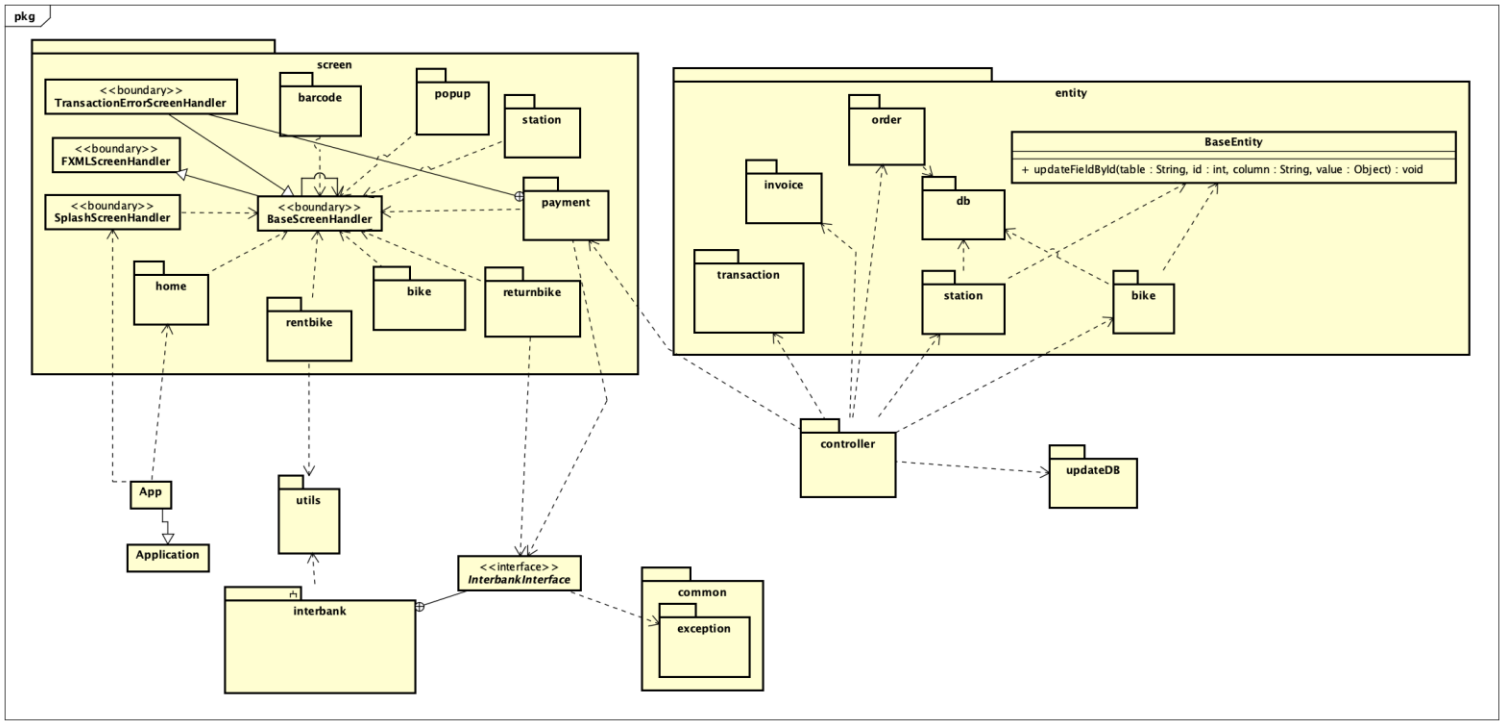


Fig 14. Unified Analysis Class Diagram

3.5 Security Software Architecture

In this project, we will not consider features such as user authentication (e.g., sign up, sign in, sign out), but we focus on features related to bike renting and returning.

4 Detailed Design

4.1 User Interface Design

4.1.1 Screen Configuration Standardization

Display

Number of colors supported: 16,777,216 colors

Resolution: 1366 × 768 pixels

Screen

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

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

- Display of the screen title: The title is located at the top of the frame in the middle.

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

Control

- Size of the text: medium size (mostly 24px). Font: Arial. Color: # 201C1C

- 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

3. View Bike information screen – view information of a bike before renting

4. View Renting Bike information screen – view information of a renting bike

5. View Dock information screen – view information of a dock

6. View distance screen – view distance from user's location to selected dock

7. Payment screen – fill payment information

8. Transaction Error screen – display detailed error of a transaction

9. Return Bike screen – display information to return a bike

10. Select Dock to return Bike screen – display list of docks to return a bike

11. Rent Bike screen – display detailed information for renting a bike

12. Invoice screen – display detailed invoice

13. Enter barcode screen – display a text area for entering barcode to rent bike

Direct input from the keyboard

There will be no shortcuts. There are back buttons to move back to the previous screen.

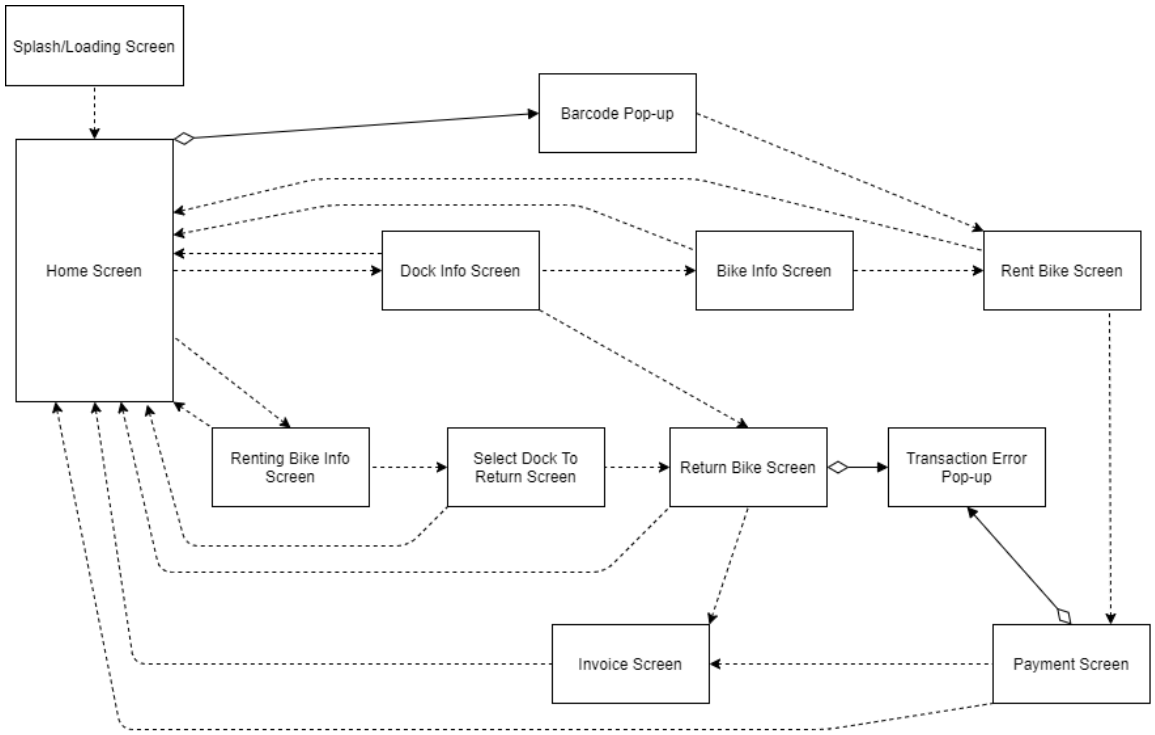
There is a Home button located at the top right of screen to go to Home screen.

Also, there are a close button "X" to close the screen, a resize button to resize screen and a minimize button " _ " to shrinks the window and places it on the taskbar while leaving the software running located at the title bar to the right.

Error

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

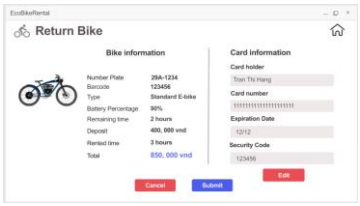
4.1.2 Screen Transition Diagrams



4.1.3 Screen Specifications

4.1.3.1. Return bike screen

Screen specification

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Return Bike screen	7/11/2020			Trần Thị Hằng
		Control	Operation	Function	
		Area for displaying Bike information	Initial	Display detail information of renting bike	
		Area for displaying Card information	Initial	Display detail information of Card which will be used for payment	
		Edit button	Click	Display the Payment Screen	
		Cancel button	Click	Display Home Screen	
		Submit button	Click	Display Invoice Screen/Transaction Error Screen	

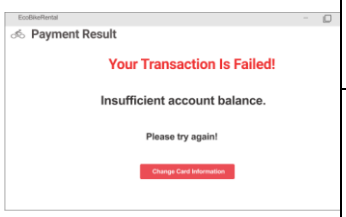
Defining the field attributes

Screen name	Return bike screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks

Number Plate	10	String	Black	Left justified
Barcode	6	String	Black	Left justified
Type	20	String	Black	Left justified
Battery Percentage	3	Numeral	Black	Left justified
Remaining time	2	Numeral	Black	Left justified
Deposit	10	Numeral	Black	Left justified
Rented time	5	Numeral	Black	Left justified
Total	20	Numeral	Black	Left justified
Card holder	50	String	Black	Left justified
Card number	20	Numeral	Black	Left justified
Expiration Date	10	String	Black	Left justified
Security Code	10	Numeral	Black	Left justified

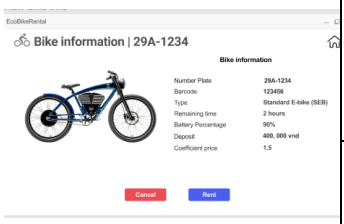
4.1.3.2. Transaction Error Screen

Screen specification

<table>
<tr><td colspan="2">EcoBikeRental Software</td><td>Date of creation</td><td>Approved by</td><td>Reviewed by</td><td>Person in charge</td></tr>
<tr><td>Screen specification</td><td>Transaction Error Screen</td><td>7/11/2020</td><td></td><td></td><td>Trần Thị Hằng</td></tr>
<tr><td colspan="2" rowspan="3"> 
</td><td>Control</td><td>Operation</td><td colspan="2">Function</td></tr>
<tr><td>Change Card Information button</td><td>Click</td><td colspan="2">Display Payment Screen</td></tr>
<tr><td>Area for display error</td><td>initial</td><td colspan="2">Display error of transaction</td></tr>
</table>

4.1.3.3. View Bike Information Screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	View bike information screen	07/11/2020			Dương Thị Huê
		Control	Operation	Function	
		Area for displaying the number plate	Initial	Display the number plate	
		Area for display image of bike	Initial	Display image	
		Area for display barcode	initial	Display the barcode	
		Area for display remaining time	initial	Display remaining time of bike if it is electric bike	
		Area for display battery percentage	initial	Display battery percentage of bike if it is electric bike	
		Area for display deposit	initial	Display deposit	
		Area coefficient price	initial	Display coefficient price to calculate amount	
		Cancel button	Click	Back to station information screen	
		Rent button	Click	Move to payment screen	

	Home icon	Click	Back to home

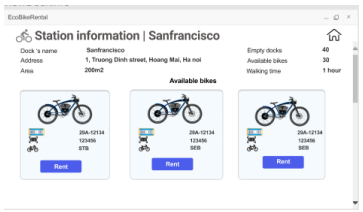
Defining the field attributes

Screen name	View bike information screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Number plate	8	Numeral	Black	Left-justified
Barcode	6	Numeral	Black	Left justified
Type	100	Text	Black	Left-justified
Remaining time	50	Text	Black	Left-justified
Battery percentage	4	Text	Black	Left-justified
Deposit	6	Number	Black	Left-justified
Coefficient price	3	Number	Black	Left-justified

4.1.3.4. View Station Information Screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	View station information screen	07/11/2020			Dương Thị Huê
		Control	Operation	Function	
		Area for displaying the	Initial	Display the name of dock	

	Dock ‘s name		
	Area for display dock ‘s address	Initial	Display dock ‘s address
	Area for display dock ‘s area	initial	Display the area of the dock
	Area for display number available bike	initial	Display remaining number available bike
	Area for display the number of empty docks	initial	Display number empty docks
	Area for walking time	initial	Display walking time from current user location to dock
	Area for display list available bike and its informatio n	initial	Display list available bike in the dock and its information
	Home icon	Click	Back to home
	Rent button	Click	Move to payment screen

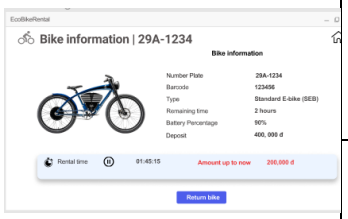
Defining the field attributes

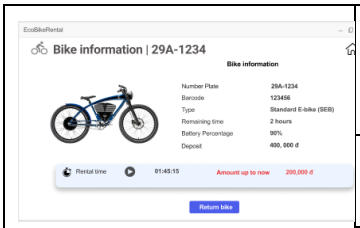
Screen name	View bike information screen	

Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Number plate	8	Numeral	Black	Left-justified
Barcode	6	Numeral	Black	Left justified
Type	100	Text	Black	Left-justified

4.1.3.5. View Renting Bike Screen

Screen specification

<table>
<tr><td colspan="2">EcobikeRental Software</td><td>Date of creation</td><td>Approved by</td><td>Reviewed by</td><td>Persion in charge</td></tr>
<tr><td>Screen specification</td><td>View renting bike information screen</td><td>07/11/2020</td><td></td><td></td><td>Dương Thị Huê</td></tr>
<tr><td colspan="2" rowspan="4"> 
</td><td>Control</td><td>Operation</td><td colspan="2">Function</td></tr>
<tr><td>Area for displaying the bike information</td><td>Initial</td><td colspan="2">Display bike information</td></tr>
<tr><td>Area for display Renting time</td><td>initial</td><td colspan="2">Display the renting time</td></tr>
<tr><td>Area for display amount up to now</td><td>initial</td><td colspan="2">Display amount to be paid up to now</td></tr>
<tr><td colspan="2" rowspan="2"></td><td>Area for home icon</td><td>click</td><td colspan="2">Back to home</td></tr>
<tr><td>Area for display deposit</td><td>initial</td><td colspan="2">Display deposit</td></tr>
</table>

	Area coefficient price	initial	Display coefficient price to calculate amout
	Area for display type	inititial	Display type of bike
	Pause icon	click	Stop counting clock
	Play icon	Click	Continue counting clock

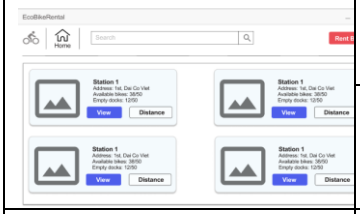
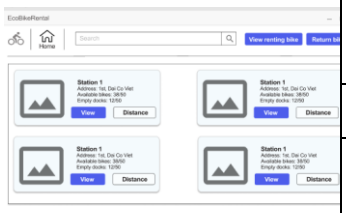
Defining the field “Renting time”attributes

Screen name	View bike information screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Amount up to now	8	Numeral	red	Left-justified
Retal time	8	Hh:mm:ss	black	Left-justified

4.1.3.6. Home Screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Review ed by	Persion in charge
Screen specificati on	Home screen	07/11/202 0			Đỗ Minh Thông
		Control	Operation	Function	
		Area for	Initial	Display list of docks	

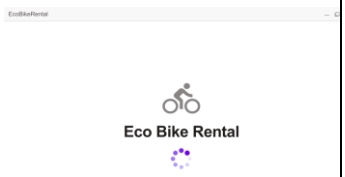
	displaying the list of docks		
	View button	Click	Move to view station information screen
	Distance button	Click	View distance popup
	Home icon	click	Reload home screen
	Rent bike button	click	Enter barcode and move to rent bike screen
	View renting bike button	Click	Move to view renting bike screen
	Return bike button	Click	Move to return bike screen
	Area for search	Type	Filter all dock then display

Defining the field “Station information” attributes

Screen name	View bike information screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Station name	50	Text	Black	Left-justified
Address	50	Text	Black	Left-justified
Available bike	11	Number	Black	Left-justified
Empty dock point	11	Number	Black	Left-justified

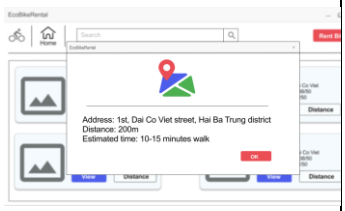
4.1.3.7. Splash Screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	Splash screen	07/11/2020			Đỗ Minh Thông
		Control	Operation	Function	
		Area for displaying the loading screen	Initial	Display loading screen	

4.1.3.8. View Distance Popup Screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	View distance popup screen	07/11/2020			Đỗ Minh Thông
		Control	Operation	Function	
		Area for displaying the station information and walking time estimate	Initial	Display station information and walking time estimate	
		OK button	Click	Move to home screen	

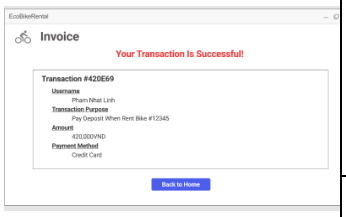
Defining the field attributes

Screen name	View bike information screen			

Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Station information	256	Text	Black	Left-justified
Distance	50	String	Black	Left-justified
Estimated time	50	String	Black	Left-justified

4.1.3.9 Invoice screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	Invoice Screen	07/11/2020			Trần Thị Hằng
		Control	Operation	Function	
		Area for displaying transaction result	Initial	Display transaction result	
		Area for display username	initial	Display the user name	
		Area for display transaction content	initial	Display transaction content	
		Back to home button	click	Back to home	
		Area for display amount	initial	Display amount of invoice	

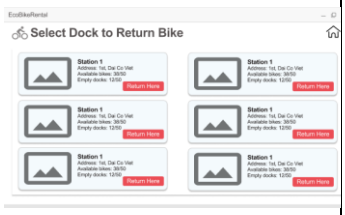
	Area for payment method	initial	Display payment method

Defining the field “Renting time”attributes

Screen name	View bike information screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
username	50	String	black	Left-justified
Transaction content	50	String	black	Left-justified
amount	11	String	black	Left-justified

4.1.3.10 Select dock to return

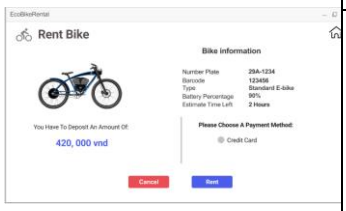
Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	Select dock to return screen	07/11/2020			Trần Thị Hằng
		Control	Operation	Function	
		Area for	Initial	Display list of docks	

	displaying the list of docks		
	Reuturn button	Click	Move to return bike screen
	Home icon	click	Reload home screen
	Area for display dock's infomation	Initial	Display dock 's information

4.1.3.11 Rent bike screen

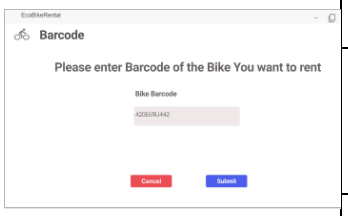
Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	Rent bike screen	07/11/2020			PhạmNhật Linh
		Control	Operation	Function	
		Area for displaying bike information	Initial	Display bike information (like view bike screen)	
		Rent button	Click	Move to payment screen	
		Home icon	click	Reload home screen	
		Cancel button	Click	Back to previous screen	
		Area for display amount	Intitial	Deposit amount	

4.1.3.12 Barcode screen

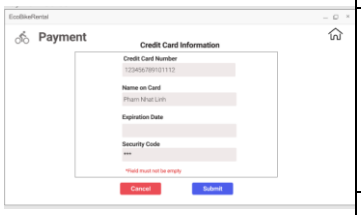
Screen specification

EcobikeRental Software	Date of creation	Approved by	Reviewed by	Persion in charge

Screen specification	Barcode screen	07/11/2020			Đỗ Nhật Linh
		Control	Operation	Function	
		Area for Entering barcode	Initial	Area for entering barcode	
		Submit button	Click	Move to rent bike screen	
		Cancel button	Click	Back to home	

4.1.3.13 Payment screen

Screen specification

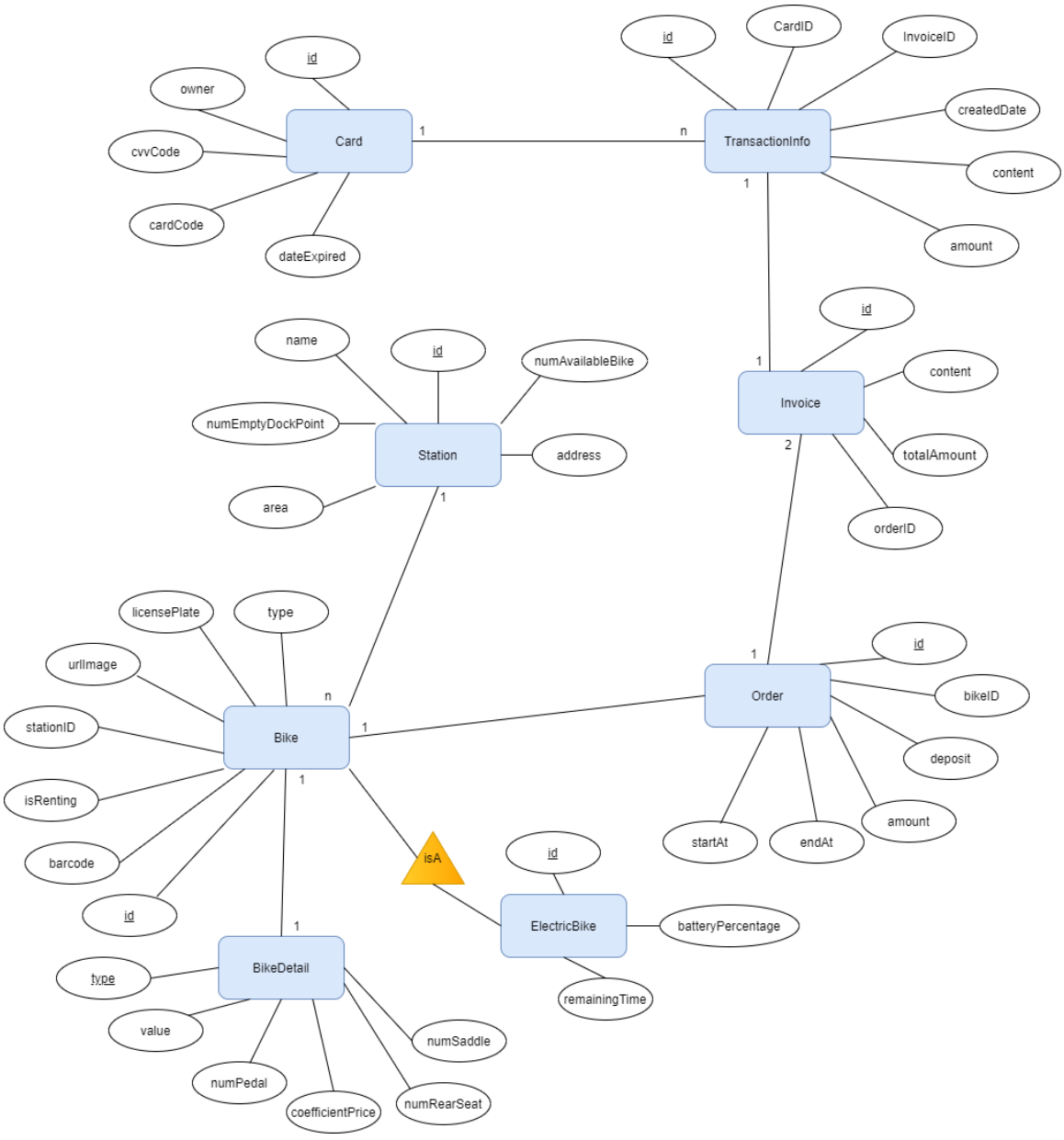
EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	*Payment* screen	07/11/2020			Phạm Nhật Linh
		Control	Operation	Function	
		Area for Entering Card information	Initial	Area for entering card information	
		Submit button	Click	Move to result screen	
		Cancel button	Click	Back to previous screen	
		Area for display error	intitial	Display error when submit	

Defining the field attributes

Screen name	Return bike screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Card holder	50	Text	Black	Left justified
Card number	30	Text	Black	Left justified
Expiration Date	10	Text	Black	Left justified
Security Code	6	Text	Black	Left justified

4.2 Data Modeling

4.2.1 Conceptual Data Modeling



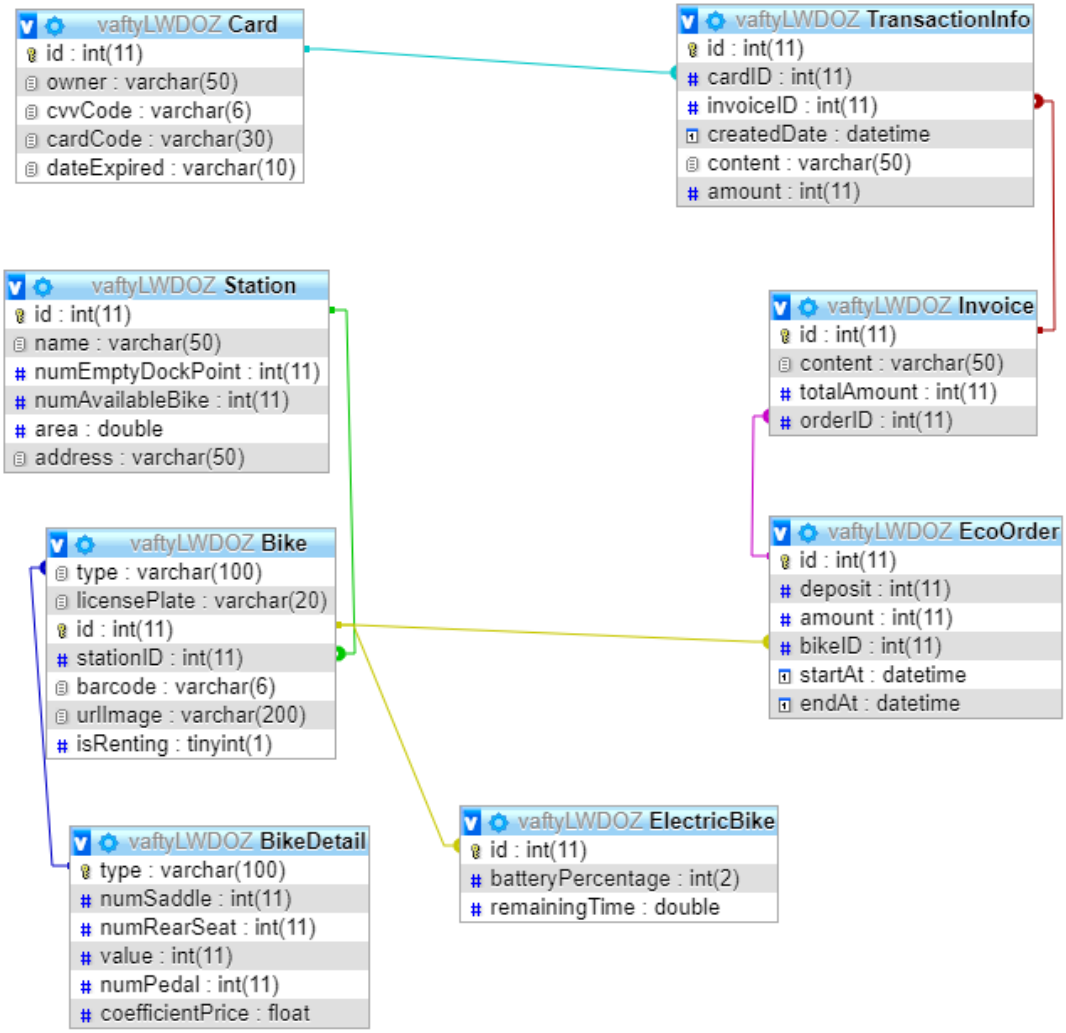
4.2.2 Database Design

4.2.2.1 Database Management Systems

- Database Management System: MySql

- MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.

4.2.2.2 Logical Data Model



4.2.2.3 Physical Data Model

- **Card**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	ID of card, auto increment

2			owner	Varchar(50)	Yes	Name of Card's holder
3			cvvCode	Varchar(6)	Yes	Security code of card
4			cardCode	Varchar(30)	Yes	Card code number
5			dateExpired	Varchar(10)	Yes	Expiration date of card

- **TransactionInfo**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	ID of transaction, auto increment
2			content	Varchar(50)	Yes	Content of transaction
3		x	cardID	Integer	Yes	cardID, same as ID of card which is used for the transaction
4		x	invoiceID	Integer	Yes	invoiceID, same as ID of invoice which belongs to the transaction
5			createDate	datetime	Yes	Creation date of the transaction
6			amount	double	Yes	Total amount of money is used for the transaction

- **Station**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	ID of station, auto increment
2			name	Varchar(50)	Yes	Name of the station
3			numEmptyDockPoint	Integer	Yes	Number of empty dock points in the station
4			numAvailableBike	Integer	Yes	Number of available bike for renting in the station
5			area	double	Yes	Area of the station
6			address	varchar(50)	Yes	Address of the station

- **Bike**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	id of the bike , auto increment
2		x	type	Varchar(100)	Yes	type of bike
3			licensePlate	Varchar(20)	Yes	license plate of the bike

4		x	stationID	Integer	Yes	Station to which this bike belongs
5			urlImage	Varchar(200)	Yes	Url of bike image
6			isRenting	Tinyint(1)	Yes	To check if the bike has been rented or not
7			Barcode	Varchar(6)	Yes	Barcode of the bike

- **ElectricBike**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x	X	id	Integer	Yes	id of the bike
2			batteryPercentage	Integer	Yes	percentage of battery
3			remainingTime	double	Yes	remaining time of the bike

- **BikeDetail**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		Type	Varchar(100)	Yes	type of the bike
2			numSaddle	Integer	Yes	Number of saddle of the bike
3			numRearSeat	Integer	Yes	Number of rear seat of the bike
4			Value	Integer	Yes	Value of the bike
5			numPedal	Integer	Yes	Number of Pedal
6			coefficientPrice	Float	Yes	Coefficient when deposit

- **Order**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	ID of order, auto increment
2			deposit	Integer	Yes	Amount of deposit when renting bike
3			startAt	datetime	Yes	When user rents bike
4			endAt	datetime	No	When the user returns bike
5			Amount	Integer	No	The amount of renting money (not include deposit)
6		x	bikeID	integer	Yes	The id of the bike that user is/was renting

- **Invoice**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	ID of the invoice, auto increment
2			content	Varchar(50)	Yes	The content of the

						Invoice
3			totalAmount	Integer	Yes	The amount of money for the transaction
4		x	orderID	Integer	Yes	The Order of which this invoice is used for

- **Database Script:**

```
CREATE TABLE `Bike` (
 `type` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
 `licensePlate` varchar(20) COLLATE utf8_unicode_ci DEFAULT NULL,
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `stationID` int(11) DEFAULT NULL,
 `barcode` varchar(6) COLLATE utf8_unicode_ci NOT NULL,
 `urlImage` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
 `isRenting` tinyint(1) DEFAULT NULL,
 PRIMARY KEY (`id`),
 KEY `bike_type` (`type`),
 KEY `bike_station` (`stationID`),
 CONSTRAINT `bike_station` FOREIGN KEY (`stationID`) REFERENCES `Station` (`id`),
 CONSTRAINT `bike_type` FOREIGN KEY (`type`) REFERENCES `BikeDetail` (`type`)
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
CREATE TABLE `BikeDetail` (
 `type` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
```

```
  `numSaddle` int(11) DEFAULT NULL,
  `numRearSeat` int(11) DEFAULT NULL,
  `value` int(11) DEFAULT NULL,
  `numPedal` int(11) DEFAULT NULL,
  `coefficientPrice` float DEFAULT NULL,
  PRIMARY KEY (`type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci

CREATE TABLE `Card` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `owner` varchar(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  `cvvCode` varchar(6) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  `cardCode` varchar(30) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  `dateExpired` varchar(10) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci

CREATE TABLE `EcoOrder` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `deposit` int(11) NOT NULL,
  `amount` int(11) DEFAULT NULL,
  `bikeID` int(11) NOT NULL,
  `startAt` datetime NOT NULL,
  `endAt` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `bikeID` (`bikeID`),
  CONSTRAINT `EcoOrder_ibfk_1` FOREIGN KEY (`bikeID`) REFERENCES `Bike` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=46 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
```

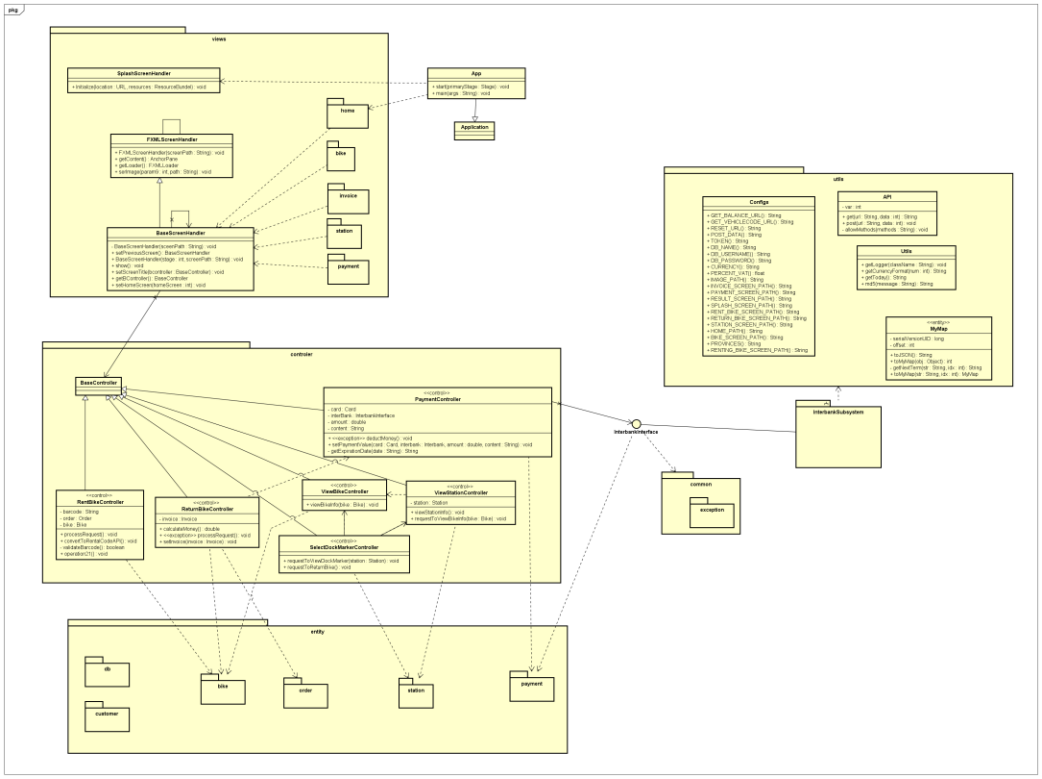
```
CREATE TABLE `ElectricBike` (
  `id` int(11) NOT NULL,
  `batteryPercentage` int(2) DEFAULT NULL,
  `remainingTime` double DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `ElectricBike_ibfk_1` FOREIGN KEY (`id`) REFERENCES `Bike` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
CREATE TABLE `Invoice` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `content` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `totalAmount` int(11) NOT NULL,
  `orderID` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `orderID` (`orderID`),
  CONSTRAINT `Invoice_ibfk_1` FOREIGN KEY (`orderID`) REFERENCES `EcoOrder` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=104 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
CREATE TABLE `Station` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `numEmptyDockPoint` int(11) NOT NULL,
  `numAvailableBike` int(11) NOT NULL,
  `area` double NOT NULL,
  `address` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
CREATE TABLE `TransactionInfo` (
```

```
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `cardID` int(11) DEFAULT NULL,
 `invoiceID` int(11) DEFAULT NULL,
 `createdDate` datetime DEFAULT NULL,
 `content` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL,
 `amount` int(11) DEFAULT NULL,
 PRIMARY KEY (`id`),
 KEY `cardID` (`cardID`),
 KEY `invoiceID` (`invoiceID`),
 CONSTRAINT `TransactionInfo_ibfk_1` FOREIGN KEY (`cardID`) REFERENCES `Card` (`id`),
 CONSTRAINT `TransactionInfo_ibfk_2` FOREIGN KEY (`invoiceID`) REFERENCES `Invoice` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=36 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
```

4.3 Non-Database Management System Files

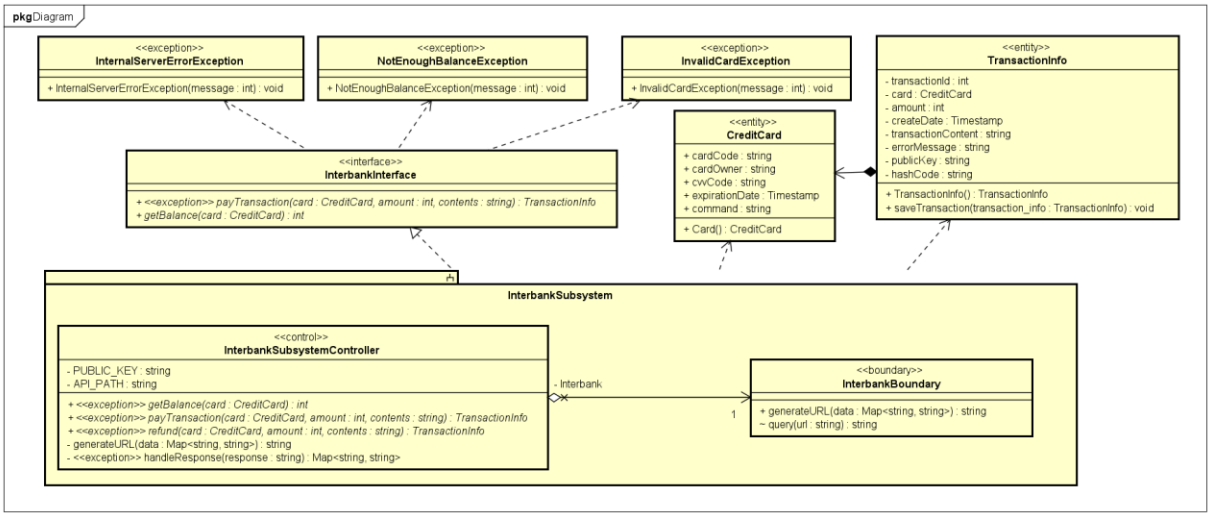
4.4 Class Design

4.4.1 General Class Diagram

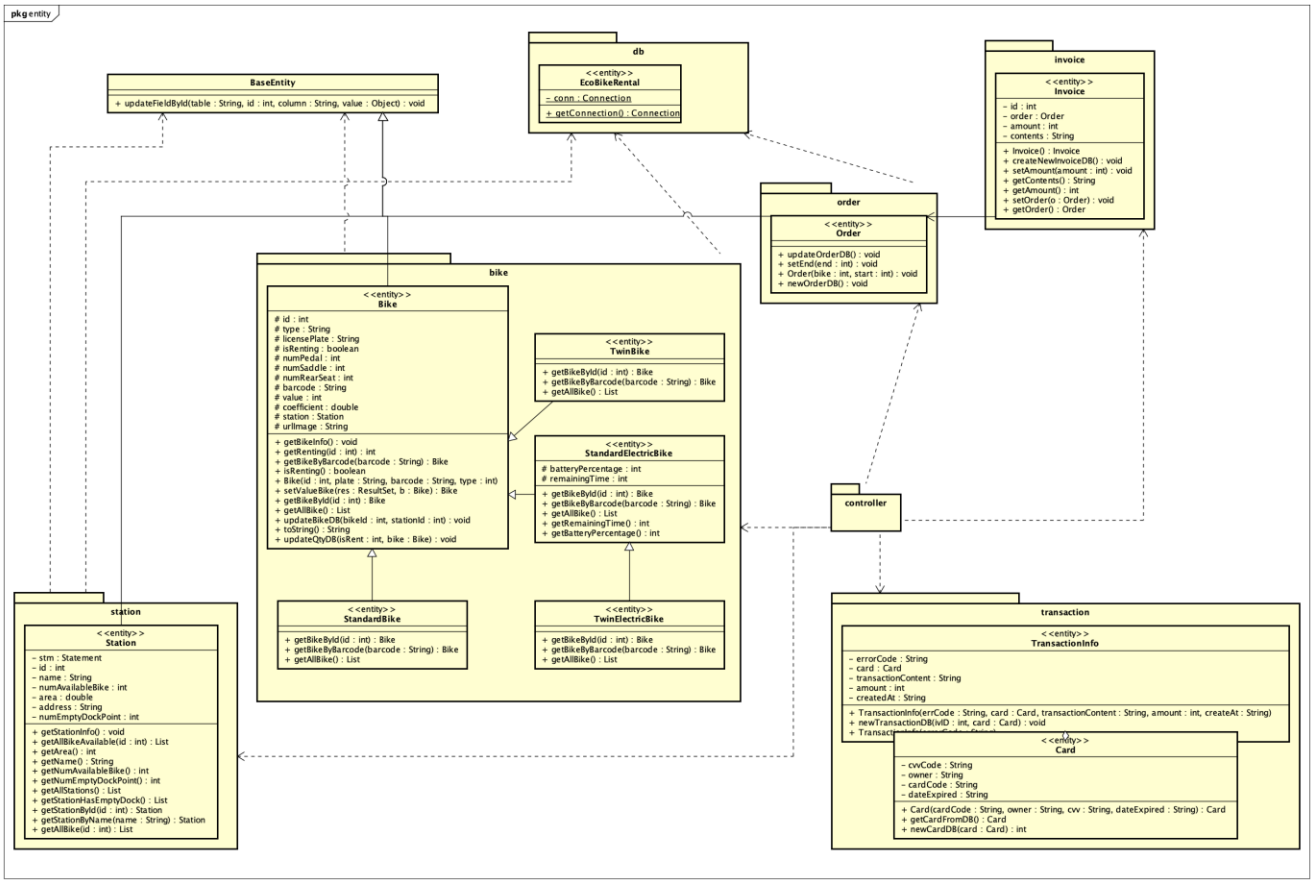


4.4.2 Class Diagrams

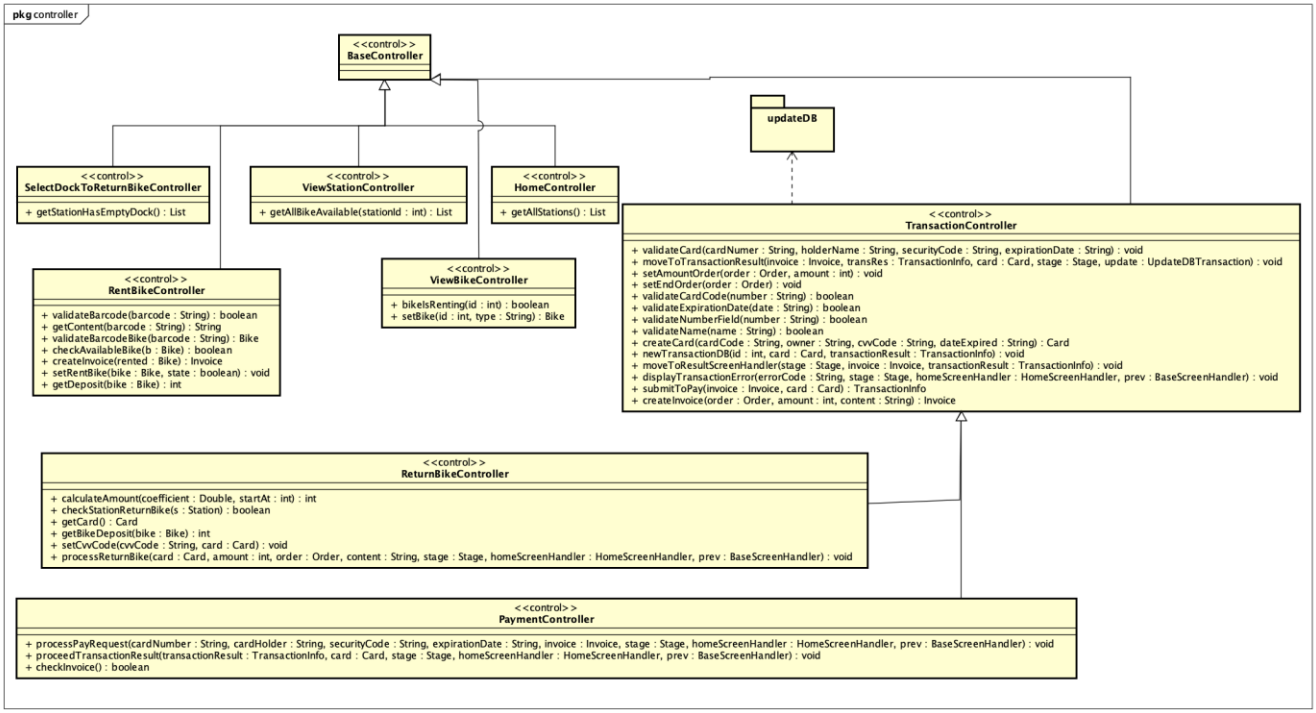
4.4.2.1 Class Diagram for Interbank Subsystem



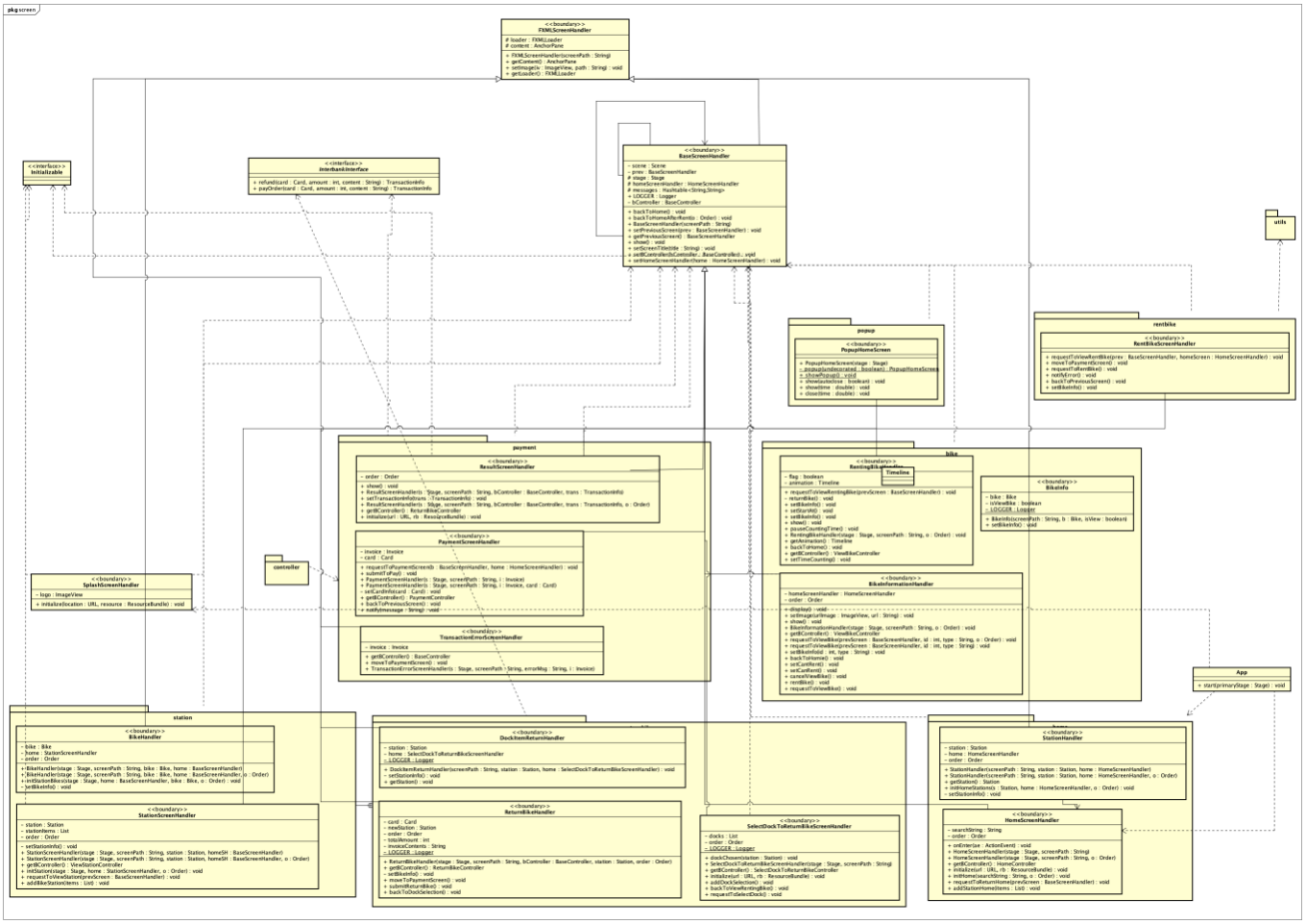
4.4.2.2 Class Diagram for Entity package



4.4.2.3 Class Diagram for Controller package

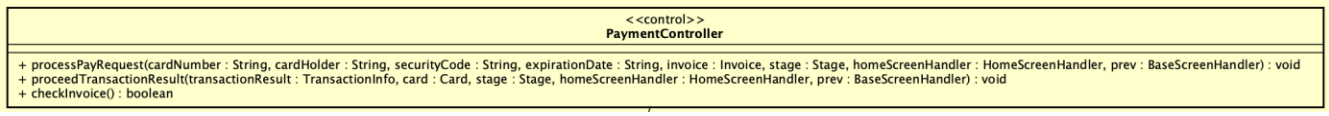


4.4.2.4 Class Diagram for Screen package



4.4.3 Class Design

4.4.3.1 Class "PaymentController"



Operation

#	Name	Return type	Description (purpose)
1	processPayRequest	void	Process pay request from user
2	proceedTransactionResult	void	Process transaction result, decide which screen to go

Exception: None

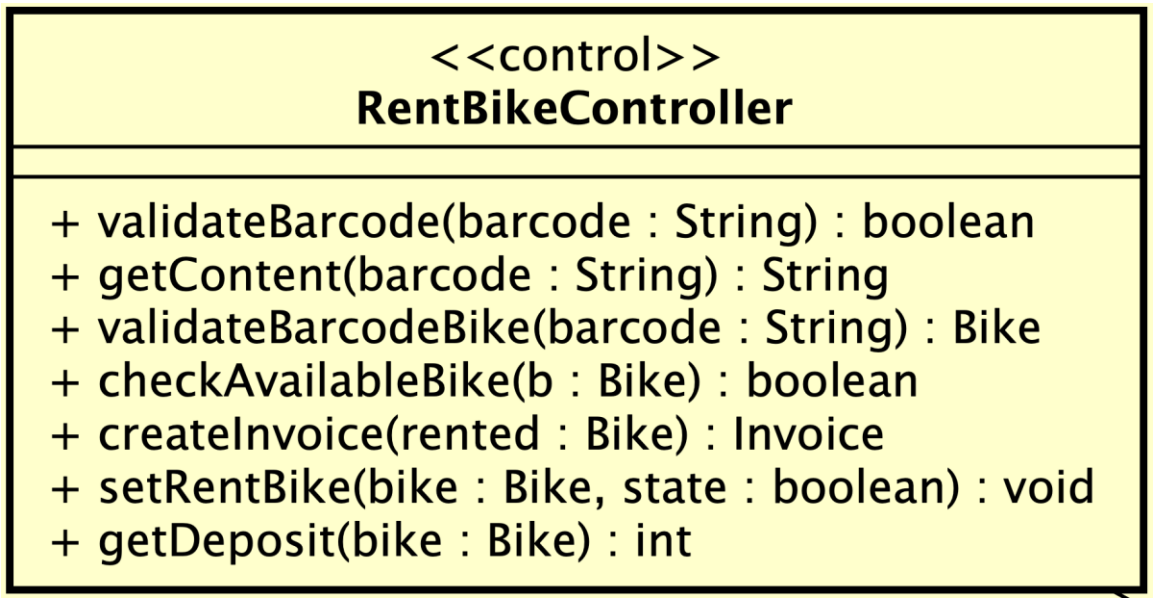
Parameter:

- card – the credit card used for payment
- stage – stage of screen
- homeScreenHandler – home screen handler
- prev – base screen handler
- transactionResult – result of transaction use to process
- cardNumber – number of card
- cardHolder – card's owner
- securityCode – cvvCode
- expirationDate – date expired of card
- invoice – Invoice of pay request

State : None

Method : None

4.4.3.2 Class "RentBikeController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	validateBarcode	boolean	Validate input barcode
2	getContent	String	Get content for the order
3	validateBarcodeBike	Bike	Get bike from given barcode

4	CheckAvailableBike	boolean	Check if bike has been rented or not
5	createInvoice	Invoice	Create invoice for transaction
6	setRentBike	void	Set isRenting for bike
7	getDeposit	int	Get deposit amount of bike

Parameter:

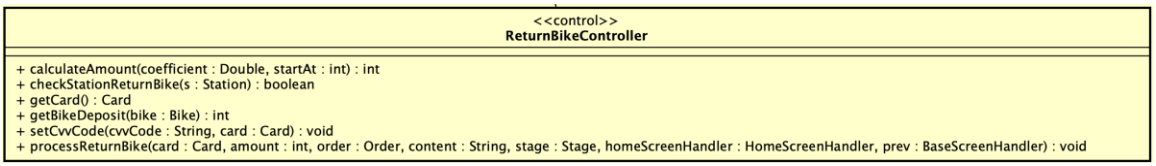
- Barcode: barcode use to rent bike
- Bike : bike use to rent
- State: state set for *isRenting* attribute of bike

Exception: None

Method: None

State: None

4.4.3.3 Class "ReturnBikeController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	calculateAmount	int	Calculate total amount that customer has to pay when return bike
2	processReturnBike	void	Process returning bike request
3	checkStationReturnBike	boolean	check station have empty dock point to return bike
4	getCard	Card	Get card's information from database
5	setCvvCode	void	Set cvvCode for card from user
6	getBikeDeposit	int	Get deposit of given bike

Parameter:

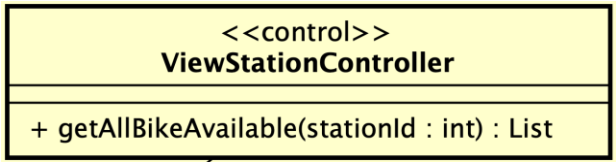
- Coefficient: coefficient of bike use to calculate total amount
- StartAt: start time when renting bike
- S: station will use to check for return bike
- Bike: Bike use for get deposit
- Card: Card use for transaction
- Amount: total amount user has to pay
- Order: Order use to make invoice
- Content: content of invoice
- stage – stage of screen
- homeScreenHandler – home screen handler
- prev – base screen handler

Exception: None

State: None

Method: None

4.4.3.4 Class "ViewStationController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	getAllBikeAvailable	List	Get all bikes are available for renting

Parameter:

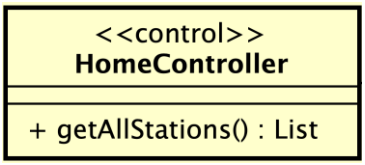
– StationId: id of station use to get bike list

Exception: None

Method None

State None

4.4.3.5 Class "HomeController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	getAllStations	List	Get all stations to display

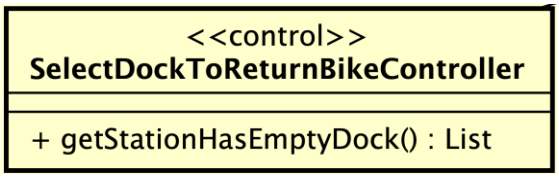
Parameter: None

Exception: None

Method None

State None

4.4.3.6 Class "SelectDockToReturnBikeController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	getStationHas EmptyDock	List	Get all stations have empty dock to return bike

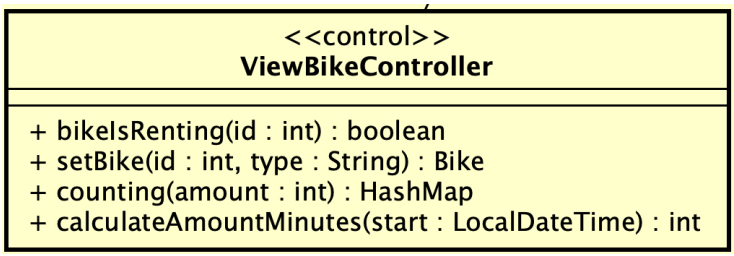
Parameter: None

Exception: None

Method: None

State: None

4.4.3.7 Class "ViewBikeController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	bikeIsRenting	boolean	Check if a bike is renting by id
2	setBike	Bike	create bike instance base on id and type
3	counting	HashMap	count 1 second
4	calculateAmountMinutes	int	convert time in localDateTime to second

Parameter:

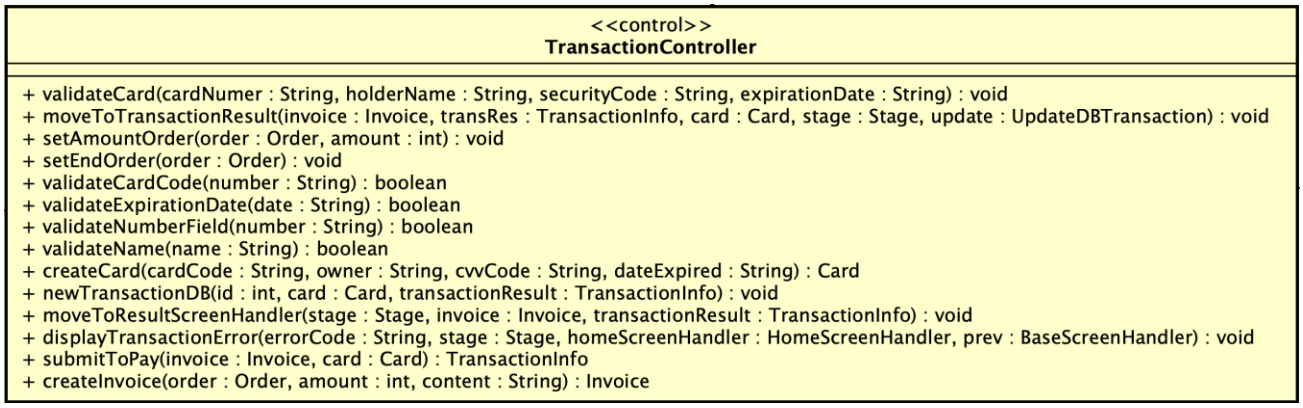
- Id: id of bike
- Type: type of the bike
- Amount: amount that wants to count
- Start: start time use for counting

Exception: None

Method: None

State: None

4.4.3.8 Class "TransactionController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	validateCard	void	Validate card
2	moveToTransactionResult	void	Move to TransactionResult screen
3	setAmountOrder	void	Set amount for order
4	setEndOrder	void	Set end time for order
5	validateCardCode	boolean	Validate card code
6	validateExpirationDate	boolean	Validate expiration date
7	validateNumberField	boolean	validate Security Code/cvvCode
8	validateName	boolean	Validate card's owner
9	createCard	Card	Create new card
10	newTransactionDB	void	insert newTransaction to database
11	moveToResultScreenHandler	void	move to resultScreenHandler
12	displayTransactionError	void	move to transaction error screen
13	submitToPay	TransactionInfo	Submit to pay for transaction
14	createInvoice	Invoice	create invoice

Parameter:

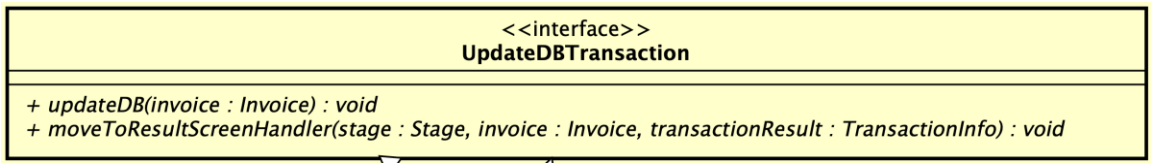
- card – the credit card used for payment
- stage – stage of screen
- homeScreenHandler – home screen handler
- prev – base screen handler
- transactionResult – result of transaction use to process
- cardNumber – number of card
- cardHolder – card's owner
- securityCode – cvvCode
- expirationDate – date expired of card
- invoice – Invoice of pay request
- TransRes – transaction result
- Update – UpdateDBTransaction interface
- ErrorCode – error code responsed when process transaction
- Amount – total amount of transaction
- Number – input for validation

Exception: None

Method: none

State: none

4.4.3.9 Class "UpdateDBTransaction"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	updateDB	void	insert invoice, insert/update order of the invoice to database
2	moveToResult ScreenHandler	void	Move to result screen

Parameter

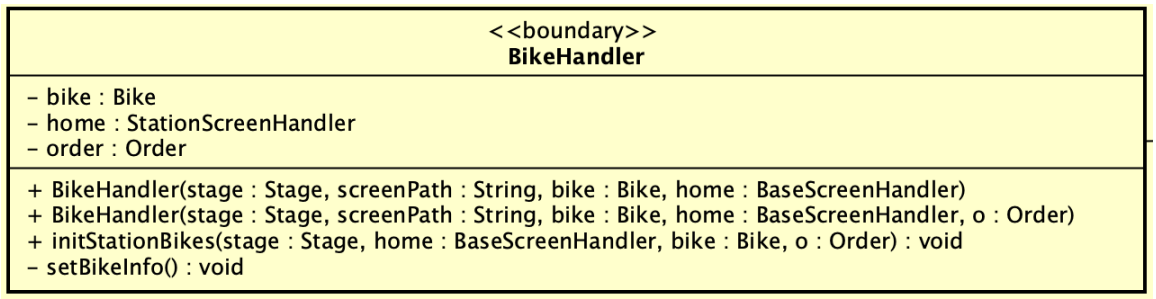
- Invoice: invoice use to update database
- Stage: stage of screen
- TransactionResult: result of transaction

Exception: none

Method: none

State: none

4.4.3.10 Class "BikeHandler"



Attribute

#	Name	Data type	Default value	Description
1	bike	Bike	NULL	Bike whose infomation will be displayed
2	order	Order	NULL	Order of renting bike if any
3	home	StationScreenHandler	NULL	StationScreenHandler which BikeHandler attached to

Operation

#	Name	Return type	Description (purpose)
1	initStationBikes	void	initialize station with bikes
2	setBikeInfo	void	Set information of bike to screen

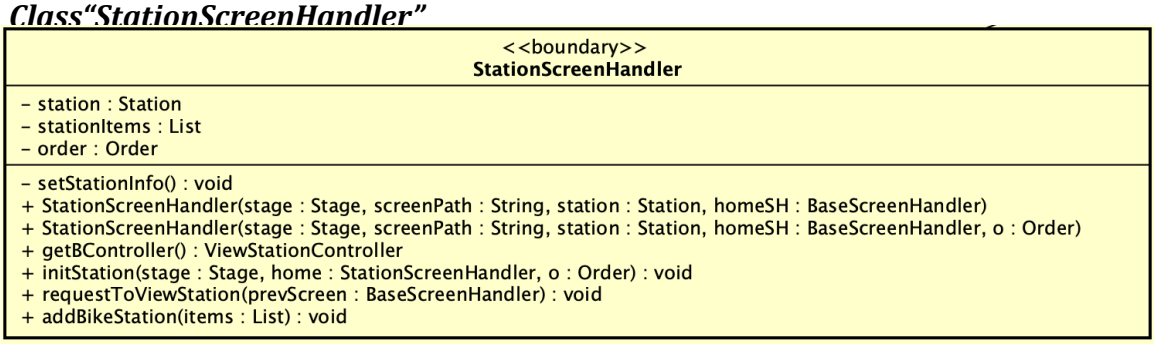
Parameter

- bike - current bike
- stage – stage of screen
- homeScreenHandler – home screen handler
- O – current order

Method: none

State: none

4.4.3.11 Class"StationScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	stationItems	List	NULL	List of all stations
2	order	Order	NULL	Current order if any
3	station	Station	NULL	Current station

Operation

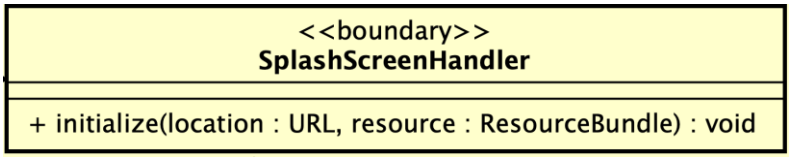
#	Name	Return type	Description (purpose)
1	getBController	ViewStation Controller	Get base controller
2	setStationInfo	void	Set station information
3	initStation	void	Initial station information
4	requestToView Station	void	Request to view a station
5	addBikeStation	void	Add bike to station

Parameter:

- stage – stage of screen
- ScreenPath – path to fxml screen
- homeSH - base screen handler
- O - current order
- Items – list of bikes add to station

Method: none

4.4.3.12 Class "SplashScreenHandler"



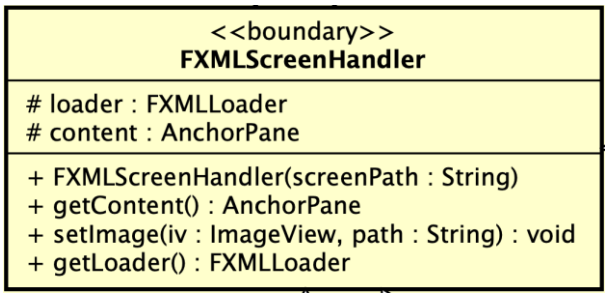
Attribute

Operation:

Method: none

State: none

4.4.3.13 Class "FXMLScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	loader	FXMLLoader	null	
2	content	AnchorPane	null	

Operation:

#	Name	Return type	Description (purpose)
1	getContent	AnchorPane	Get content of anchorPane
2	setImage	void	Set image to ImageView
3	getLoader	FXMLLoader	Get loader

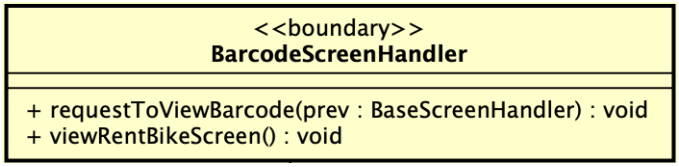
Parameter:

- ScreenPath – path to fxml screen
- iv – image view
- Path – path to image use to set

Method: none

State: none

4.4.3.14 Class "BarcodeScreenHandler"



Attribute

Operation:

#	Name	Return type	Description (purpose)
1	requestToViewBarcode	void	move to the barcode screen
2	viewRentBikeScreen	void	move to the rent bike screen

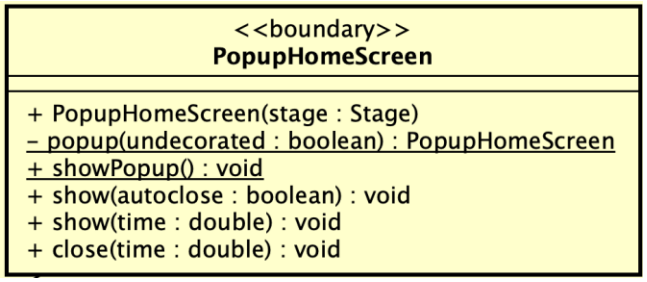
Parameter:

- prev – base screen handler

Method: none

State: none

4.4.3.15 Class "PopupHomeScreen"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	popup	PopupHomeScreen	style the pop-up
2	showPopup	void	Show popup
3	show	void	show and auto close after sometime
4	show	void	show and close after sometime

5	close	void	close the pop-up after sometime

Parameter

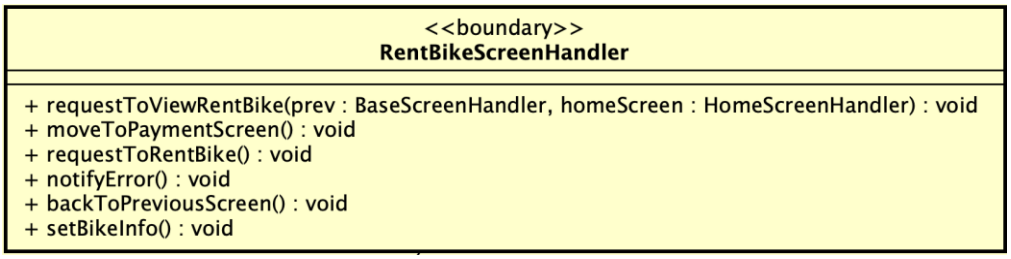
- Stage: current stage
- Autoclose: set autoclose for popup
- Time: amount of time that popup will close

Exception: none

Method: none

State: none

4.4.3.16 Class "RentBikeScreenHandler"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	notifyError	void	Notify error
2	requestToViewRentBike	Void	Process to request to view renting bike information
3	moveToPaymentScreen	void	Move to payment screen
4	requestToRentBike	void	Request to rent bike
5	backToPreviousScreen	void	Back to previous screen
6	setBikeInfo	void	Set bike's information

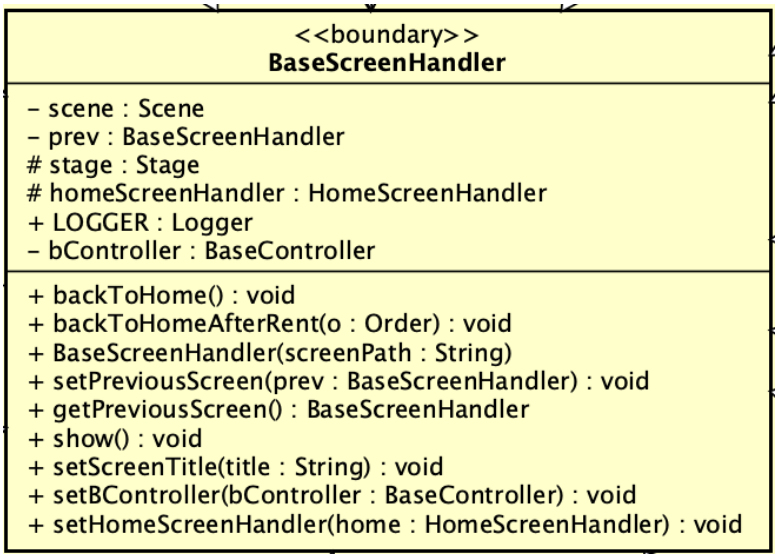
Parameter

- Prev – BaseScreenHandler
- HomeScreen - HomeScreenHandler

Method: none

State: none

4.4.3.17 Class "BaseScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	scene	Scene	NULL	scene
2	prev	BaseScreenHandler	NULL	Previous screen
3	stage	Stage	NULL	Current stage
4	homeScreenHandler	HomeScreenHandler	NULL	Home screen handler
5	LOGGER	Logger	NULL	Logger
6	bController	BaseController	NULL	Base controller

Operation

#	Name	Return type	Description (purpose)
1	backToHome	void	Back to home before renting
2	backToHomeAfterRent	void	Back to home after renting
3	setPreviousScreen	void	Set previous screen

4	getPreviousScreen	BaseScreen Handler	get previous screen
5	show	void	Show this screen
6	setScreenTitle	void	Set title for screen
7	setBController	void	Set bController
8	setHomeScreenHandler	void	Set home screen

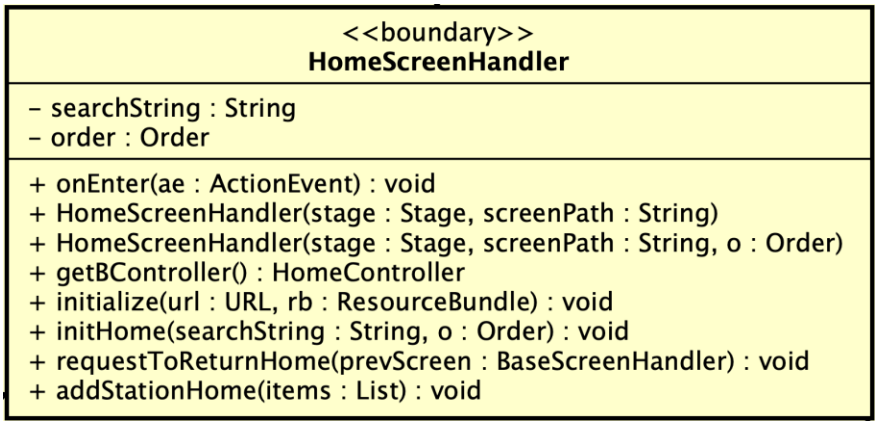
Parameter

- Prev – BaseScreenHandler
- HomeScreen - HomeScreenHandler
- Order – current order
- ScreenPath – path to file fxml screen
- Title – title of screen
- BController – BaseController

Method: none

State: none

4.4.3.18 Class "HomeScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	searchString	String	null	String use to search
2	order	Order	null	Current order if any

Operation

#	Name	Return type	Description (purpose)
1	onEnter	void	Search station by name on enter to input field
2	initHome	Void	Initialize home screen
3	requestToReturnHome	void	Process request to return home screen
4	addStationHome	Void	Add stations to home screen

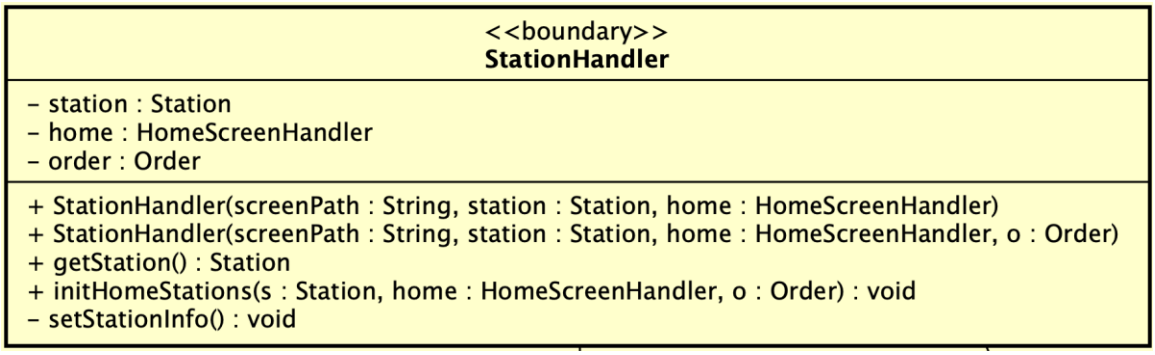
Parameter

- stage – stage of screen
- ScreenPath – path to fxml screen
- SearchString – input string use to search
- Items – list of stations

Method: none

State: none

4.4.3.19 Class "StationHandler"



Attribute

#	Name	Data type	Default value	Description
1	station	Station	null	Current station
2	order	Order	null	Current order if any
3	home	HomeScreenHandler	Null	Home screen

Operation

#	Name	Return type	Description (purpose)

1	getStation	Station	Get current station
2	initHomeStations	void	initialize stations in home screen

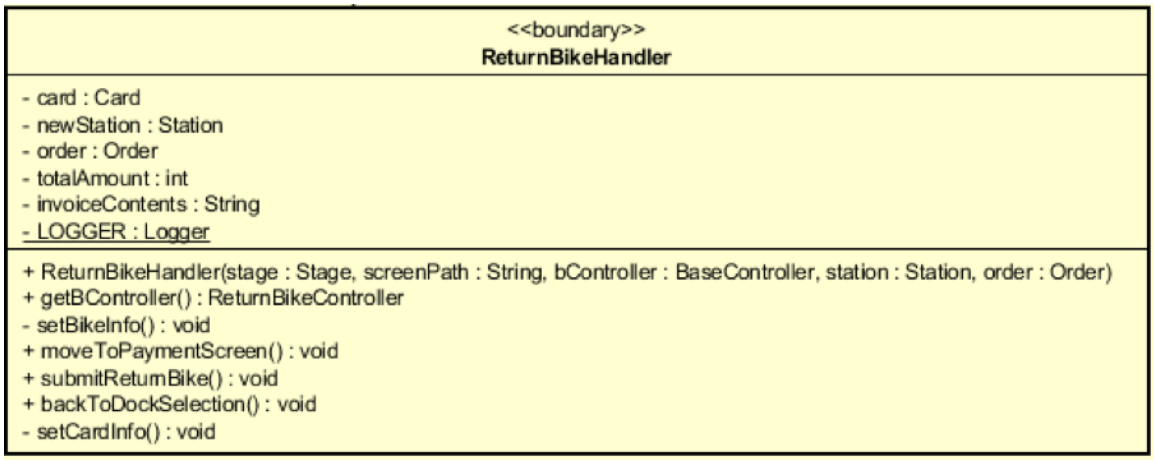
Parameter:

- stage – stage of screen
- ScreenPath – path to fxml screen
- Home – Home Screen handler
- o – current order
- s – current station

Method: none

State: none

4.4.3.20 Class "ReturnBikeHandler"



Attribute

#	Name	Data type	Default value	Description
1	card	Card	NULL	The card used for returning bike
2	newStation	Station	NULL	The station which the bike returns to
3	order	Order	NULL	The order of returning bike
4	totalAmount	String	NULL	The total amount that user has to pay when reting bike

5	invoiceContents	String	NULL	Invoice Contents, could be pay or refund
6	Logger	LOGGER	NULL	For logging purpose

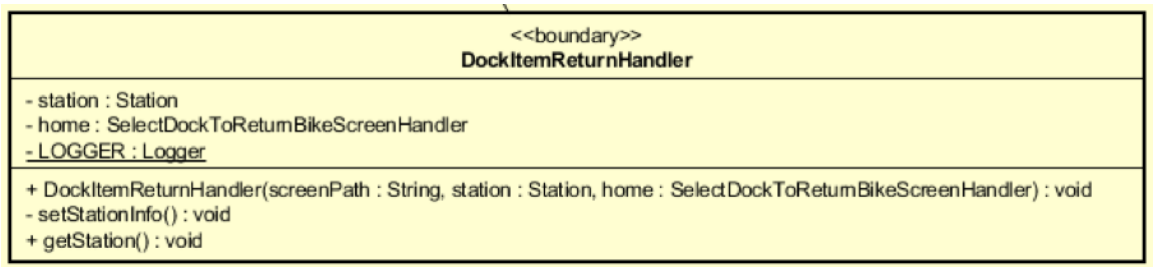
Operation

#	Name	Return type	Description (purpose)
1	ReturnBikeHandler		Constructor
2	GetBController	ReturnBikeController	Set the controller for this screen
3	setBikeInfo	void	Set the bike info for displaying
4	moveToPaymentScreen	void	Move to payment screen
5	submitToReturnBike	void	handle the process when user click submit to return bike
6	backToDockSelection	Bike	Handle the process when user want to choose another station to return bike
7	setCardInfo	Void	Set the card info for displaying in the screen

Method: none

State: none

4.4.3.21 Class "DockItemReturnHandler"



Attribute

#	Name	Data type	Default value	Description
1	station	Station	NULL	The station we want to display
2	home	SelectDockToReturnBike ScreenHandler	NULL	The root screen which use this screen to display station
3	Logger	LOGGER	NULL	For logging purpose

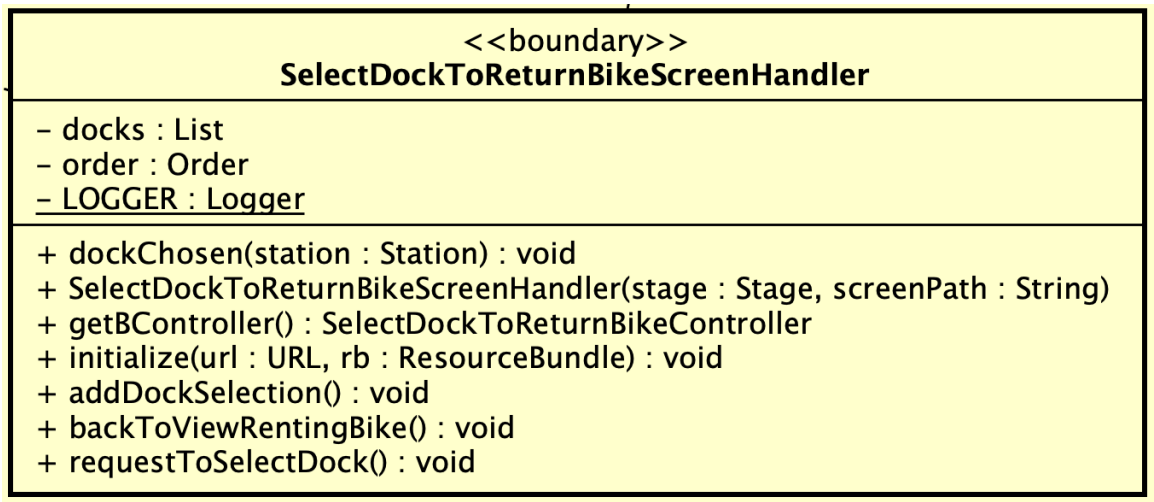
Operation

#	Name	Return type	Description (purpose)
1	DockItemReturnHandl er		Constructor
2	setStationInfo	Void	Set the info we want to display
3	getStation	Void	Get the current Station

Method: none

State: none

4.4.3.22 Class " SelectDockToReturnBikeScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	docks	List[Station]	NULL	A list of stations that user can return bike to

2	order	Order	NULL	The current order, while renting bike
3	Logger	LOGGER	NULL	For logging purpose

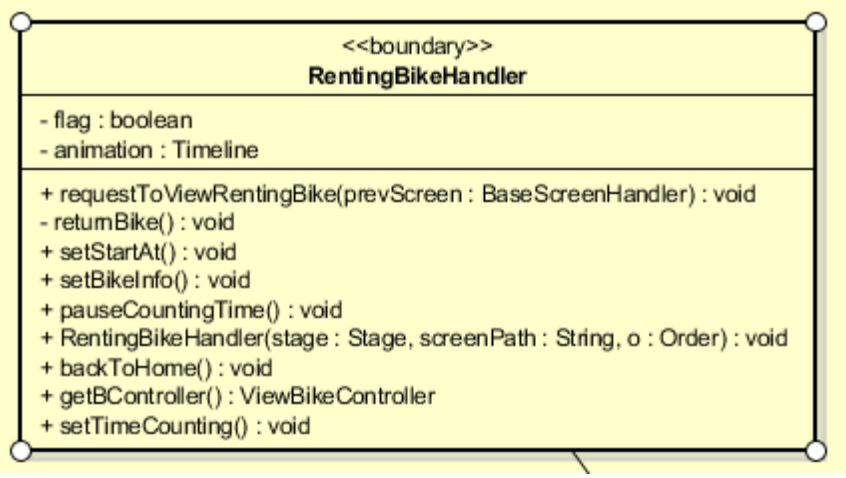
Operation

#	Name	Return type	Description (purpose)
1	dockChosen	Void	Move to the return bike screen when a station is chosen
2	SelectDockToReturnBikeScreenHandler		Constructor
3	getBController	SelectDockToReturnBikeController	Set the controller for this screen
4	initialize	void	initialize the content of the screen
5	addDockSeletion	void	Display list of station for user to choose
6	backToViewRentingBike	Void	Back to previous screen, which is view renting bike screen
7	requestToSelectDock	Void	Request to move to this screen

Method: none

State: none

4.4.3.23 Class “RentingBikeHandler”



Attribute

#	Name	Data type	Default value	Description
1	flag	Boolean	NULL	For stop renting bike purpose
2	Animation	Timeline	NULL	For display time purpose

Operation

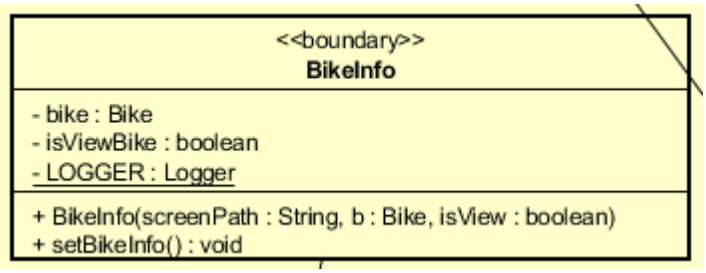
#	Name	Return type	Description (purpose)
1	requestToViewRentingBike	Void	Request to move to this screen
2	returnBike	Void	Request to move to select dock to return bike screen
3	setBikeInfo	void	Set the bike info for displaying
4	setStartAt	void	Set start time for time displaying
5	pauseCountingTime	void	For pause and continue renting time
6	RentingBikeHandler		Contructor

7	backToHome	Void	Back to home screen
8	getBController	ViewBikeController	Set controller for this screen
9	setTimeCoungting	Void	Set the counting time on the screen

Method: none

State: none

4.4.3.24 Class "BikeInfo"



Attribute

#	Name	Data type	Default value	Description
1	bike	Bike	NULL	The bike we want to display
2	isViewBike	Boolean	NULL	To check if this bike is the viewing bike
3	Logger	LOGGER	NULL	For logging purpose

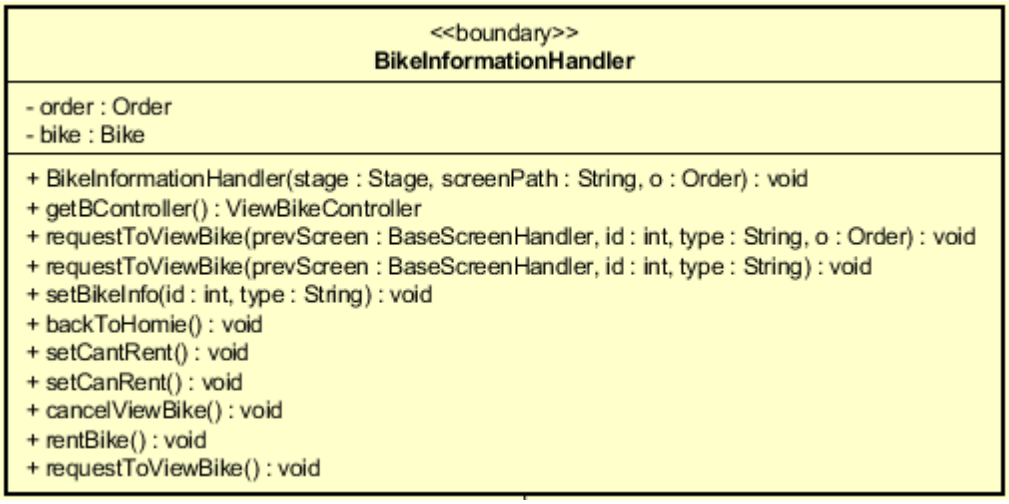
Operation

#	Name	Return type	Description (purpose)
1	BikeInfo		Constructor
2	setBikeInfo	Void	Set the info we want to display

Method: none

State: none

4.4.3.25 Class " BikeInformationHandler"



Attribute

#	Name	Data type	Default value	Description
1	order	Order	NULL	The current order, to check if the customer is using any rented bike
2	bike	Bike	NULL	The bike we want to display

Operation

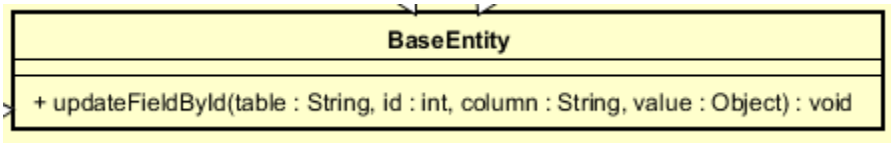
#	Name	Return type	Description (purpose)
1	BikeInformationHandler		Constructor
2	requestToViewBike	Void	Request to view this screen
3	getBController	ViewBikeController	Set the controller for this screen
4	setBikeInfo	void	Set the bike information to display

5	backToHomie	void	Back to home screen
6	setCantRent	Void	User cannot perform the action of renting bike while he/she is using a rented bike
7	setCanRent	Void	User can perform the action of renting bike
8	cancelViewBike	Void	Back to previous screen
9	rentBike	Void	Move to rent bike screen

Method: none

State: none

4.4.3.26 Class "BaseEntitty"



Attribute:

None

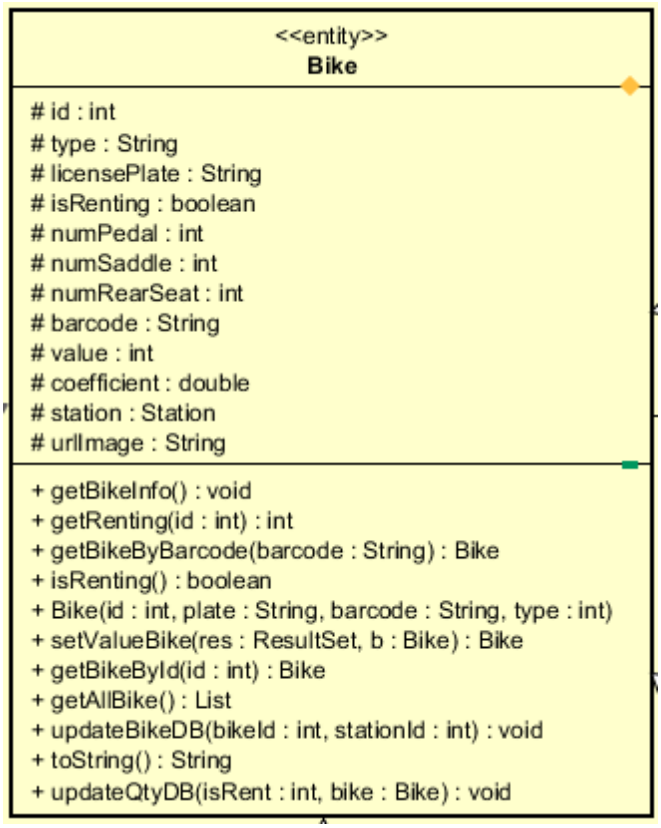
Operation:

#	Name	Return type	Description (purpose)
1	updateFieldById	void	Update column any table in database with primary key id by value

Method: None

State: None

4.4.3.27 Class "Bike"



Attribute

#	Name	Data type	Default value	Description
1	numSaddle	int	NULL	Number saddle of the bike
2	numPedal	int	NULL	Number pedal of the bike
3	numRearSeat	Int	NULL	Number rear seat of the bike
4	licensePlate	String	NULL	Represent license plate of the bike
5	value	int	NULL	Represent value of the bike
6	barcode	String	NULL	Represent barcode of the bike

7	type	String	NULL	Represent type of the bike
8	station	Station	NULL	Represent bike in which station
9	Coefficient	double	NULL	The coefficient uses when calculate deposit
10	urlImage	String	NULL	The path to the bike image
11	isRenting	boolean	NULL	Check if the bike is being rented or not
12	Id	Int	NULL	The id of the bike in database

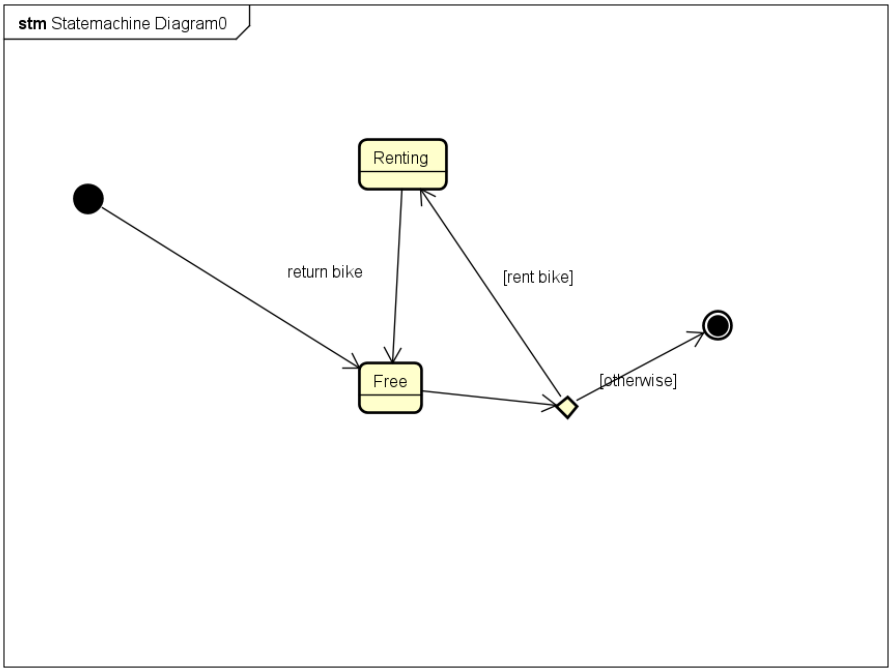
Operation

#	Name	Return type	Description (purpose)
1	getBikeInfo	void	Get bike information for display
2	getRenting	int	Check the bike with corresponding id is being rented or not
3	Bike	void	Constructor
4	getter	void	Get all attribute in acronym
5	setter	void	Set value for each attribute in acronym
6	getBikeByBarcode	Bike	Get a bike using its barcode
7	isRenting	Boolean	Check if a bike is currently being rented
8	setValueBikes	Bike	Set a bike info base on a result set
9	getBikeByID	Bike	Get a bike by using its id
10	getAllBike	List[bike]	Get all the bike in database

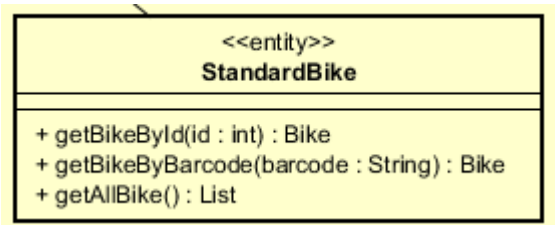
11	updateBikeDB	Void	Change the station of a bike in database
12	toString	String	Convert a bike object to string
13	updateQtyDB	Void	update station and bike when a bike is returned/rented

Method: none

State:



4.4.3.28 Class "Standard Bike"



Attribute

Operation: Inherit from bike entity and override

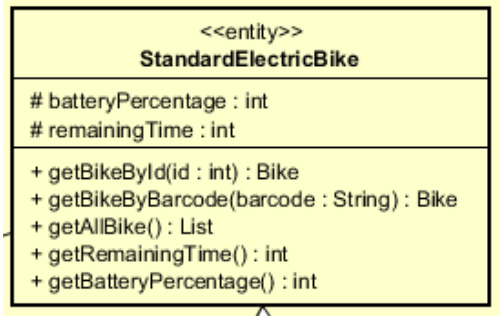
#	Name	Return type	Description (purpose)

1	getBikeById	Bike	Get a standard bike by its id
2	getBikeByBarcode	Bike	Get a standard bike by its barcode
3	getAllBike	List[Bike]	Get all standard bike

Method: none

State: none

4.4.3.29 Class "Standard Electric Bike"



Attribute

#	Name	Data type	Default value	Description
1	batteryPercentage	int	NULL	The battery percentage of the electric bike
2	remaningTime	int	NULL	The remaining time of the electric bike

Operation:

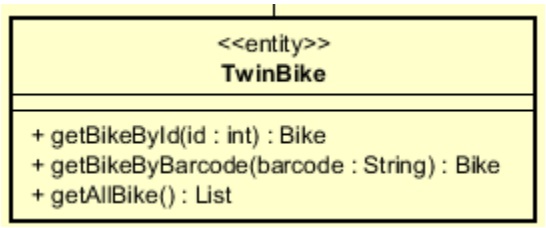
- **Inherit from bike entity and override**

#	Name	Return type	Description (purpose)
1	getBikeById	Bike	Get a standard electric bike by its id
2	getBikeByBarcode	Bike	Get a standard electric bike by its barcode
3	getAllBike	List[Bike]	Get all standard electric bike
4	Getter/setter	Void	Get/set attribute in acronym

Method: none

State: none

4.4.3.30 Class "Twin Bike"



Attribute

Operation:

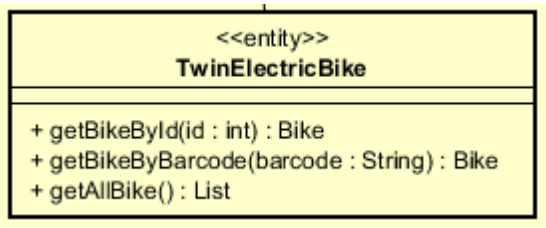
- **Inherit from bike entity and override**

#	Name	Return type	Description (purpose)
1	getBikeById	Bike	Get a twin bike by its id
2	getBikeByBarcode	Bike	Get a twin bike by its barcode
3	getAllBike	List[Bike]	Get all twin bike

Method: none

State: none

4.4.3.31 Class "Twin Electric Bike"



Attribute

Operation:

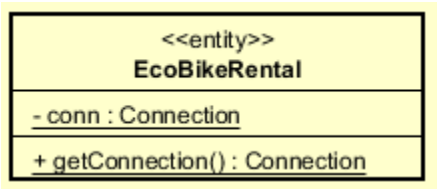
- **Inherit from standard electric bike entity and override**

#	Name	Return type	Description (purpose)
1	getBikeById	Bike	Get a twin electric bike by its id
2	getBikeByBarcode	Bike	Get a twin electric bike by its barcode
3	getAllBike	List[Bike]	Get all twin electric bike

Method: none

State: none

4.4.3.32 Class "EcoBikeRental"



Attribute

#	Name	Data type	Default value	Description
1	Conn	Connection	NULL	Connection to the remote database

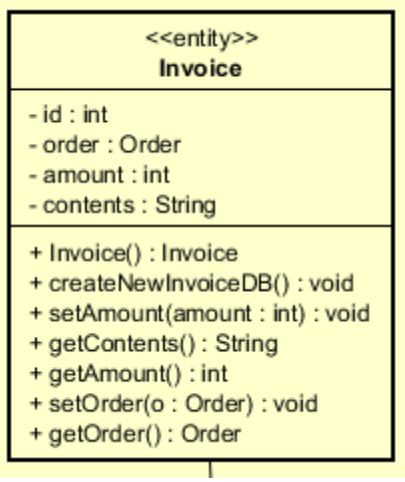
Operation

#	Name	Return type	Description (purpose)
1	getConnetion	Connection	Connect to the remote database

Method: None

State: None

4.4.3.33 Class "Invoice"



Attribute

#	Name	Data type	Default value	Description
1	Id	Int	NULL	The id of the invoice
2	order	Order	NULL	The order that was used for this invoice
3	Amount	Int	NULL	The amount the user has to pay/is refunded to.
4	content	String	NULL	The details of the invoice

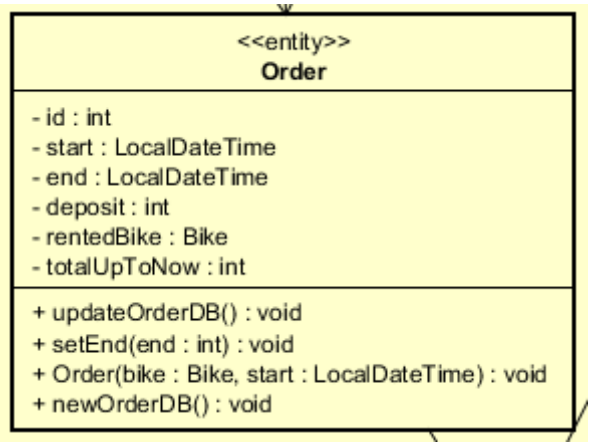
Operation

#	Name	Return type	Description (purpose)
1	Invoice		Constructor
2	newInvoiceDB	void	Save the invoice to the database, after successful transaction
3	Getter/setter	void	Get/set attribute in acronym

Method: None

State: None

4.4.3.34 Class "Order"



Attribute

#	Name	Data type	Default value	Description
1	Id	Int	NULL	The id of the order
2	start	LocalDateTime	NULL	The time when user starts renting bike
3	End	LocalDateTime	NULL	The time when user returns bike
4	Deposit	Int	NULL	The amount of deposit when rent bike
5	rentedBike	Bike	NULL	The bike which is being rented
6	totalUpToNow	Int	NULL	Total renting amount (not include deposit)

Operation

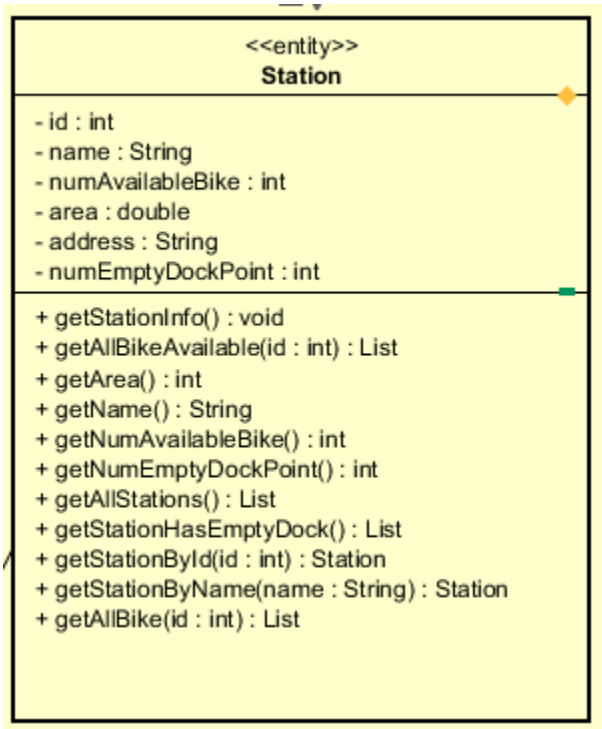
#	Name	Return type	Description (purpose)
1	Order		Constructor
2	newOrderDB	void	Insert new order to database, after renting bike successfully
3	updateOrderDB	void	Update the order in the database, after returning bike successfully

4	Getter/setter	void	Get/set attribute in acronym

Method: None

State: None

4.4.3.35 Class "Station"



Attribute

#	Name	Data type	Default value	Description
1	Id	Int	NULL	The id of the station
2	name	String	NULL	The name of the station
3	numAvailableBike	Int	NULL	Number of available bike in the station
4	Area	Double	NULL	Area of the station
5	Address	String	NULL	Address of the station
6	numEmptyDockPoint	Int	NULL	Number of empty dock point in station

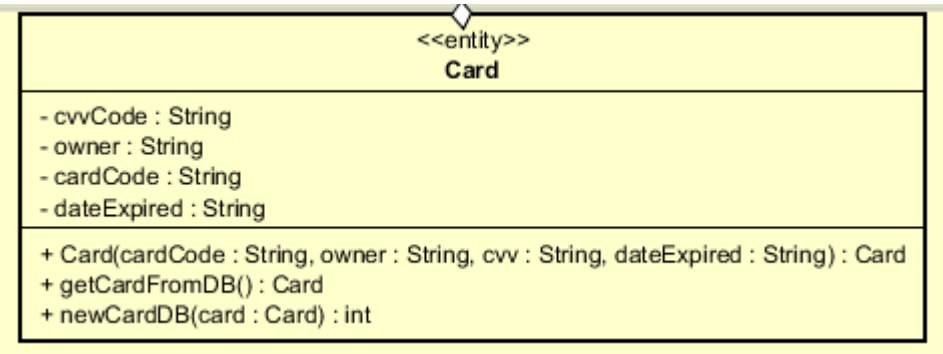
Operation

#	Name	Return type	Description (purpose)
1	Station		Constructor
2	getAllBikeAvaiable	List[Bike]	Get all bike in a station with given station id, and the bike is currently not being rented
3	getAllStation	List[Station]	Get all the station in database
4	Getter/setter	void	Get/set attribute in acronym
5	getStationById	Station	Get a station with its id
6	getStationByName	Station	Get a station by its name
7	getAllBike	List[Bike]	Get all bike in a station with given station id
8	getStationHasEmptyDock	List[Station]	Get all stations which have empty dock for returning bike

Method: None

State: None

4.4.3.36 Class "Card"



Attribute

#	Name	Data type	Default value	Description
1	owner	String	NULL	Name of the owner of the credit card

2	cardCode	String	NULL	code of the credit card
3	cvvCode	String	NULL	Security code of the credit card
4	dateExpired	String	NULL	The expiration date of the card, in form MMYY

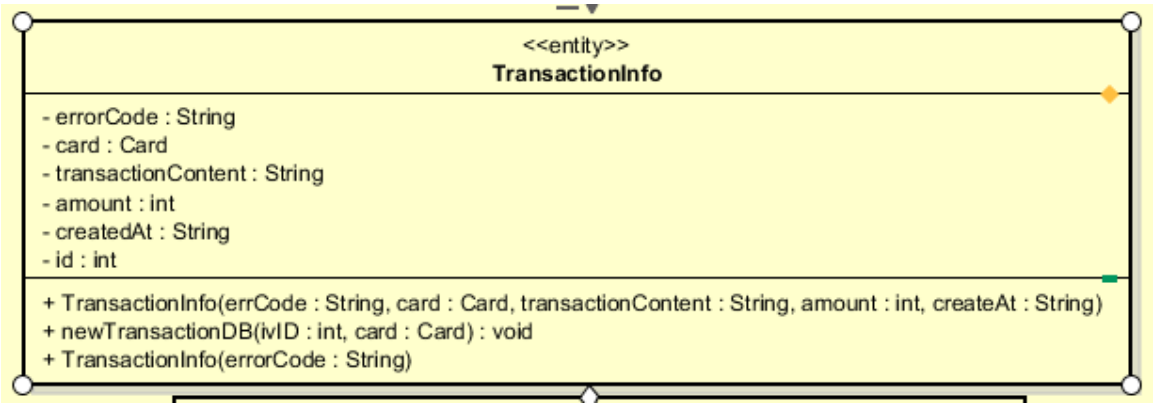
Operation

#	Name	Return type	Description (purpose)
1	Card		Constructor
2	newCardDB	int	Save the card in the database if this is a new card. Get the id of the corresponding card in database.
3	getCardFromDB	Card	Get the card from database
4	Getter/setter	void	Get/set attribute in acronym

Method: None

State: None

4.4.3.37 Class "TransactionInfo"



Attribute

#	Name	Data type	Default value	Description
1	card	Card	NULL	Represent the card used for payment

2	amount	int	NULL	Represent total amount of the transaction
3	Transaction Content	String	NULL	Represent content of transaction
4	createdAt	String	NULL	The time when the transaction was created
5	Id	Int	NULL	The id of the transaction
6	errorCode	String	NULL	The error code represents the state of the transaction

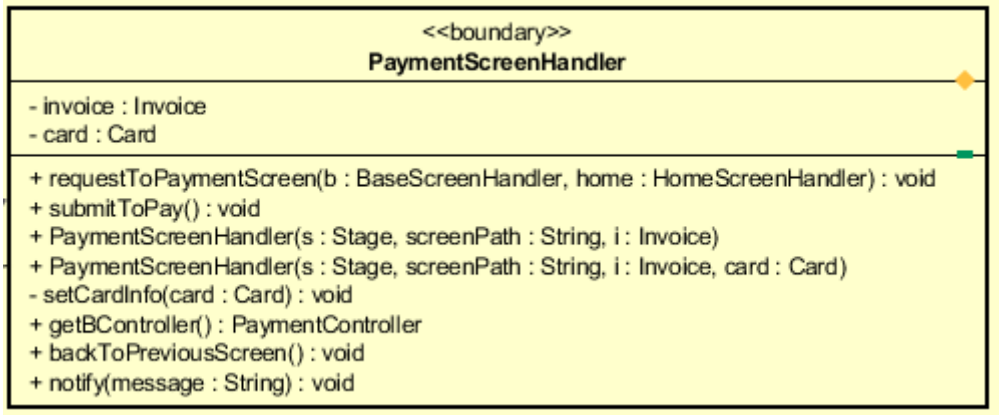
Operation

#	Name	Return type	Description (purpose)
1	TransactionInfo		Constructor
2	newTransaction DB	void	Save the successful transaction to database, and the card used for the transaction
3	Getter/setter	void	Get/set attribute in acronym

Method: None

State: None

4.4.3.38 Class "PaymentScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	invoice	Invoice	NULL	The invoice created for this payment
2	card	Card	NULL	The card used for this payment

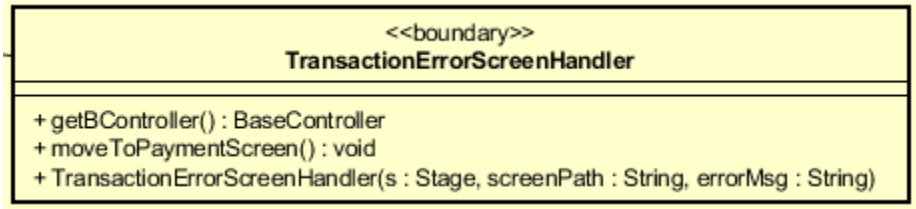
Operation

#	Name	Return type	Description (purpose)
1	requestToPaymentScreen	Void	Request to move to this screen
2	submiToPay	Void	Handle the process when user click submit button
3	PaymentScreenHandler		constructor
4	setCardInfo	void	Set the card info to display on this screen
5	getBController	PaymentController	Set the controller for this screen
6	backToPreviousScreen	Void	Back to previous screen
7	notify	Void	Display error on screen

Method: none

State: none

4.4.3.39 Class "TransactionErrorScreenHandler"



Attribute

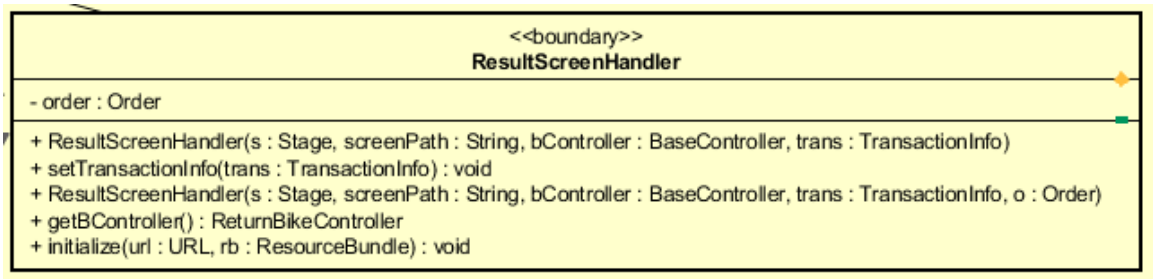
Operation

#	Name	Return type	Description (purpose)
1	getBController	BaseController	Set controller for this screen
2	moveToPaymentScreen	Void	Move back to payment screen
3	TransactionErrorScreenHandler		Constructor

Method: none

State: none

4.4.3.40 Class "ResultScreenHandler"



Attribute

#	Name	Data type	Default value	Description
1	order	Order	NULL	The order used for the rent/return payment process

Operation

#	Name	Return type	Description (purpose)
1	ResultScreenHandler		Constructor

2	setTransactionInfo	Void	Set all the info displayed on the screen
3	getBController	ReturnBikeController	Set the controller for this screen
4	initialize	void	initialize the content of the screen

Method: none

State: none

5 Design Considerations

5.1 Goals and Guidelines

Goals:

- Bring a good looking and good experience for users
- The response time for the system is 1 second at normal and 2 seconds during a peak load

Guidelines

- Observe java convention in coding, OOP principles
- Avoid hash code
- Explain code, write java doc for maintenance

5.2 Architectural Strategies

Our design decisions focus on reusing components, unified system following

+ Programing Language: java

+ Database: MySQL

+ Unified on error detection and recovery

We always toward save memory and spaces, also speed up response time and nice looking. In the future, we plan to extend software: have site for admin to add, delete bike, statistics, business strategies. These targets make us concentrate totally on architectural design.

5.3 Coupling and Cohesion

In our software design, we detect that there are some components have Control Coupling and Communicational Cohesion problems.

We are trying our best to resolve these problems, decrease Coupling level and increase Cohesion level. However, due to lack of time, we might not be able to fix this before announced deadline.

We will resolve these as soon as possible.

5.4 Design Principles

We design simple classes that means a class should have only one job, one responsibility. Object or entities are open for extension but close for modification. We also use interfaces, abstract classes. We put all class with same properties into one

package to manage easily. Therefore, we can reuse source code, adapt any changing requirements.

5.5 Design Patterns

We don’t use any design patterns.