HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

EcoBikeRental

Subject: 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 anyone who is interested in designing software.

1.2 Scope

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

1.2.2 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.

1.2.3 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 a high-level diagram to help you understand the core of our design:

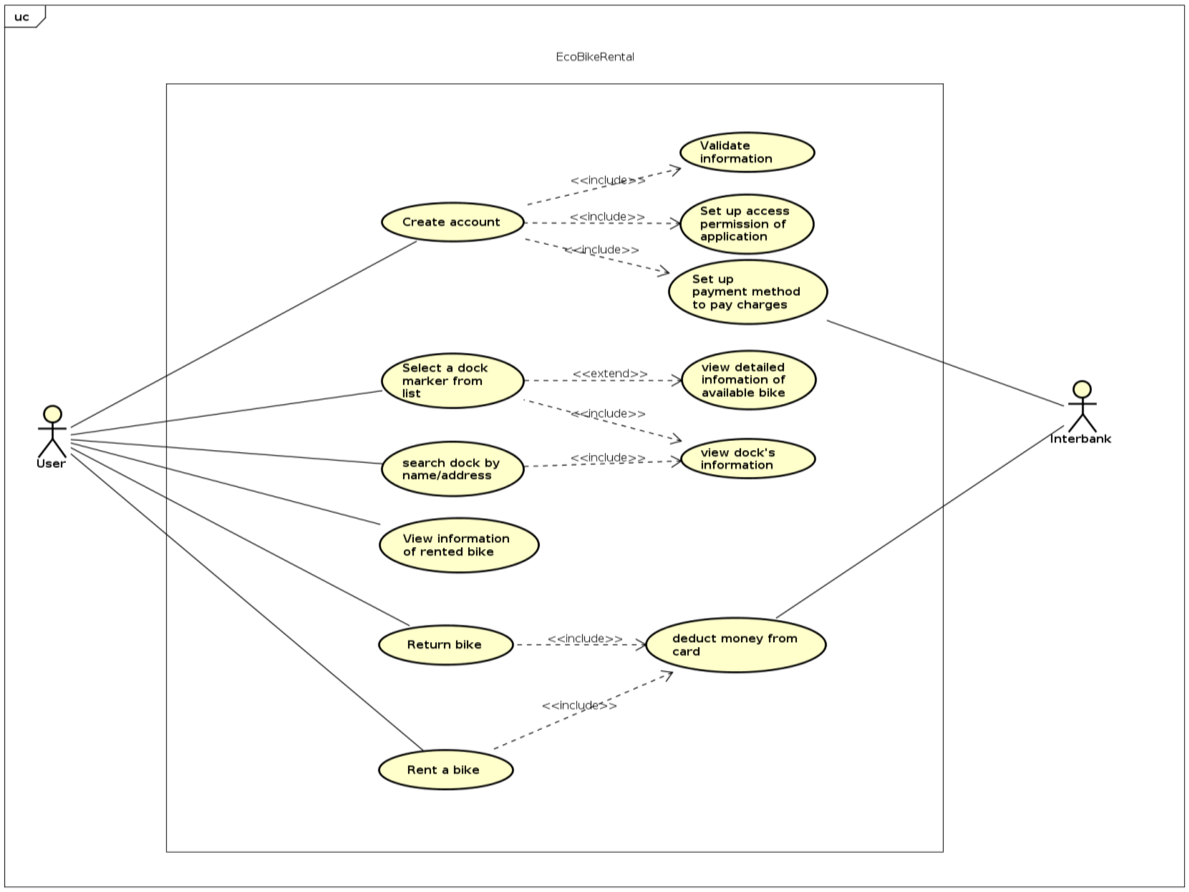


Fig 1. General Use Case Diagram

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

- No risks to be discussed, yet.

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

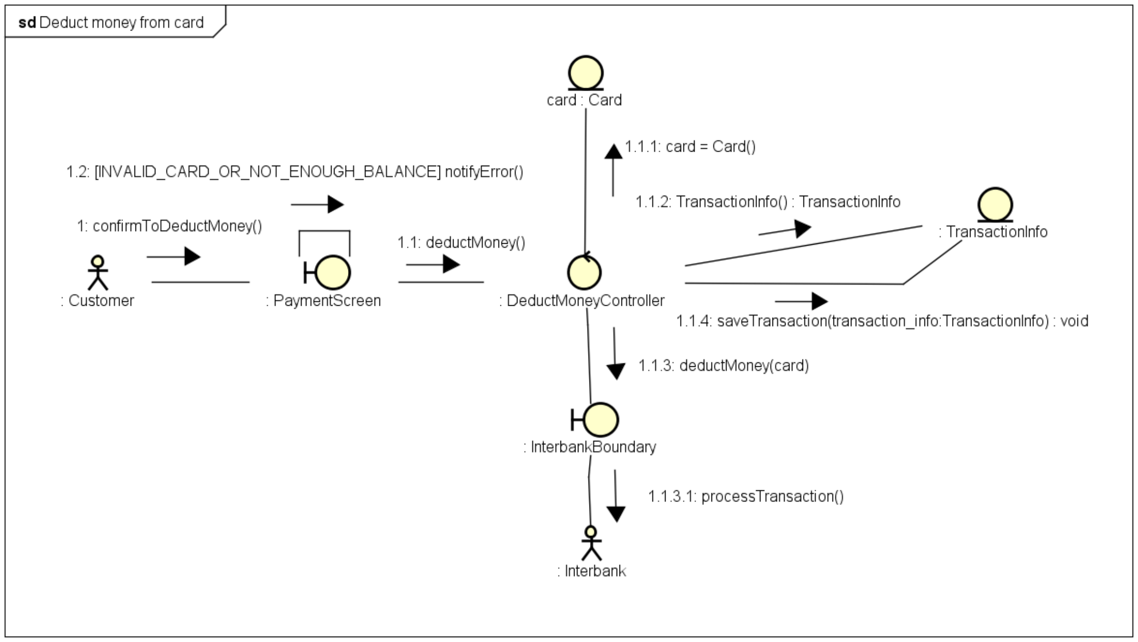

sd Deduct money from card
card : Card
1.1.1: card = Card()
1.2: [INVALID_CARD_OR_NOT_ENOUGH_BALANCE] notifyError()
1.1.2: TransactionInfo() : TransactionInfo
: TransactionInfo
1: confirmToDeductMoney()
1.1: deductMoney()
: Customer
: PaymentScreen
: DeductMoneyController
1.1.4: saveTransaction(transaction_info:TransactionInfo) : void
1.1.3: deductMoney(card)
: InterbankBoundary
1.1.3.1: processTransaction()
: Interbank

Fig 2. Communication Diagram for Deduct money from card

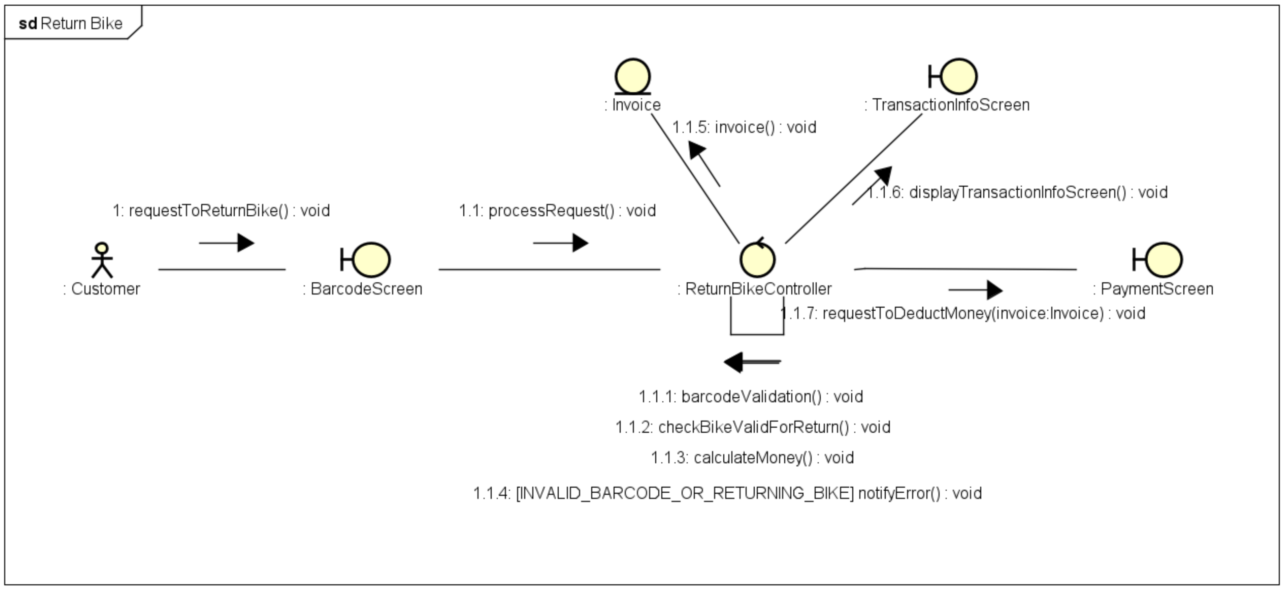

sd Return Bike
: Invoice
: TransactionInfoScreen
1.1.5: invoice() : void
1.1.6: displayTransactionInfoScreen() : void
1: requestToReturnBike() : void
1.1: processRequest() : void
: Customer
: BarcodeScreen
: ReturnBikeController
: PaymentScreen
1.1.7: requestToDeductMoney(invoice:Invoice) : void
1.1.1: barcodeValidation() : void
1.1.2: checkBikeValidForReturn() : void
1.1.3: calculateMoney() : void
1.1.4: [INVALID_BARCODE_OR_RETURNING_BIKE] notifyError() : void

Fig 3. Communication Diagram for Return Bike

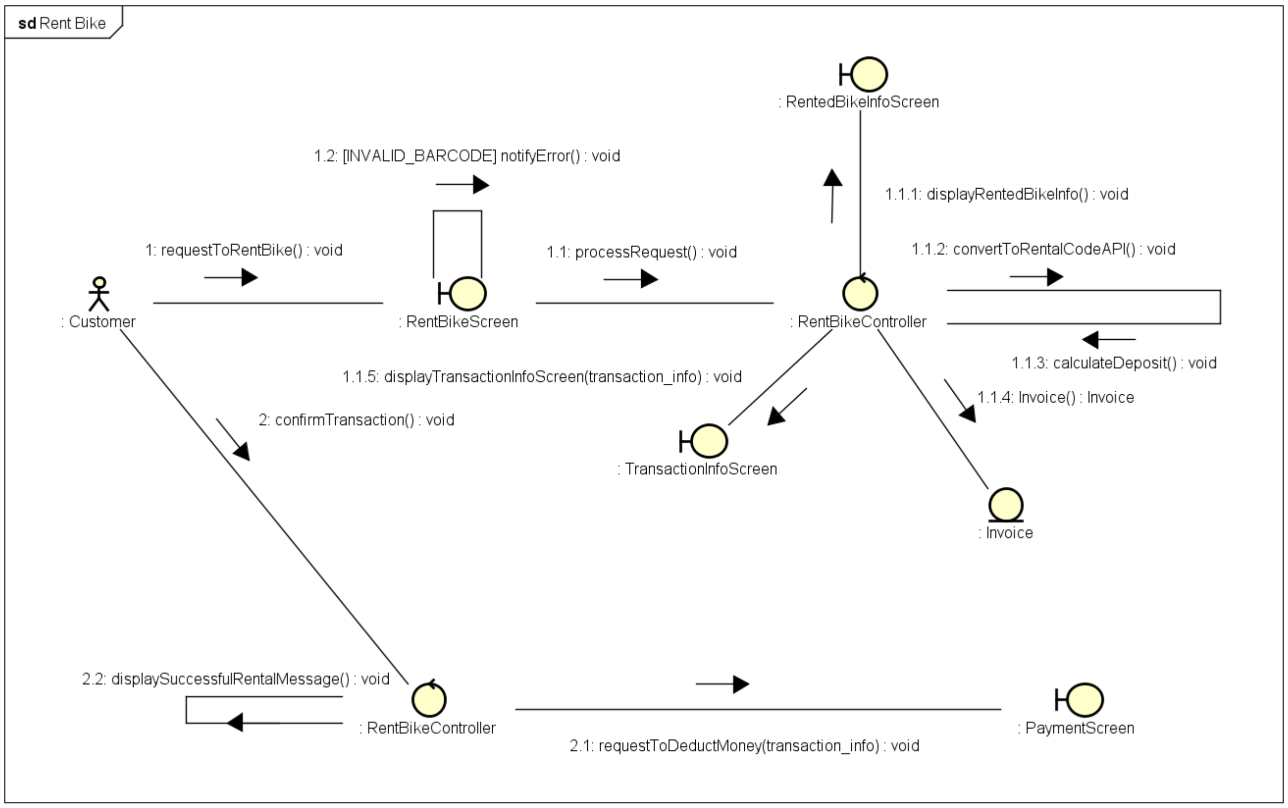

sd Rent Bike
: RentedBikeInfoScreen
1.2: [INVALID_BARCODE] notifyError() : void
1.1.1: displayRentedBikeInfo() : void
1: requestToRentBike() : void
1.1: processRequest() : void
1.1.2: convertToRentalCodeAPI() : void
: Customer
: RentBikeScreen
: RentBikeController
1.1.3: calculateDeposit() : void
1.1.5: displayTransactionInfoScreen(transaction_info) : void
1.1.4: Invoice() : Invoice
2: confirmTransaction() : void
: TransactionInfoScreen
: Invoice
2.2: displaySuccessfulRentalMessage() : void
: RentBikeController
: PaymentScreen
2.1: requestToDeductMoney(transaction_info) : void

Fig 4. Communication Diagram for Rent Bike

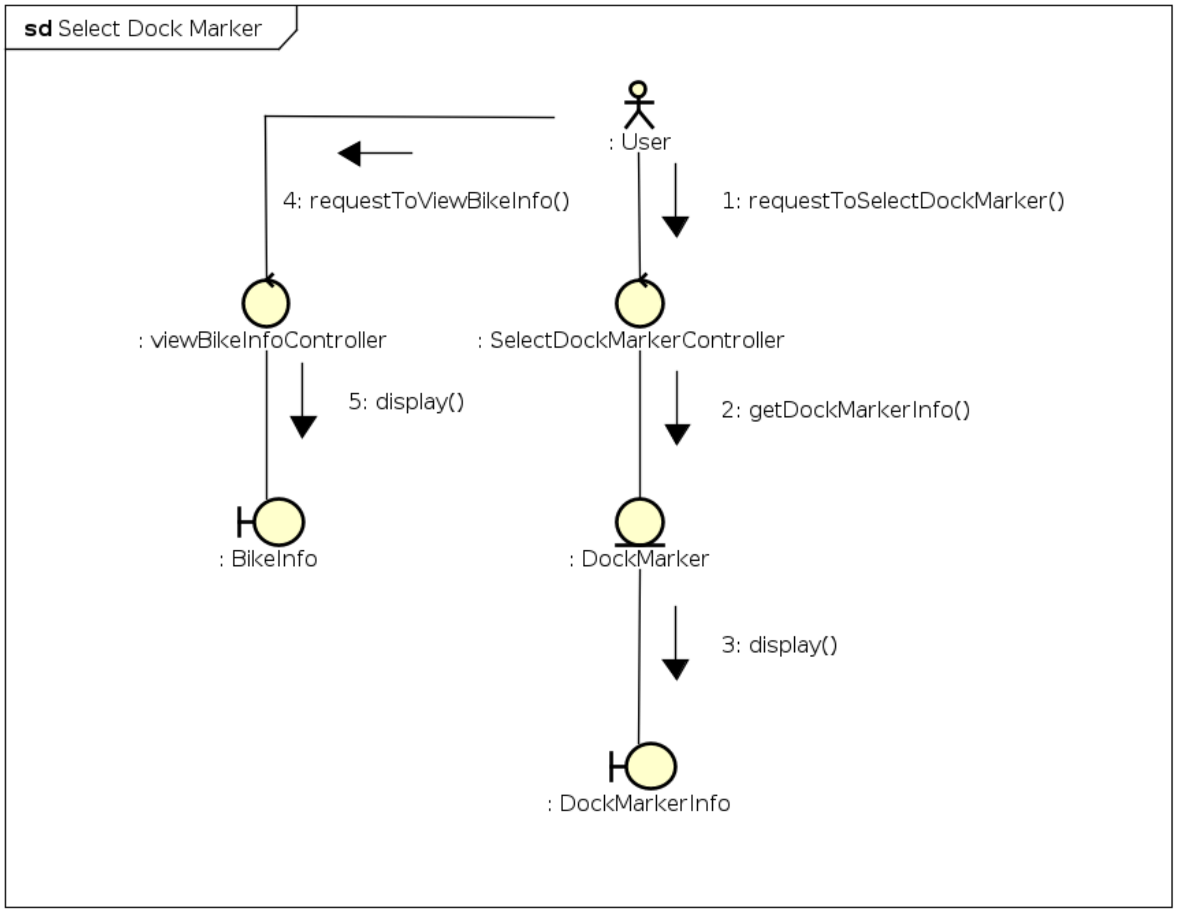


Fig 5. Communication Diagram for Select Dock marker

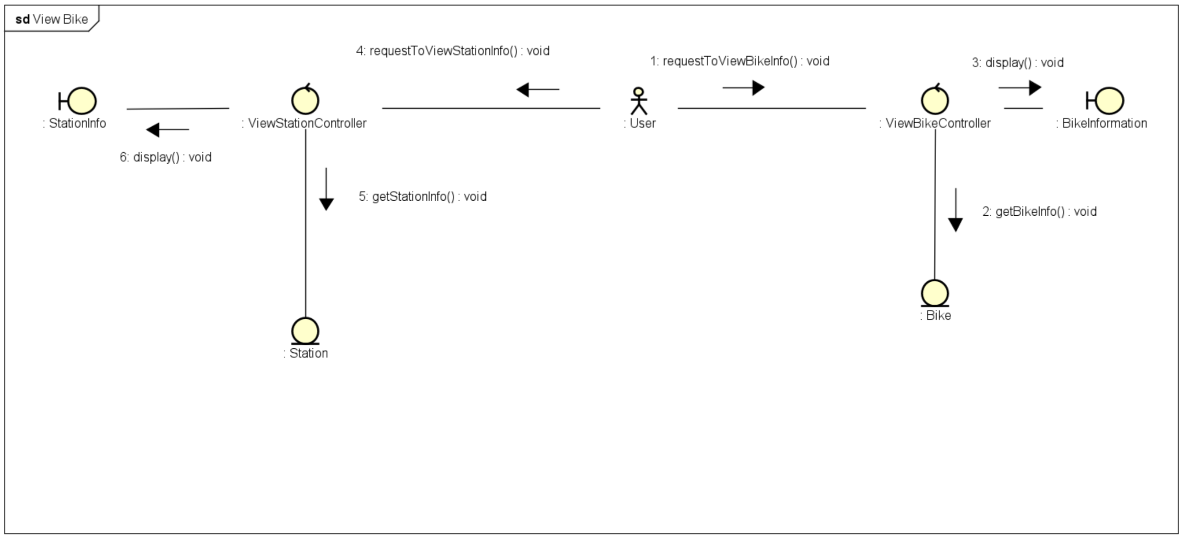


Fig 6. Communication Diagram for View Bike or Station information

3.2.2. Sequence Diagrams

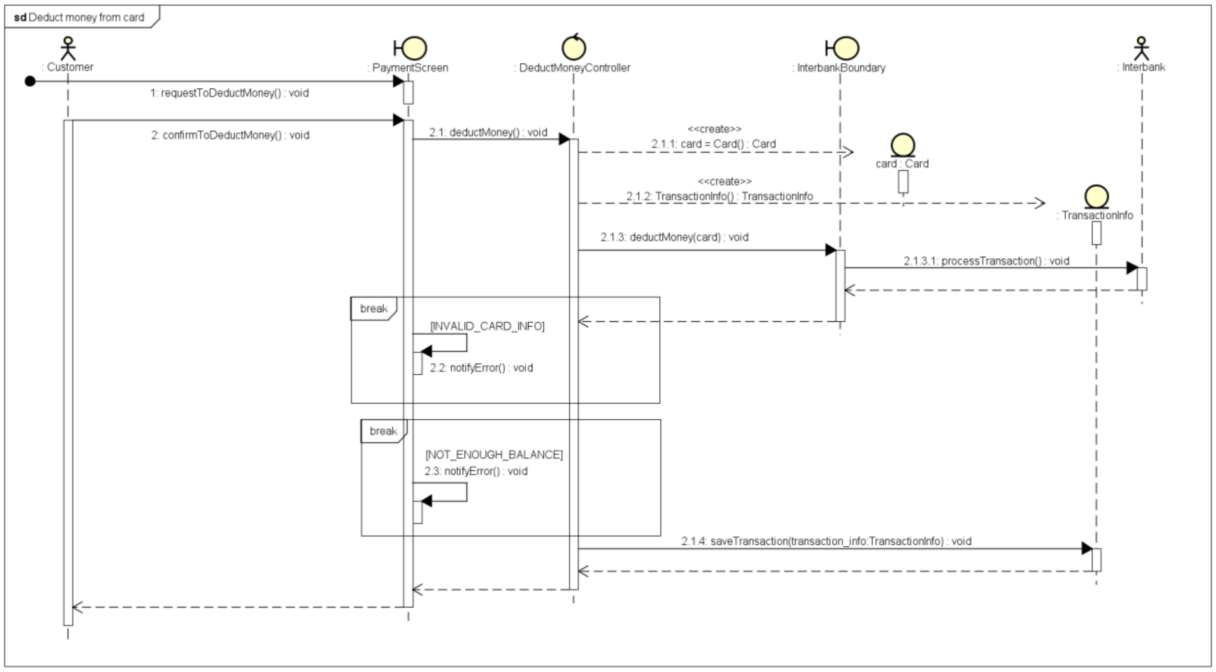

sd Deduct money from card
: Customer
: PaymentScreen
: DeductMoneyController
: InterbankBoundary
: Interbank
1: requestToDeductMoney() : void
2: confirmToDeductMoney() : void
2.1: deductMoney() : void
<<create>>
2.1.1: card = Card() : Card
card : Card
<<create>>
2.1.2: TransactionInfo() : TransactionInfo
: TransactionInfo
2.1.3: deductMoney(card) : void
2.1.3.1: processTransaction() : void
break
[INVALID_CARD_INFO]
2.2: notifyError() : void
break
[NOT_ENOUGH_BALANCE]
2.3: notifyError() : void
2.1.4: saveTransaction(transaction_info:TransactionInfo) : void

Fig 7. Sequence Diagram for Deduct money from card

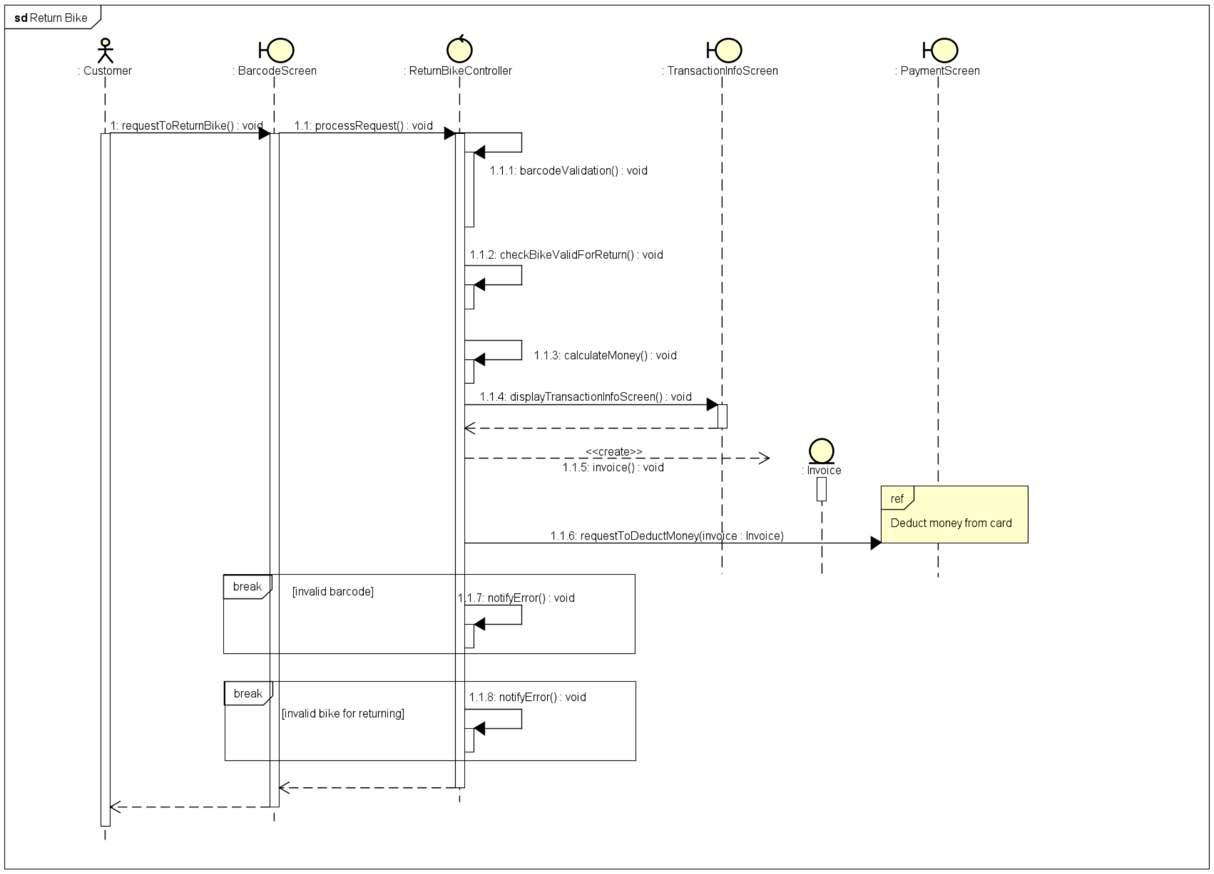

sd Return Bike
: Customer
: BarcodeScreen
: ReturnBikeController
: TransactionInfoScreen
: PaymentScreen
1: requestToReturnBike() : void
1.1: processRequest() : void
1.1.1: barcodeValidation() : void
1.1.2: checkBikeValidForReturn() : void
1.1.3: calculateMoney() : void
1.1.4: displayTransactionInfoScreen() : void
<<create>>
1.1.5: invoice() : void
: Invoice
ref
Deduct money from card
1.1.6: requestToDeductMoney(invoice : Invoice)
break
[invalid barcode]
1.1.7: notifyError() : void
break
[invalid bike for returning]
1.1.8: notifyError() : void

Fig 8. Sequence Diagram for Return bike

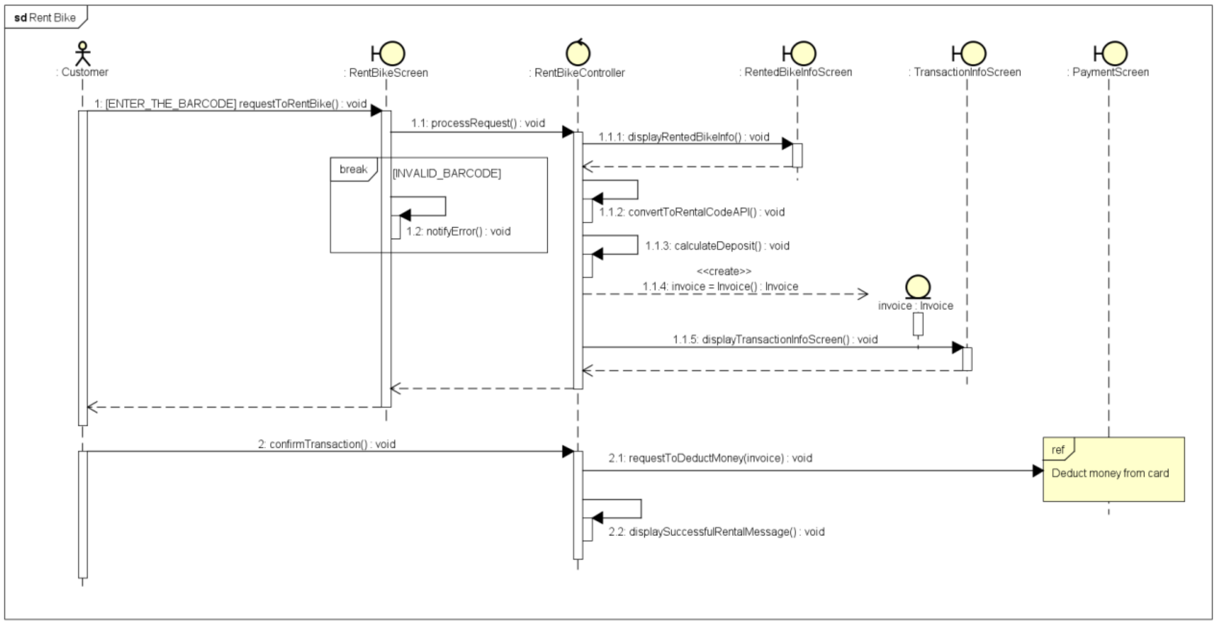

sd Rent Bike
: Customer
: RentBikeScreen
: RentBikeController
: RentedBikeInfoScreen
: TransactionInfoScreen
: PaymentScreen
1: [ENTER_THE_BARCODE] requestToRentBike() : void
1.1: processRequest() : void
1.1.1: displayRentedBikeInfo() : void
break
[INVALID_BARCODE]
1.2: notifyError() : void
1.1.2: convertToRentalCodeAPI() : void
1.1.3: calculateDeposit() : void
<<create>>
1.1.4: invoice = Invoice() : Invoice
invoice : Invoice
1.1.5: displayTransactionInfoScreen() : void
2: confirmTransaction() : void
2.1: requestToDeductMoney(invoice) : void
ref
Deduct money from card
2.2: displaySuccessfulRentalMessage() : void

Fig 9. Sequence Diagram for Rent bike

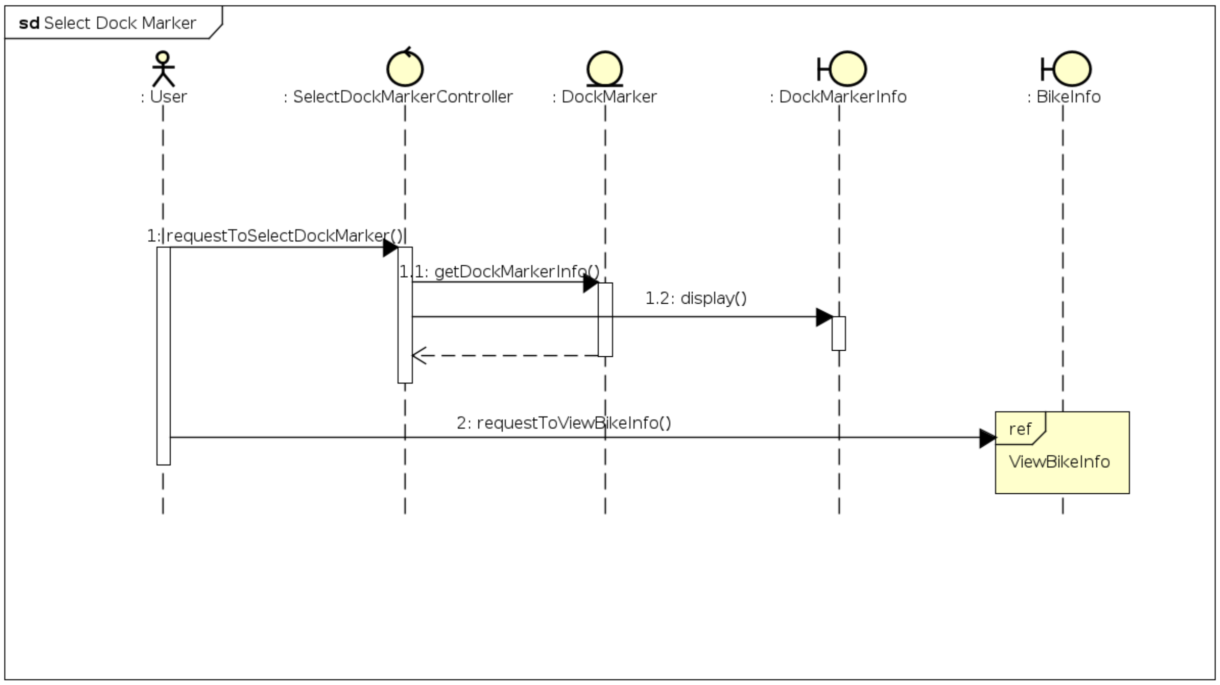


Fig 10. Sequence Diagram for Select Dock marker

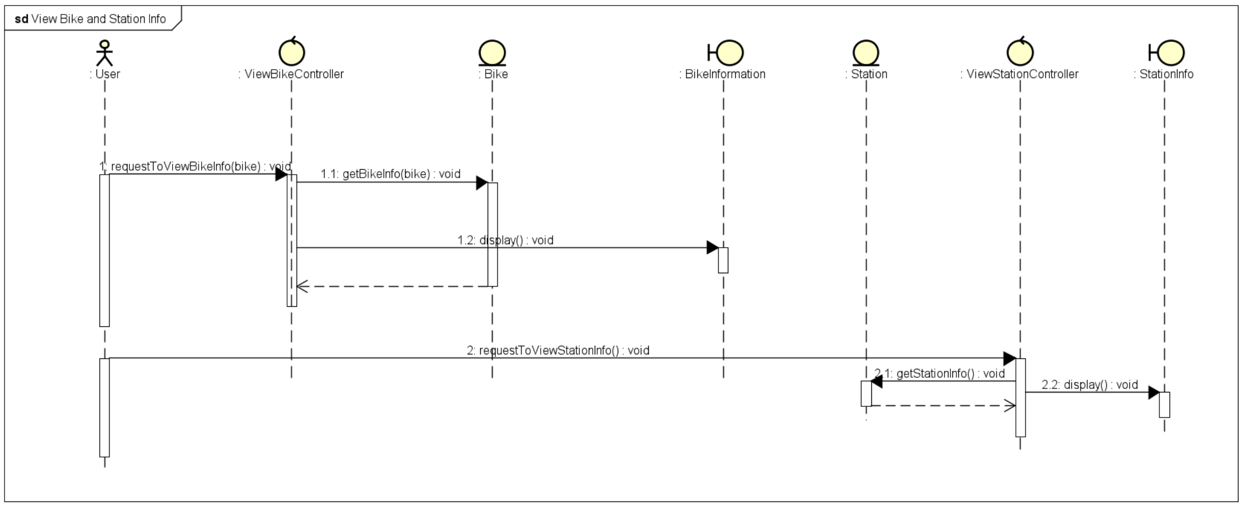

sd View Bike and Station Info
: User
: ViewBikeController
: Bike
: BikeInformation
: Station
: ViewStationController
: StationInfo
1: requestToViewBikeInfo(bike) : void
1.1: getBikeInfo(bike) : void
1.2: display() : void
2: requestToViewStationInfo() : void
2.1: getStationInfo() : void
2.2: display() : void

Fig 11. Sequence Diagram for View Bike or Station information

3.3 Analysis Class Diagrams

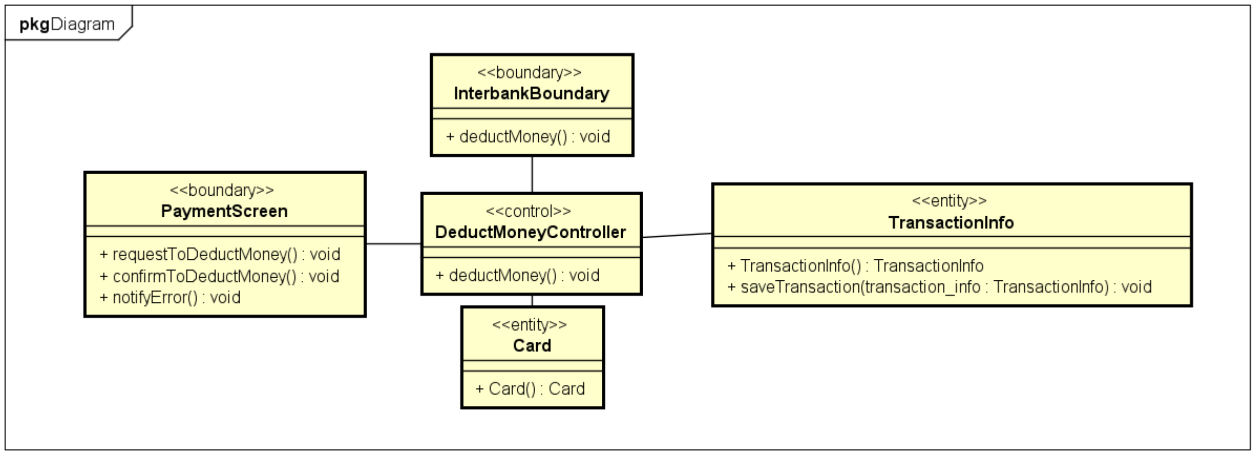


Fig 12. Analysis Class Diagram for Deduct money from card

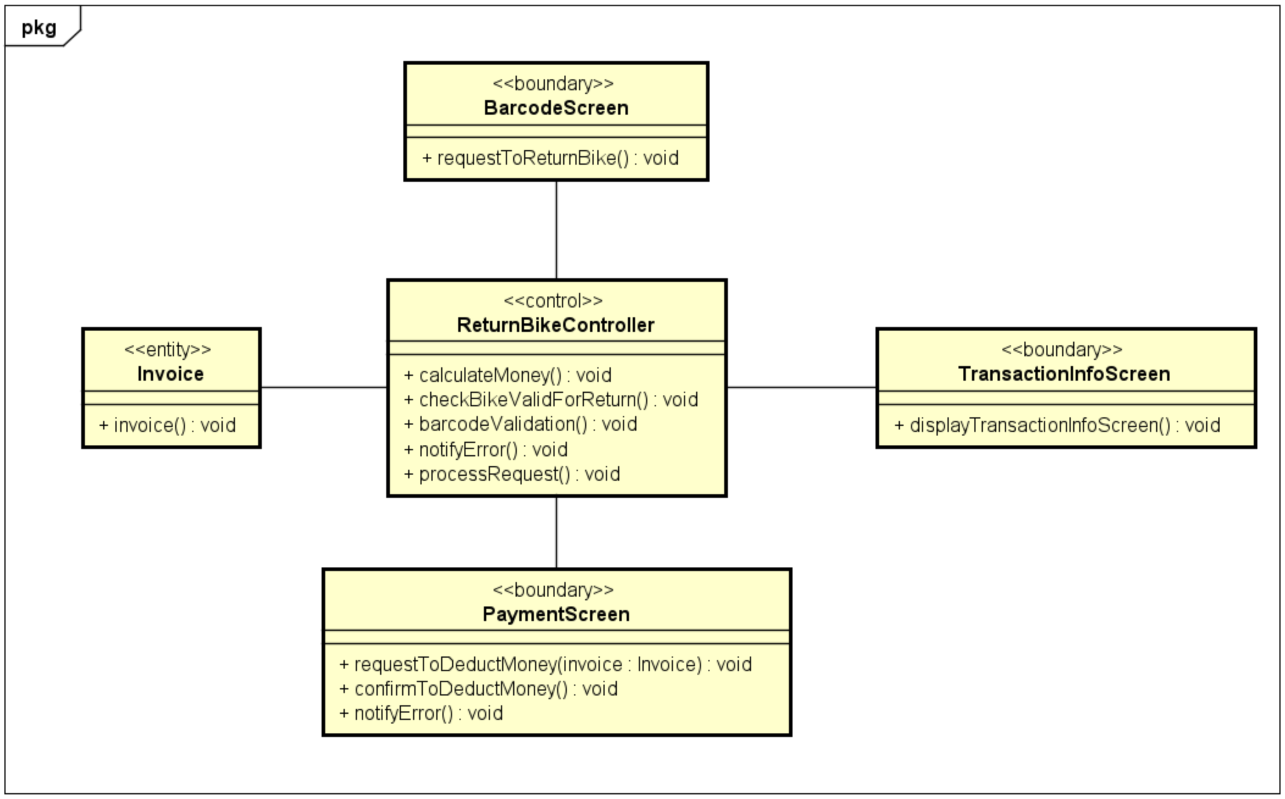

pkg
<<boundary>>
BarcodeScreen
+ requestToReturnBike() : void
<<entity>>
Invoice
+ invoice() : void
<<control>>
ReturnBikeController
+ calculateMoney() : void
+ checkBikeValidForReturn() : void
+ barcodeValidation() : void
+ notifyError() : void
+ processRequest() : void
<<boundary>>
TransactionInfoScreen
+ displayTransactionInfoScreen() : void
<<boundary>>
PaymentScreen
+ requestToDeductMoney(invoice : Invoice) : void
+ confirmToDeductMoney() : void
+ notifyError() : void

Fig 13. Analysis Class Diagram for Return bike

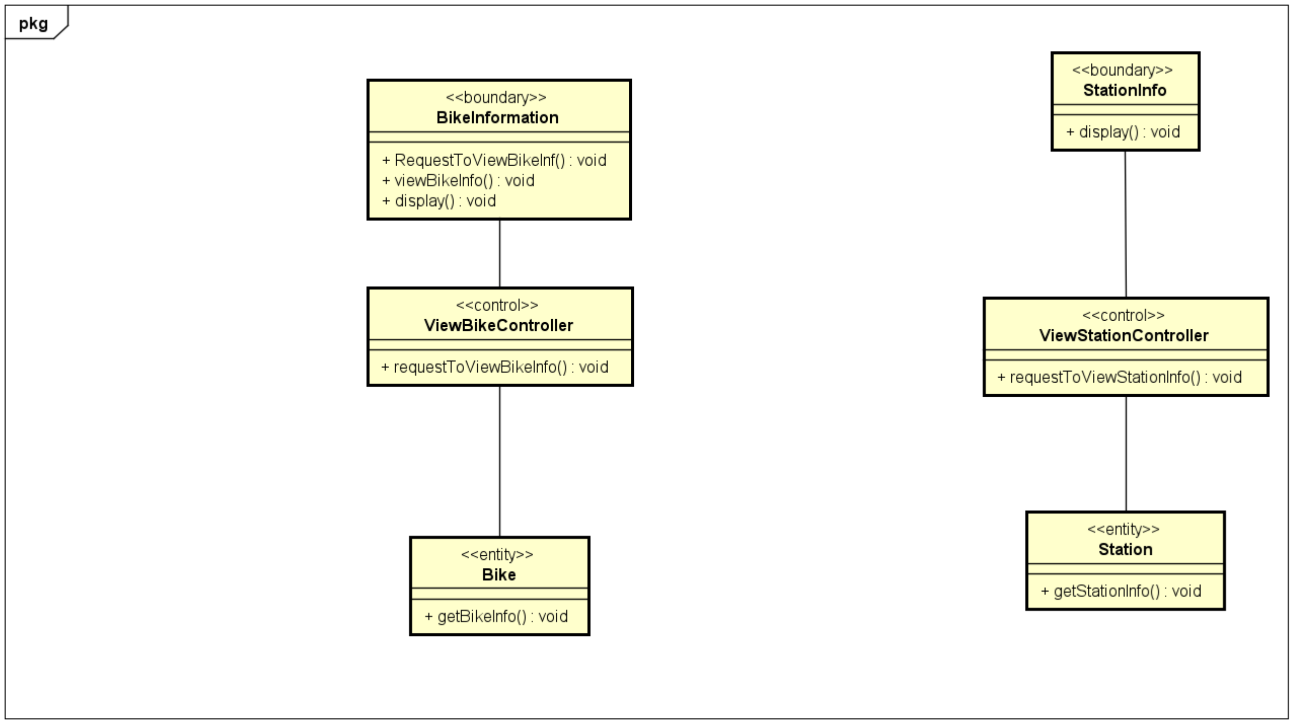


Fig 14. Analysis Class Diagram for View Bike or Station information

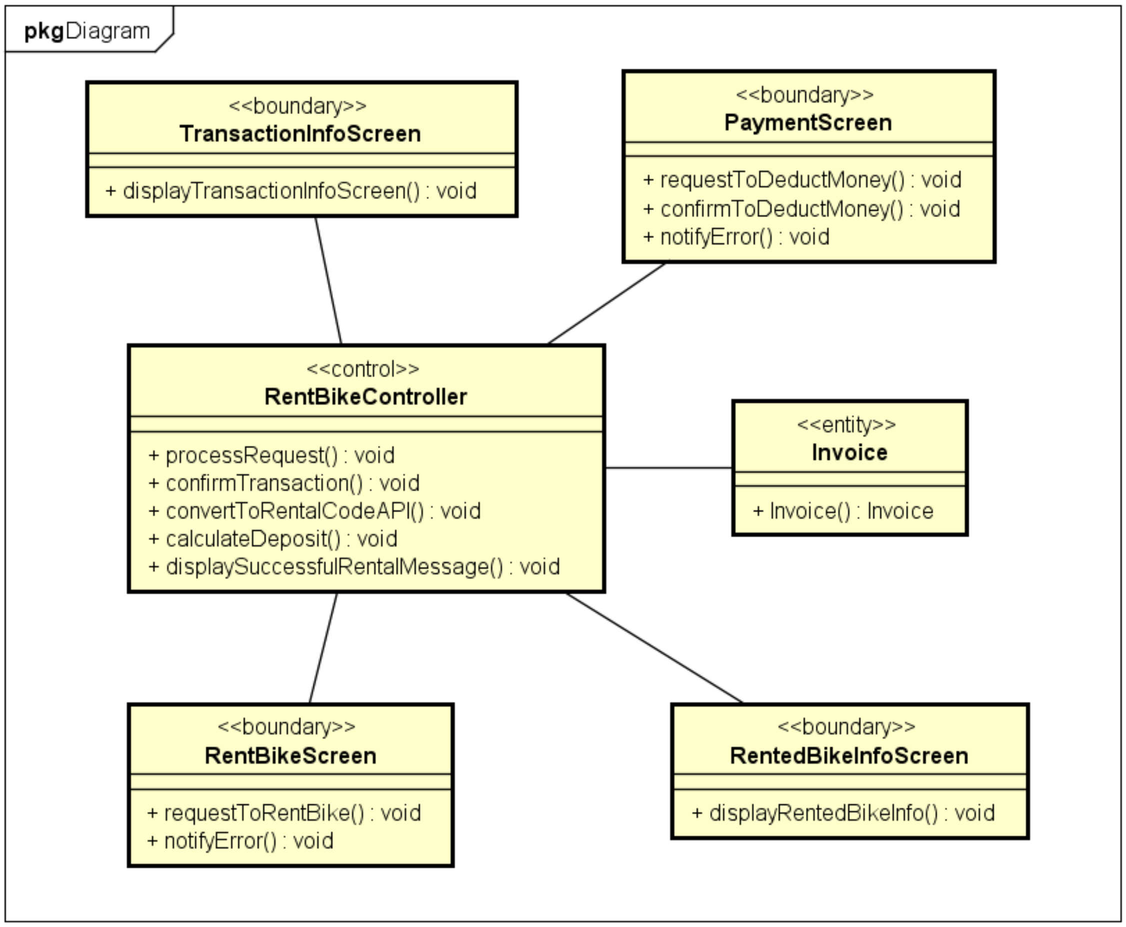

pkgDiagram
<<boundary>>
TransactionInfoScreen
+ displayTransactionInfoScreen() : void
<<boundary>>
PaymentScreen
+ requestToDeductMoney() : void
+ confirmToDeductMoney() : void
+ notifyError() : void
<<control>>
RentBikeController
+ processRequest() : void
+ confirmTransaction() : void
+ convertToRentalCodeAPI() : void
+ calculateDeposit() : void
+ displaySuccessfulRentalMessage() : void
<<entity>>
Invoice
+ Invoice() : Invoice
<<boundary>>
RentBikeScreen
+ requestToRentBike() : void
+ notifyError() : void
<<boundary>>
RentedBikeInfoScreen
+ displayRentedBikeInfo() : void

Fig 15. Analysis Class Diagram for Rent bike

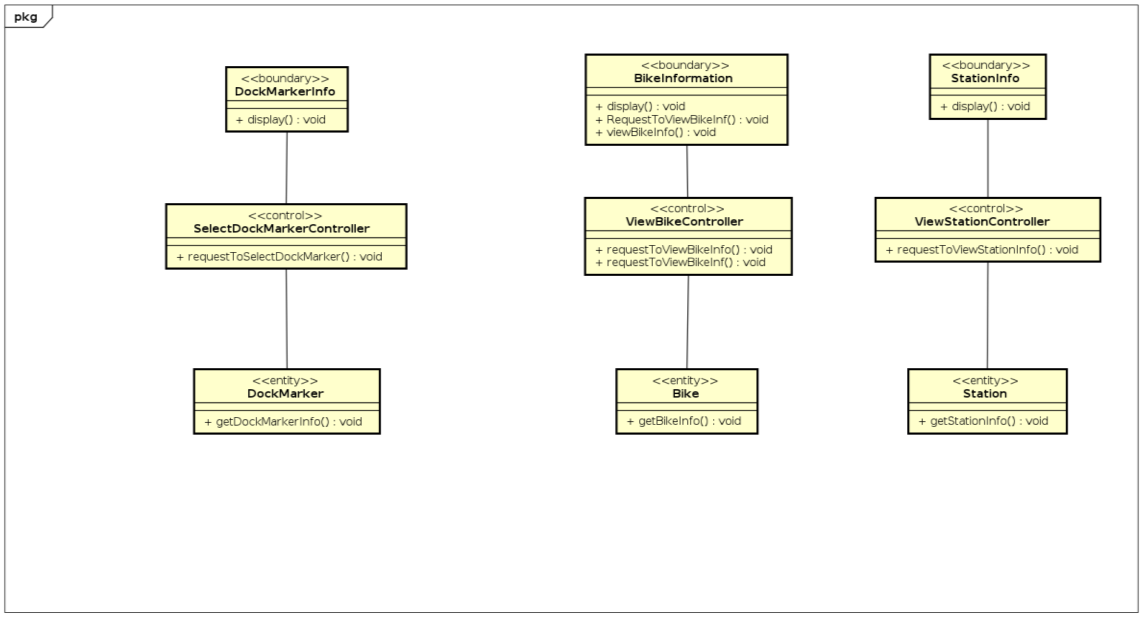


Fig 16. Analysis Class Diagram for Select Dock marker

3.4 *Unified Analysis Class Diagram*

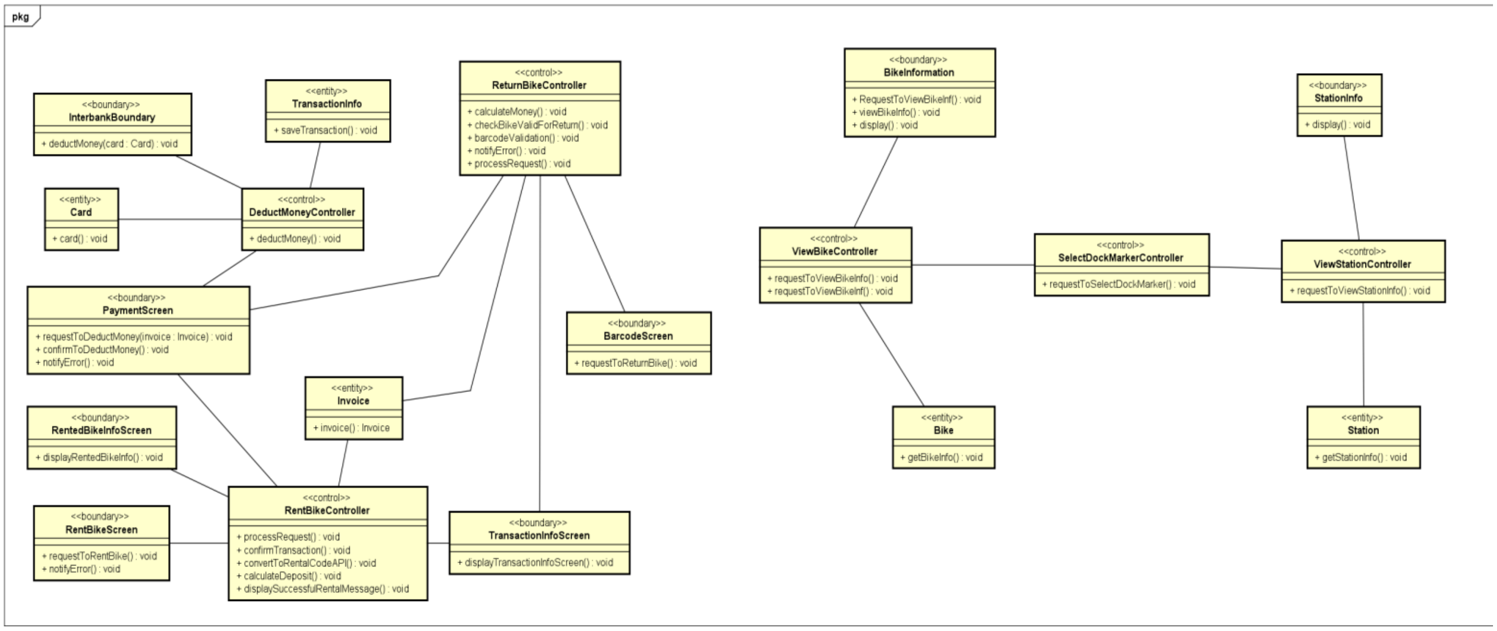

pkg
<<boundary>> InterbankBoundary
+ deductMoney(card : Card) : void
<<entity>> TransactionInfo
+ saveTransaction() : void
<<control>> ReturnBikeController
+ calculateMoney() : void
+ checkBikeValidForReturn() : void
+ barcodeValidation() : void
+ notifyError() : void
+ processRequest() : void
<<entity>> Card
+ card() : void
<<control>> DeductMoneyController
+ deductMoney() : void
<<boundary>> PaymentScreen
+ requestToDeductMoney(invoice : Invoice) : void
+ confirmToDeductMoney() : void
+ notifyError() : void
<<boundary>> BarcodeScreen
+ requestToReturnBike() : void
<<entity>> Invoice
+ invoice() : Invoice
<<boundary>> RentedBikeInfoScreen
+ displayRentedBikeInfo() : void
<<control>> RentBikeController
+ processRequest() : void
+ confirmTransaction() : void
+ convertToRentalCodeAPI() : void
+ calculateDeposit() : void
+ displaySuccessfulRentalMessage() : void
<<boundary>> RentBikeScreen
+ requestToRentBike() : void
+ notifyError() : void
<<boundary>> TransactionInfoScreen
+ displayTransactionInfoScreen() : void
<<boundary>> BikeInformation
+ RequestToViewBikeInf() : void
+ viewBikeInfo() : void
+ display() : void
<<boundary>> StationInfo
+ display() : void
<<control>> ViewBikeController
+ requestToViewBikeInfo() : void
+ requestToViewBikeInf() : void
<<control>> SelectDockMarkerController
+ requestToSelectDockMarker() : void
<<control>> ViewStationController
+ requestToViewStationInfo() : void
<<entity>> Bike
+ getBikeInfo() : void
<<entity>> Station
+ getStationInfo() : void

Fig 17. 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

<Suppose that you design a Graphical User Interface (GUI)>

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.

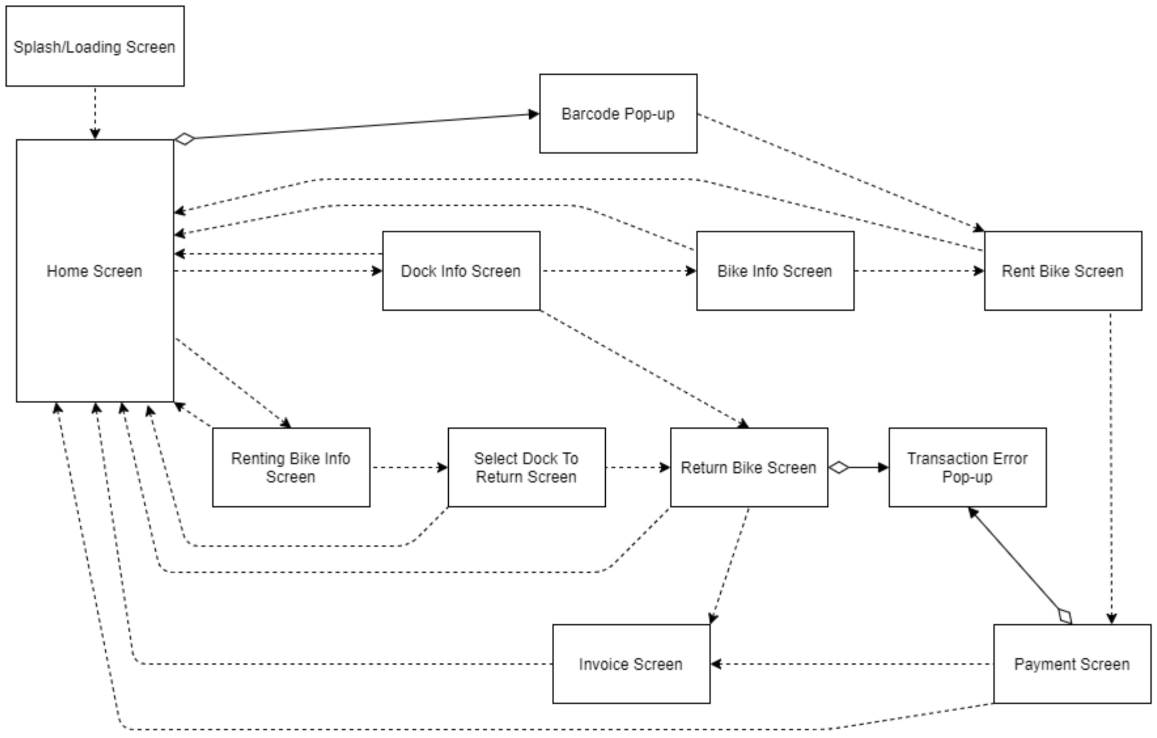
There is a Home button located at the top right of screen to go to Home screen.

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.

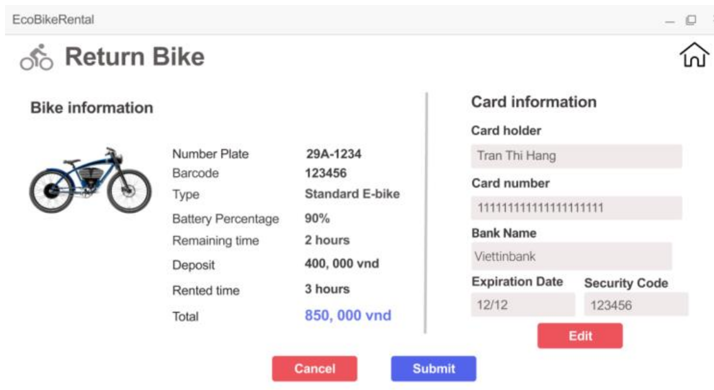
4.1.2 Screen Transition Diagrams



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	

EcoBikeRental

Return Bike

Bike information

Number Plate	29A-1234
Barcode	123456
Type	Standard E-bike
Battery Percentage	90%
Remaining time	2 hours
Deposit	400, 000 vnd
Rented time	3 hours
Total	850, 000 vnd

Card information

Card holder: Tran Thi Hang

Card number: 11111111111111111111

Bank Name: Viettinbank

Expiration Date: 12/12

Security Code: 123456

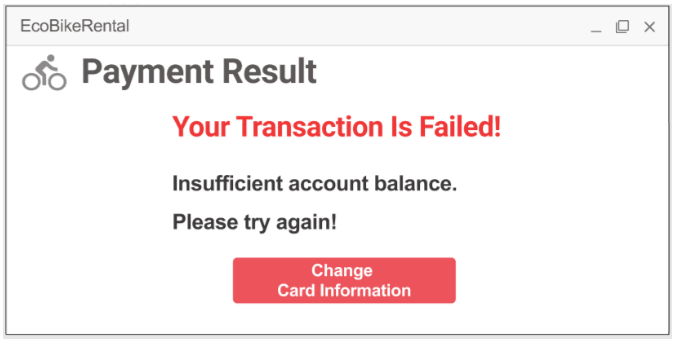
Defining the field attributes

Screen name	View cart			
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
Bank name	20	String	Black	Left justified
Expiration Date	10	String	Black	Left justified
Security Code	10	Numeral	Black	Left justified

4.1.3.2. Transaction Error Screen

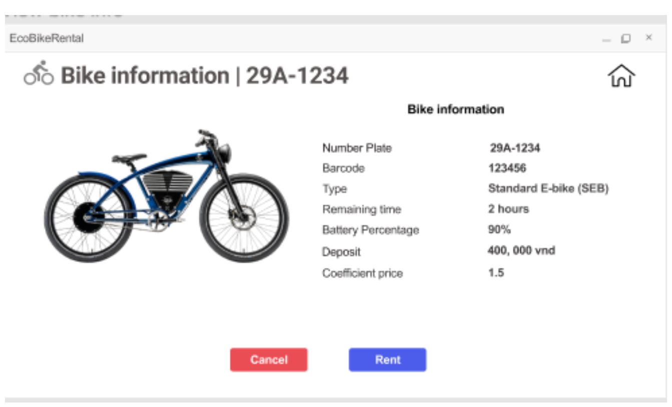
Screen specification

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Transaction Error Screen	7/11/2020			Trần Thị Hằng
 EcoBikeRental Payment Result Your Transaction Is Failed! Insufficient account balance. Please try again! Change Card Information		Control	Operation	Function	
		Change Card Information button	Click	Display Payment Screen	

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

EcoBikeRental

Bike information | 29A-1234

Bike information

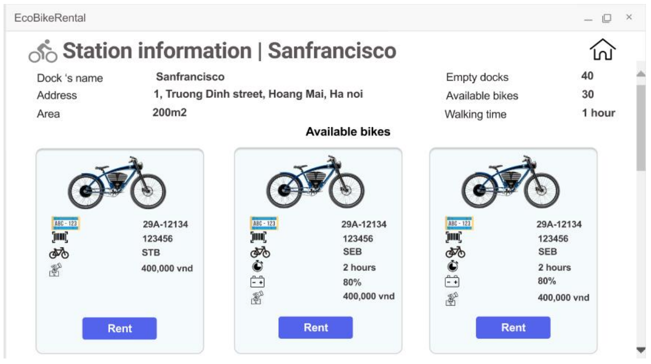
Number Plate	29A-1234
Barcode	123456
Type	Standard E-bike (SEB)
Remaining time	2 hours
Battery Percentage	90%
Deposit	400, 000 vnd
Coefficient price	1.5

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 Dock ‘s name	Initial	Display the name of dock	
		Area for display dock ‘s address	Initial	Display dock ‘s address	

EcoBikeRental

Station information | Sanfrancisco

Dock ‘s name	Sanfrancisco	Empty docks	40
Address	1, Truong Dinh street, Hoang Mai, Ha noi	Available bikes	30
Area	200m2	Walking time	1 hour

Available bikes

29A-12134	29A-12134	29A-12134
123456	123456	123456
STB	SEB	SEB
400,000 vnd	2 hours	2 hours
	80%	80%
	400,000 vnd	400,000 vnd

		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

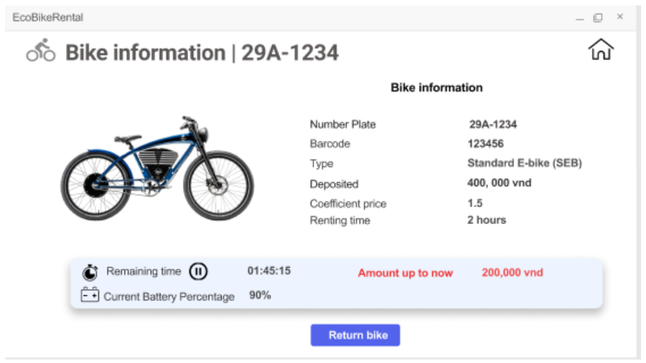
Defining the field attributes

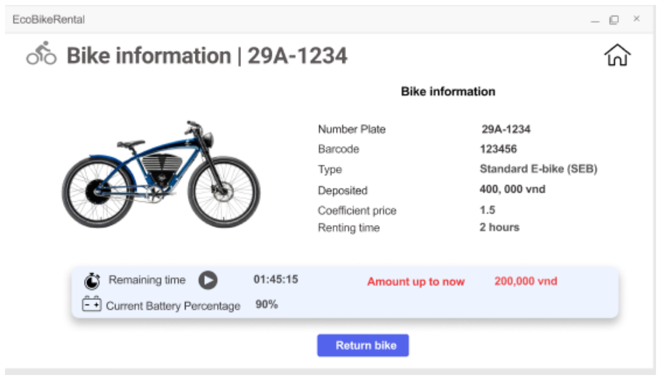
Screen name	View bike information screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Number plate	8	Numeral	Black	Left-justified
Barcode	6	Numeral	Black	Left justified

Type	100	Text	Black	Left-justified
Remaining time	50	Text	Black	Left-justified
Battery percentage	4	Text	Black	Left-justified
Deposit	6	Number	Black	Left-justified

4.1.3.5. View Renting Bike Screen

Screen specification

EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	View renting bike information screen	07/11/2020			Dương Thị Huê
 EcoBikeRental Bike information | 29A-1234 Bike information Number Plate 29A-1234 Barcode 123456 Type Standard E-bike (SEB) Deposited 400, 000 vnd Coefficient price 1.5 Renting time 2 hours Remaining time 01:45:15 Amount up to now 200,000 vnd Current Battery Percentage 90% Return bike		Control	Operation	Function	
		Area for displaying the bike information	Initial	Display bike information	
		Area for display remaining time	Initial	Display remaining time of user	
		Area for display Renting time	initial	Display the renting time	
		Area for display amount up to now	initial	Display amount to be paid up to now	
		Area for home icon	click	Back to home	

 EcoBikeRental Bike information | 29A-1234 Bike information Number Plate 29A-1234 Barcode 123456 Type Standard E-bike (SEB) Deposited 400, 000 vnd Coefficient price 1.5 Renting time 2 hours Remaining time 01:45:15 Amount up to now 200,000 vnd Current Battery Percentage 90%	Area for display deposit	initial	Display deposit
	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

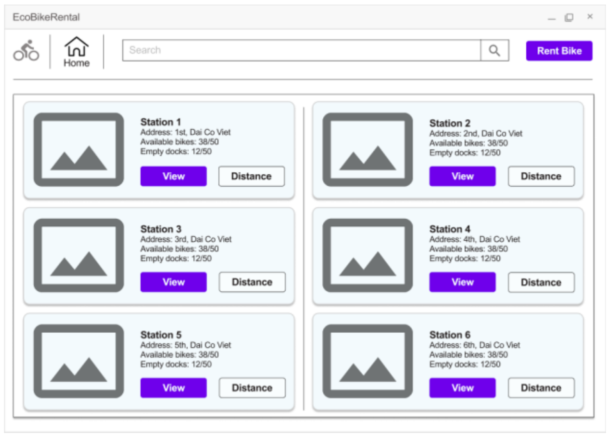
Defining the field attributes

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
Remaining time	8	Hh:mm:ss	black	Left-justified

4.1.3.6. Home Screen

Screen specification

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

Screen specification	Home screen	07/11/2020			Đỗ Minh Thông
 EcoBikeRental Home Search Rent Bike Station 1 Address: 1st, Dai Co Viet Available bikes: 38/50 Empty docks: 12/50 View Distance Station 2 Address: 2nd, Dai Co Viet Available bikes: 38/50 Empty docks: 12/50 View Distance Station 3 Address: 3rd, Dai Co Viet Available bikes: 38/50 Empty docks: 12/50 View Distance Station 4 Address: 4th, Dai Co Viet Available bikes: 38/50 Empty docks: 12/50 View Distance Station 5 Address: 5th, Dai Co Viet Available bikes: 38/50 Empty docks: 12/50 View Distance Station 6 Address: 6th, Dai Co Viet Available bikes: 38/50 Empty docks: 12/50 View Distance		Control	Operation	Function	
		Area for displaying the list of docks	Initial	Display 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	

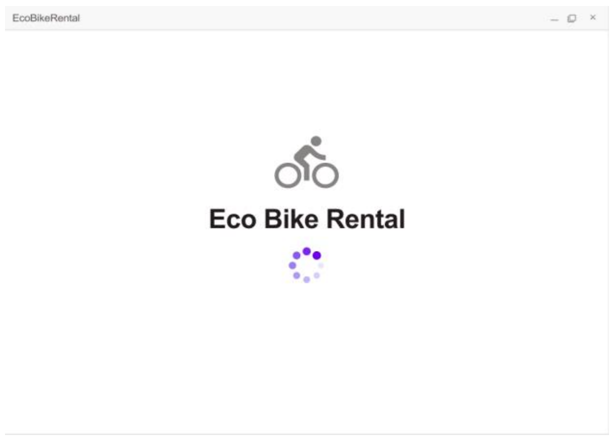
Defining the field attributes

Screen name	View bike information screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Station information	256	Text	Black	Left-justified

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

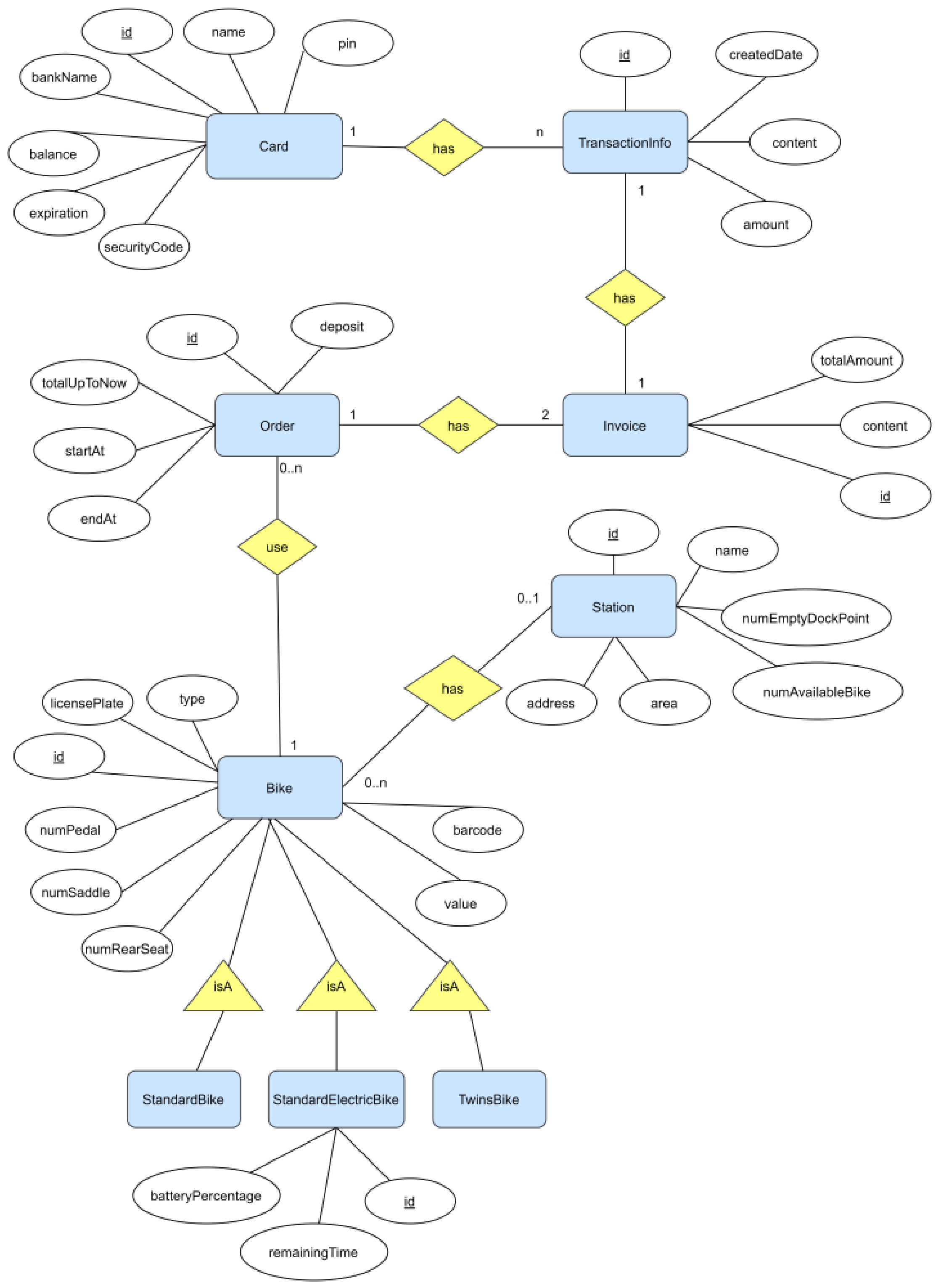
EcobikeRental Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	View distance popup screen	07/11/2020			Đỗ Minh Thông
		Control	Operation	Function	
		Area for displaying the station information and walking time estimate	Initial	Display station information and walking time estimate	
		Back 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	Numeral	Black	Left-justified
Estimated time	50	Numeral	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

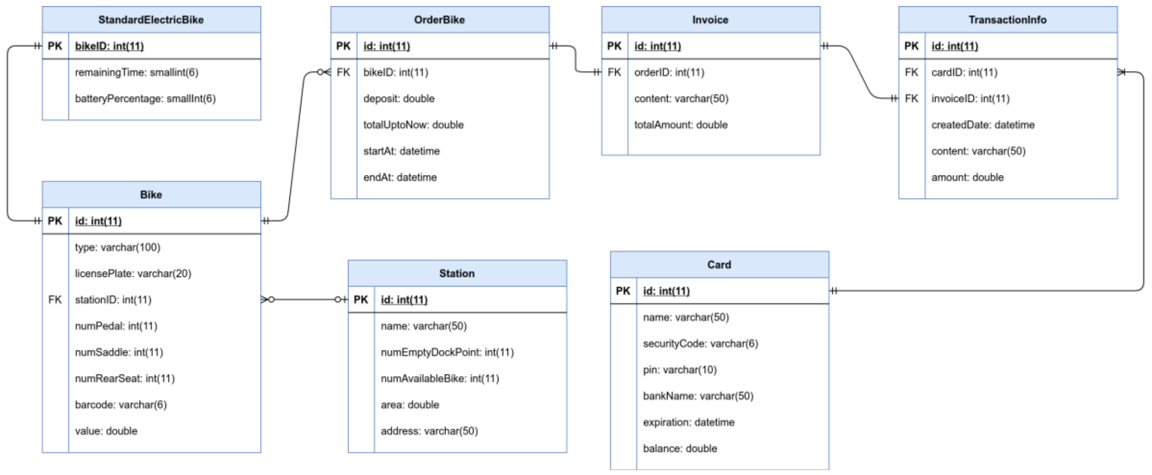
- Database Management System: MySql
- MySQLis the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo!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

4.2.2.2 Logical Data Model

<

- *Show the process to design database from E-R diagram*
- *Show the diagram of DB design*

>



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			name	Varchar(50)	Yes	Name of Card's holder
3			securityCode	Varchar(6)	Yes	Security code of card

4			pin	Varchar(10)	Yes	PIN number
5			bankName	Varchar(50)	Yes	Name of Interbank
6			expiration	datetime	Yes	Expiration date of card
7			balance	double	Yes	Balance 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			type	Varchar(50)	Yes	type of bike
3			licensePlate	Varchar(6)	Yes	license plate of the bike

4			numPedal	Integer	Yes	numbers of pedal
5			numSaddle	Integer	Yes	numbers of saddle of the bike
6			numRearSeat	Integer	Yes	numbers of rear seat
7			value	Integer	Yes	value of the bike
8			Barcode	Varchar(6)	Yes	Barcode of the bike

- **StandardElectricBike**

#	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		x	remainingTime	Integer	Yes	remaining time of the bike

- **Order**

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		id	Integer	Yes	ID of order, auto increment
2			deposit	double	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			totalUpToNow	double	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	double	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 Card(
id INT AUTO_INCREMENT PRIMARY KEY,
NAME VARCHAR(50) not null,
securityCode VARCHAR(6) not null,
pin VARCHAR(10) not null,
bankName VARCHAR(50) not null,
expiration DATETIME not null,
balance DOUBLE not null
);
CREATE TABLE Order (
id INT AUTO_INCREMENT PRIMARY KEY,
deposit double not null,
totalUpToNow DOUBLE,
bikeID int not null,
startAt DATETIME not null,
endAt DATETIME,
FOREIGN KEY bikeID REFERENCES Bike(id)
);
CREATE TABLE Invoice(
id INT AUTO_INCREMENT PRIMARY KEY,
content VARCHAR(50) not null,
totalAmount DOUBLE not null,
orderID int not null,
FOREIGN KEY orderID REFERENCES Order(id)
);
CREATE TABLE TransactionInfo(
id INT AUTO_INCREMENT PRIMARY KEY,
cardID INT,
invoiceID INT,
createdDate DATETIME,
content VARCHAR(50),
amount DOUBLE,
FOREIGN KEY cardID REFERENCES Card(id),
FOREIGN KEY invoiceID REFERENCES Invoice(id)
);
CREATE TABLE Station(
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(50) not null,
numEmptyDockPoint INT not null,
numAvailableBike INT not null,
area DOUBLE not null,
address VARCHAR(50) not null
);
create table Bike (
type varchar(100),
licensePlate varchar(20),
id int AUTO_INCREMENT PRIMARY key,
stationID int ,
numPedal int ,
numSaddle int ,
numRearSeat int ,
barcode varchar(6) not null,
value double not null,
FOREIGN key stationID REFERENCES Station(id)
)
create table StandardElectricBike(
```

id int primary key,

batteryPercentage int(2) ,

remainingTime double,

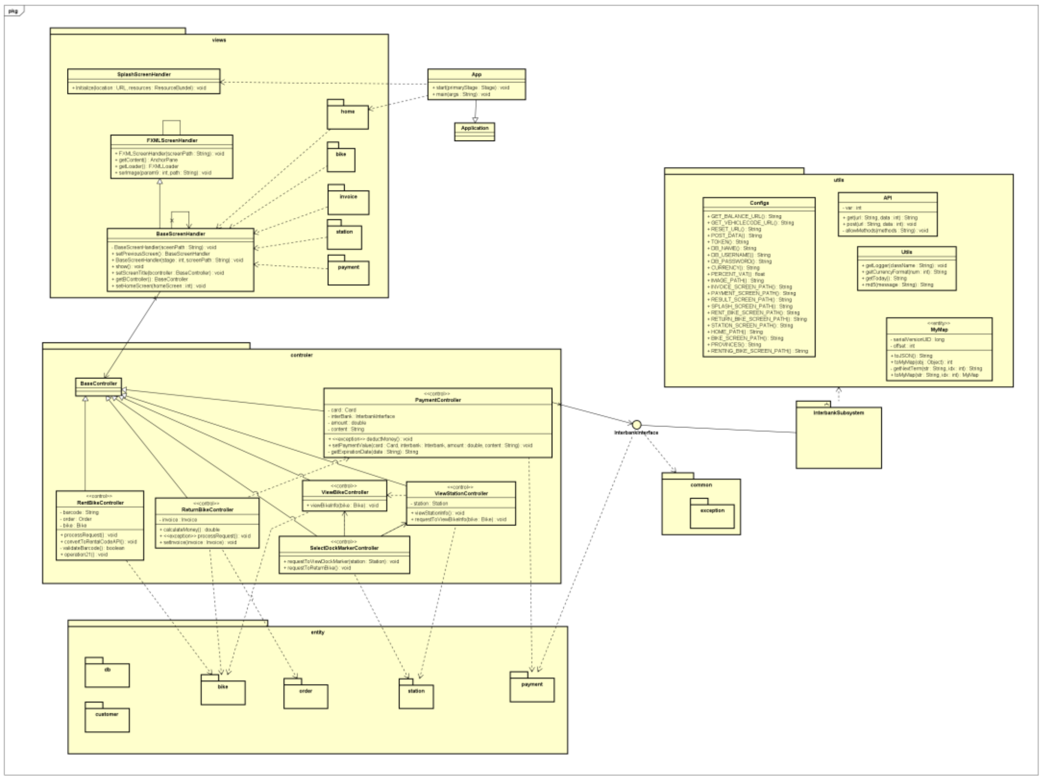
foreign key id references Bike(id)

)

4.3 Non-Database Management System Files

4.4 Class Design

4.4.1 General Class Diagram



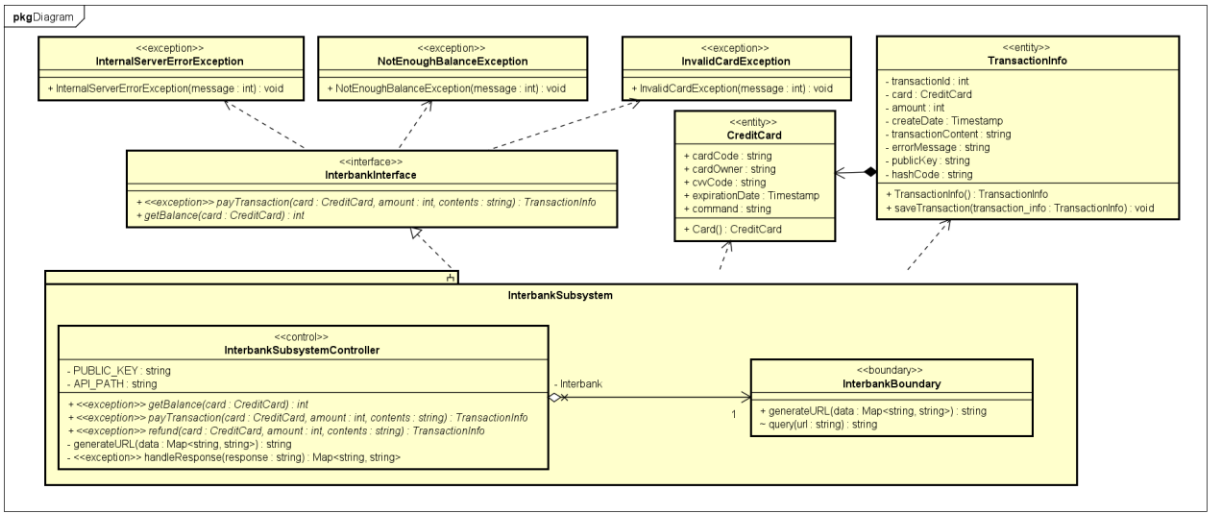
pkg
views
SplashScreenHandler
+ initialize(location : URL, resources : ResourceBundel) : void
App
+ start(primaryStage : Stage) : void
+ main(args : String) : void
Application
home
bike
invoice
station
payment
FXMLScreenHandler
+ FXMLScreenHandler(screenPath : String) : void
+ getContent() : AnchorPane
+ getLoader() : FXMLLoader
+ setImage(param9 : int, path : String) : void
BaseScreenHandler
- BaseScreenHandler(sceenPath : String) : void
+ setPreviousScreen() : BaseScreenHandler
+ BaseScreenHandler(stage : int, screenPath : String) : void
+ show() : void
+ setScreenTitle(bcontroller : BaseController) : void
+ getBController() : BaseController
+ setHomeScreen(homeScreen : int) : void
utils
Configs
+ GET_BALANCE_URL() : String
+ GET_VEHICLECODE_URL() : String
+ RESET_URL() : String
+ POST_DATA() : String
+ TOKEN() : String
+ DB_NAME() : String
+ DB_USERNAME() : String
+ DB_PASSWORD() : String
+ CURRENCY() : String
+ PERCENT_VAT() : float
+ IMAGE_PATH() : String
+ INVOICE_SCREEN_PATH() : String
+ PAYMENT_SCREEN_PATH() : String
+ RESULT_SCREEN_PATH() : String
+ SPLASH_SCREEN_PATH() : String
+ RENT_BIKE_SCREEN_PATH() : String
+ RETURN_BIKE_SCREEN_PATH() : String
+ STATION_SCREEN_PATH() : String
+ HOME_PATH() : String
+ BIKE_SCREEN_PATH() : String
+ PROVINCES() : String
+ RENTING_BIKE_SCREEN_PATH() : String
API
- var : int
+ get(url : String, data : int) : String
+ post(url : String, data : int) : void
- allowMethods(methods : String) : void
Utils
+ getLogger(className : String) : void
+ getCurrencyFormat(num : int) : String
+ getToday() : String
+ md5(message : String) : String
<<entity>>
MyMap
- serialVersionUID : long
- offset : int
+ toJSON() : String
+ toMyMap(obj : Object) : int
- getNextTerm(str : String, idx : int) : String
+ toMyMap(str : String, idx : int) : MyMap
controler
BaseController
<<control>>
PaymentController
- card : Card
- interBank : InterbankInterface
- amount : double
- content : String
+ <<exception>> deductMoney() : void
+ setPaymentValue(card : Card, interbank : Interbank, amount : double, content : String) : void
- getExpirationDate(date : String) : String
InterbankInterface
InterbankSubsystem
common
exception
<<control>>
RentBikeController
- barcode : String
- order : Order
- bike : Bike
+ processRequest() : void
+ convertToRentalCodeAPI() : void
- validateBarcode() : boolean
+ operation21() : void
<<control>>
ReturnBikeController
- invoice : Invoice
+ calculateMoney() : double
+ <<exception>> processRequest() : void
+ setInvoice(invoice : Invoice) : void
<<control>>
ViewBikeController
+ viewBikeInfo(bike : Bike) : void
<<control>>
ViewStationController
- station : Station
+ viewStationInfo() : void
+ requestToViewBikeInfo(bike : Bike) : void
<<control>>
SelectDockMarkerController
+ requestToViewDockMarker(station : Station) : void
+ requestToReturnBike() : void
entity
db
customer
bike
order
station
payment

4.4.2 Class Diagrams

4.4.2.1 Class Diagram for Package A

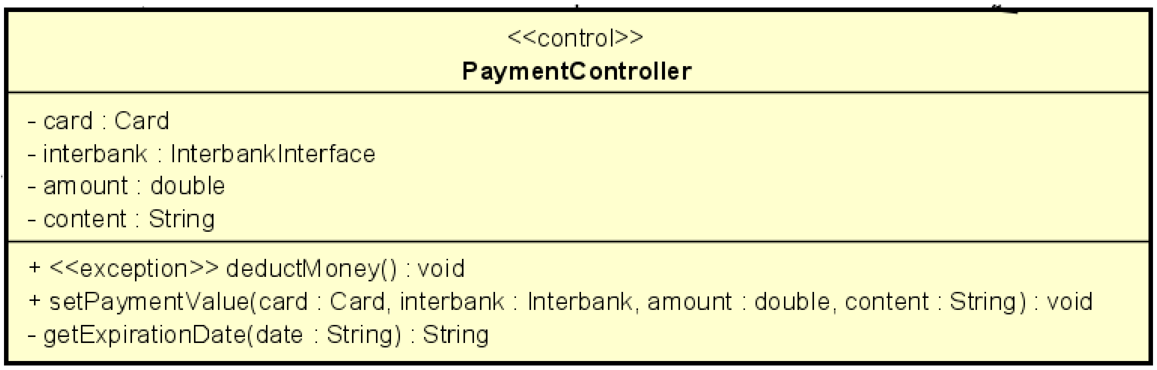
…

4.4.2.2 Class Diagram for Interbank Subsystem


pkgDiagram
<<exception>>
InternalServerErrorException
+ InternalServerErrorException(message : int) : void
<<exception>>
NotEnoughBalanceException
+ NotEnoughBalanceException(message : int) : void
<<exception>>
InvalidCardException
+ InvalidCardException(message : int) : void
<<entity>>
TransactionInfo
- transactionId : int
- card : CreditCard
- amount : int
- createDate : Timestamp
- transactionContent : string
- errorMessage : string
- publicKey : string
- hashCode : string
+ TransactionInfo() : TransactionInfo
+ saveTransaction(transaction_info : TransactionInfo) : void
<<entity>>
CreditCard
+ cardCode : string
+ cardOwner : string
+ cvvCode : string
+ expirationDate : Timestamp
+ command : string
+ Card() : CreditCard
<<interface>>
InterbankInterface
+ <<exception>> payTransaction(card : CreditCard, amount : int, contents : string) : TransactionInfo
+ getBalance(card : CreditCard) : int
InterbankSubsystem
<<control>>
InterbankSubsystemController
- PUBLIC_KEY : string
- API_PATH : string
+ <<exception>> getBalance(card : CreditCard) : int
+ <<exception>> payTransaction(card : CreditCard, amount : int, contents : string) : TransactionInfo
+ <<exception>> refund(card : CreditCard, amount : int, contents : string) : TransactionInfo
- generateURL(data : Map<string, string>) : string
- <<exception>> handleResponse(response : string) : Map<string, string>
- Interbank
1
<<boundary>>
InterbankBoundary
+ generateURL(data : Map<string, string>) : string
~ query(url : string) : string

4.4.3 Class Design

4.4.3.1 Class "PaymentController"


<<control>>
PaymentController
- card : Card
- interbank : InterbankInterface
- amount : double
- content : String
+ <<exception>> deductMoney() : void
+ setPaymentValue(card : Card, interbank : Interbank, amount : double, content : String) : void
- getExpirationDate(date : String) : String

Attribute

#	Name	Data type	Default value	Description
1	card	Card	NULL	Represent the card used for payment
2	amount	double	NULL	Represent total amount of the transaction
3	content	String	NULL	Represent content of transaction
4	interbank	InterbankInterface	NULL	Represent the interbank

Operation

#	Name	Return type	Description (purpose)
1	deductMoney	void	Send request to interbank API to deduct money from card to pay for renting bike
2	setPaymentValue	void	Set value for all attributes of *PaymentController* class

Parameter:

- card – the credit card used for payment
- interbank – interbank of transaction
- amount – total amount of transaction
- content – content of transaction

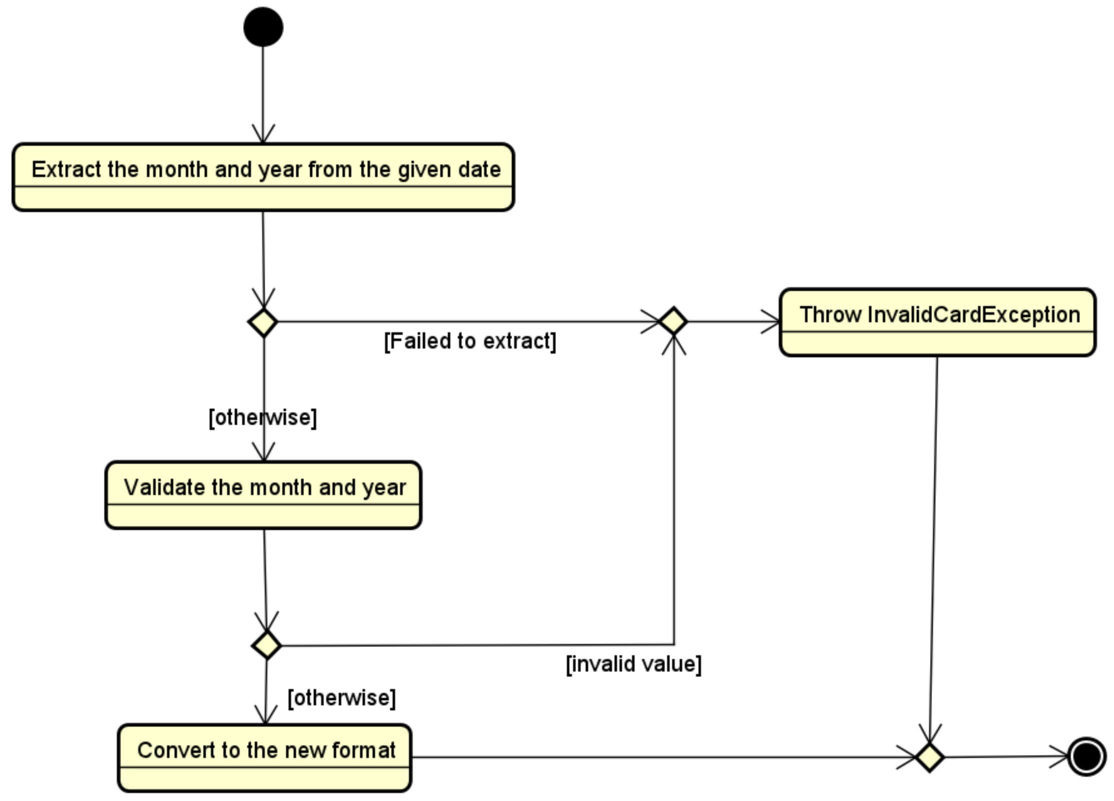
Exception:

- PaymentException: if responded with error that transaction is failed

State : None

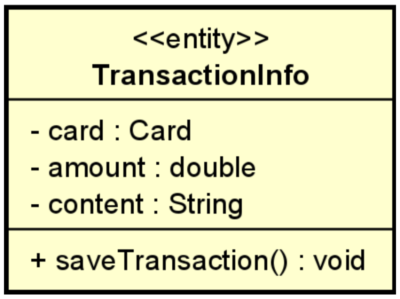
Method

- getExpirationDate: given the String "date" representing the expiration date in the format ""mm/yy", this method converts it into the required format "mmyy". The algorithm is illustrated as follows.


Extract the month and year from the given date
[Failed to extract]
Throw InvalidCardException
[otherwise]
Validate the month and year
[invalid value]
[otherwise]
Convert to the new format

-

4.4.3.2 Class "TransactionInfo"



Attribute

#	Name	Data type	Default value	Description
1	card	Card	NULL	Represent the card used for payment
2	amount	double	NULL	Represent total amount of the transaction
3	content	String	NULL	Represent content of transaction

Operation

#	Name	Return type	Description (purpose)
1	saveTransaction	void	Save the transaction between software and interbank

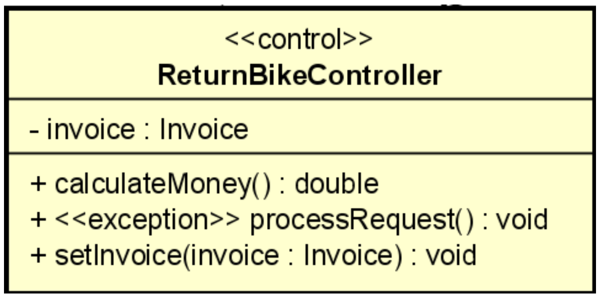
Parameter: None

Exception: None

Method: None

State: None

4.4.3.3 Class "ReturnBikeController"



<<control>>
ReturnBikeController
- invoice : Invoice
+ calculateMoney() : double
+ <<exception>> processRequest() : void
+ setInvoice(invoice : Invoice) : void

Attribute

#	Name	Data type	Default value	Description
1	invoice	Invoice	NULL	Represent invoice of returning bike

Operation

#	Name	Return type	Description (purpose)
1	calculateMoney	double	Calculate total amount that customer has to pay when return bike
2	processRequest	void	Process returning bike request, call *ReturnBikeHandler* and *InvoiceHandler* class
3	setInvoice	void	Set value for invoice after calculate money

Parameter:

- Invoice – invoice of returning bike

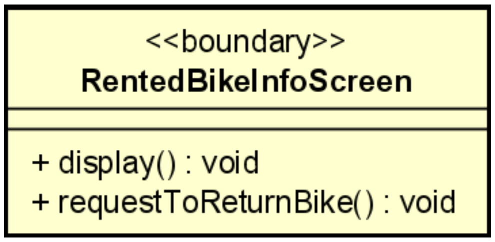
Exception:

- ReturnBikeException: if responded with error that return bike request is failed

State: None

Method: None

4.4.3.4 Class "RentedBileInfoScreen"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	display	void	Display detail of renting bike include amount up to now that customer has to pay
2	requestToReturnBike	void	Request to return bike

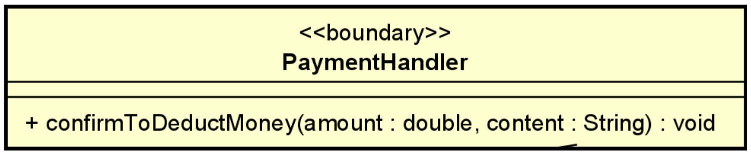
Parameter: None

Exception: None

Method None

State None

4.4.3.5 Class "PaymentHandler"


<<boundary>>
PaymentHandler
+ confirmToDeductMoney(amount : double, content : String) : void

Attribute

Operation

#	Name	Return type	Description (purpose)

1	confirmToDeductMoney	void	Confirm to deduct money from card to pay for renting bike

Parameter:

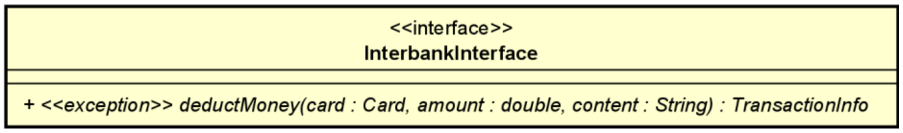
- amount: total amount that interbank will deduct from customer's card
- content: content of transaction

Exception: None

Method None

State None

4.4.3.6 Class "InterbankInterface"



<<interface>>
InterbankInterface
+ <<exception>> deductMoney(card : Card, amount : double, content : String) : TransactionInfo

Attribute

Operation

#	Name	Return type	Description (purpose)
1	deductMoney	void	Confirm to deduct money from card to pay for renting bike

Parameter:

- amount: total amount that interbank will deduct from customer's card
- content: content of transaction
- card: the credit card used for payment

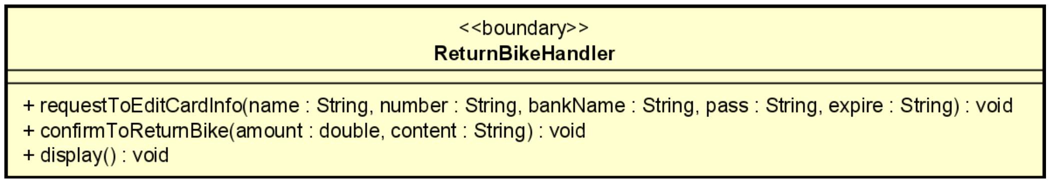
Exception:

- InterbankPaymentException: if responded with a pre-defined error code
- UnrecognizeException: if responded with an unknown error code or something goes wrong

Method: None

State: None

4.4.3.7 Class "ReturnBikeHandler"


<<boundary>>
ReturnBikeHandler
+ requestToEditCardInfo(name : String, number : String, bankName : String, pass : String, expire : String) : void
+ confirmToReturnBike(amount : double, content : String) : void
+ display() : void

Attribute

Operation

#	Name	Return type	Description (purpose)
1	requestToEditCardInfo	void	Edit card information
2	confirmToReturnBike	void	Confirm to return bike
3	display	void	Display detailed information of return bike and card information of customer

Parameter:

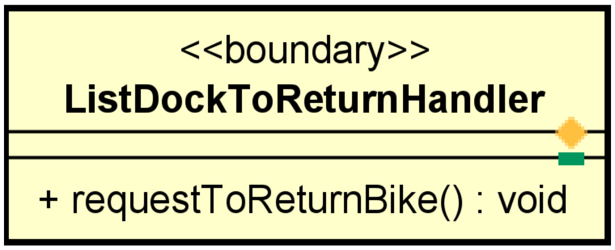
- amount: total amount that customer has to pay for returning bike
- content: content of transaction
- name: card's holder name
- number: card number
- bankName: name of interbank
- pass: security code of card
- expire: expiration date of card

Exception: None

Method: None

State: None

4.4.3.8 Class "ListDockToReturnHandler"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	requestToReturnBike	void	Request to return bike, call selectDockMakerController

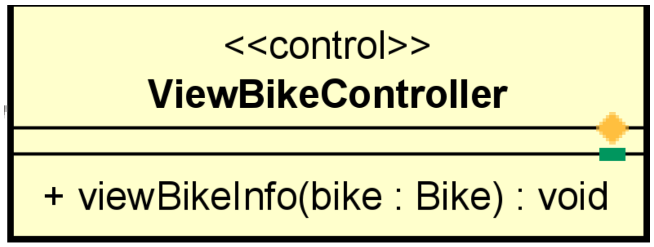
Parameter: None

Exception: None

Method: none

State: none

4.4.3.9 Class "ViewBikeController"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	viewBikeInfo	void	Process request view specific bike info

Parameter

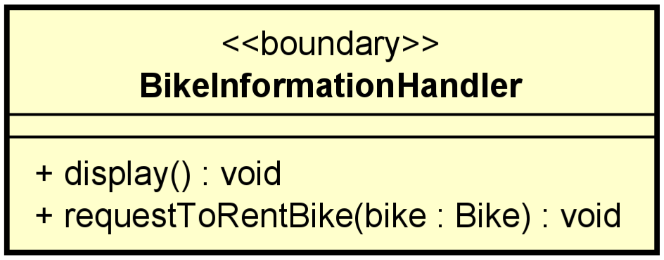
- bike: bike that want to view information

Exception: ViewBikeException

Method: none

State: none

4.4.3.10 Class "BikeInformationHandler"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	display	void	Display bike information
2	requestToRentBike	Void	When user submit to rent bike, sent request to rentBikeController

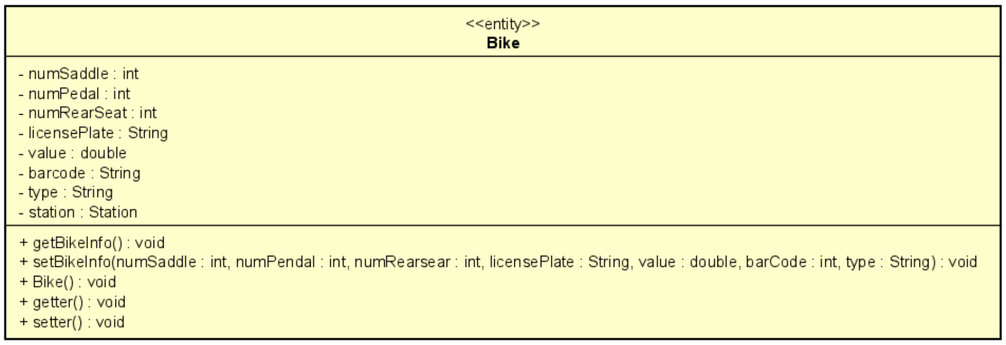
Parameter

- bike: bike that user want to rent

Method: none

State: none

4.4.3.11 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	double	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

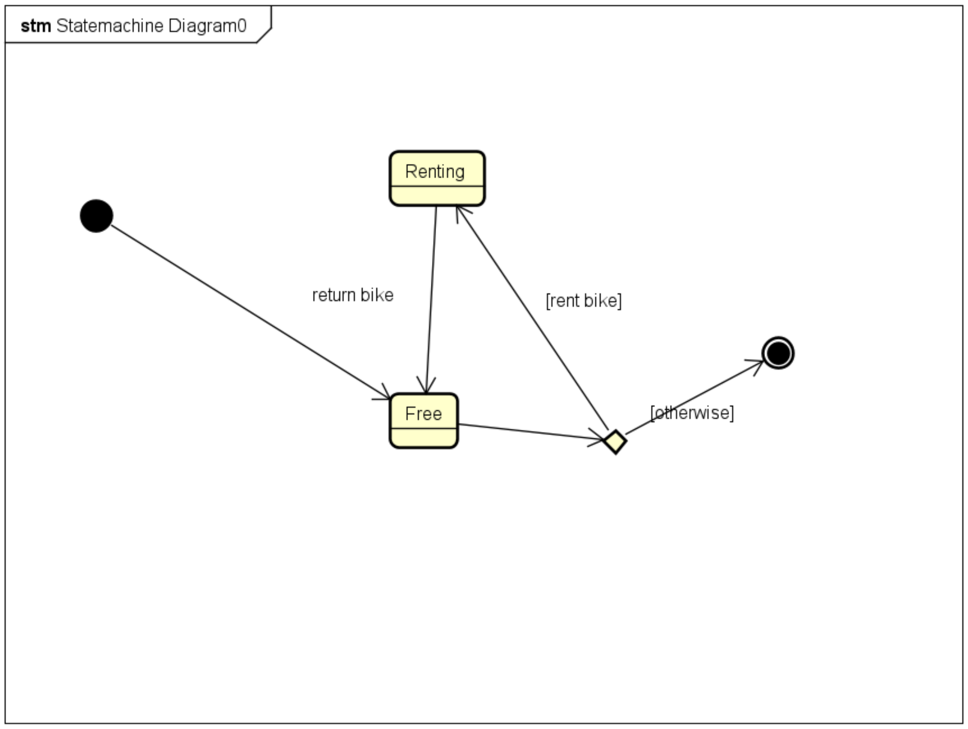
Operation

#	Name	Return type	Description (purpose)
1	getBikeInfo	void	Get bike information for display
2	setBikeInfo	void	Set bike information
3	Bike	void	Constructor
4	getter	void	Get all attribute in acronym
5	setter	void	Set value for each attribute in acronym

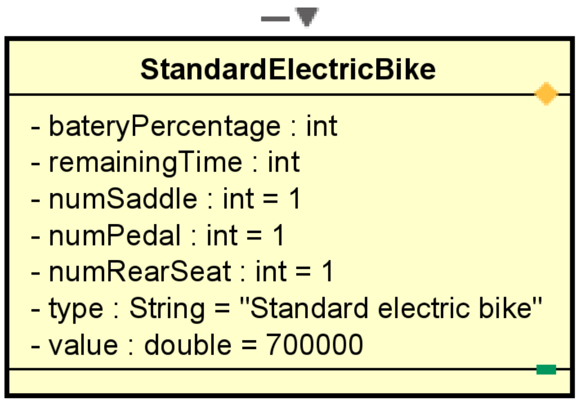
Parameter: same like attribute

Method: none

State:



4.4.3.12 Class "StandardElectricBike"



Attribute

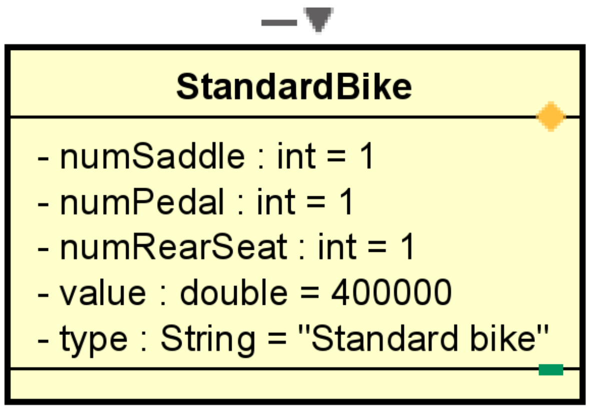
#	Name	Data type	Default value	Description
1	numSaddle	int	1	Unchanged value
2	numPedal	int	1	Constant value
3	numRearSeat	int	1	Constant value
4	value	double	700000	Constant value
5	type	String	"Standard electric bike"	Constant type

Operation: inherit

Method: none

State: none

4.4.3.13 Class "StandardBike"



Attribute

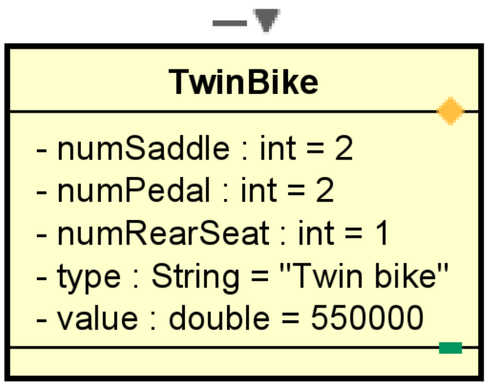
#	Name	Data type	Default value	Description
1	numSaddle	int	1	Unchanged value
2	numPedal	int	1	Constant value
3	numRearSeat	int	1	Constant value
4	value	double	400000	Constant value
5	type	String	“Standard bike”	Constant type

Operation: inherit

Method: none

State: none

4.4.3.14 Class “TwinBike”



Attribute

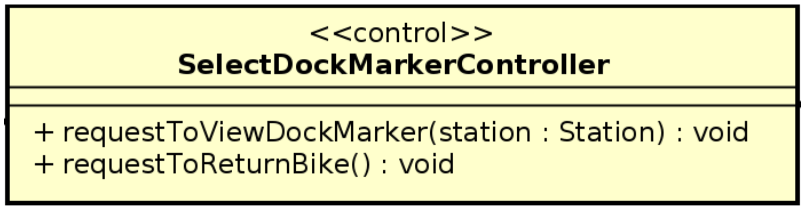
#	Name	Data type	Default value	Description
1	numSaddle	int	2	Unchanged value
2	numPedal	int	2	Constant value
3	numRearSeat	int	1	Constant value
4	value	double	550000	Constant value
5	type	String	“Twin bike”	Constant type

Operation: inherit

Method: none

State: none

4.4.3.15 Class "SelectDockMarkerController"



<<control>>
SelectDockMarkerController
+ requestToViewDockMarker(station : Station) : void
+ requestToReturnBike() : void

Attribute

Operation

#	Name	Return type	Description (purpose)
1	requestToViewDockMarker	void	Process request to view the dock marker
2	RequestToReturnBike	void	Process to request to return the bike

Parameter

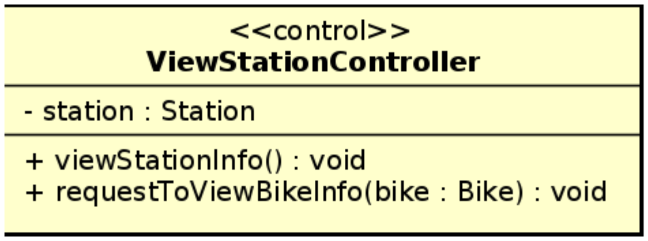
- station: station that want to view information

Exception: ViewStationException

Method: none

State: none

4.4.3.16 Class "ViewStationController"



<<control>>
ViewStationController
- station : Station
+ viewStationInfo() : void
+ requestToViewBikeInfo(bike : Bike) : void

Attribute station: station that user want to view information

Operation

#	Name	Return type	Description (purpose)
1	viewStationInfo	void	View station information
2	requestToViewBikeInfo	Void	Process to request to view bike information

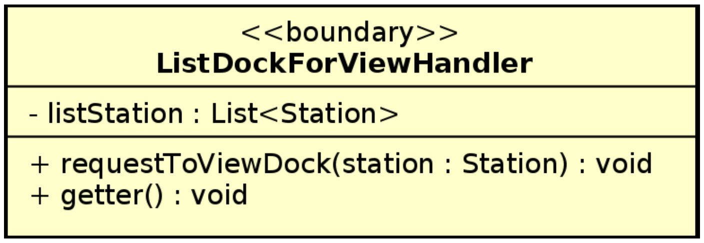
Parameter

- bike: bike that user want to rent

Method: none

State: none

4.4.3.17 Class "ListDockForViewHandler"



<<boundary>>
ListDockForViewHandler
- listStation : List<Station>
+ requestToViewDock(station : Station) : void
+ getter() : void

Attribute listStation: a list of stations

Operation

#	Name	Return type	Description (purpose)
1	requestToViewDock	void	Process to request to view dock
2	getter	Void	get method

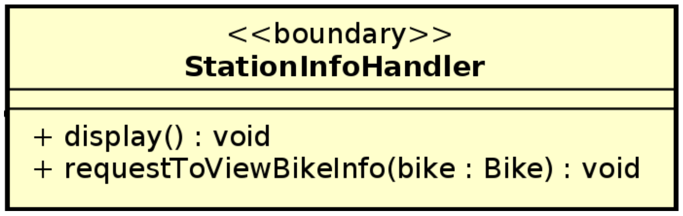
Parameter

- station: station that user want to view information

Method: none

State: none

4.4.3.18 Class "StationInfoHandler"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	display	void	Display station information

2	requestToViewBikeInfo	Void	Process to request to view bike information

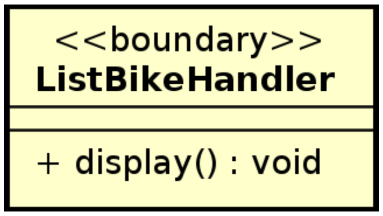
Parameter

- bike: bike that user want to view information

Method: none

State: none

4.4.3.19 Class "ListBikeHandler"



Attribute

Operation

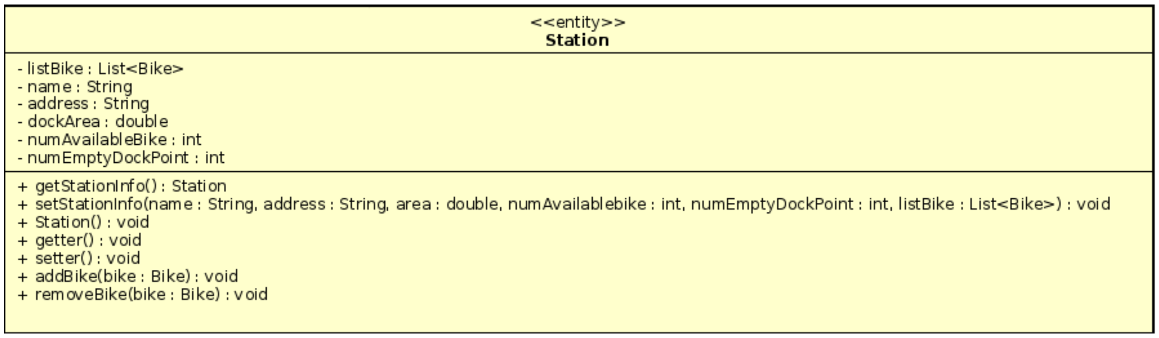
#	Name	Return type	Description (purpose)
1	display	void	Display list of bikes

Parameter: none

Method: none

State: none

4.4.3.20 Class "Station"


<<entity>>
Station
- listBike : List<Bike>
- name : String
- address : String
- dockArea : double
- numAvailableBike : int
- numEmptyDockPoint : int
+ getStationInfo() : Station
+ setStationInfo(name : String, address : String, area : double, numAvailablebike : int, numEmptyDockPoint : int, listBike : List<Bike>) : void
+ Station() : void
+ getter() : void
+ setter() : void
+ addBike(bike : Bike) : void
+ removeBike(bike : Bike) : void

Attribute

#	Name	Data type	Default value	Description
1	listBike	List<Bike>	NULL	List the bikes in the station

2	name	String	NULL	Name of the station
3	address	String	NULL	Address of the station
4	dockArea	double	NULL	Area of the dock
5	numAvailableBike	int	NULL	Number of available bikes
6	numEmptyDockPoint	int	NULL	Number of empty dock points

Operation

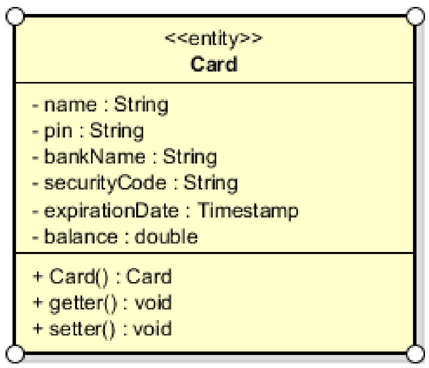
#	Name	Return type	Description (purpose)
1	getStationInfo	Station	Get station information for display
2	setStationInfo	void	Set station information
3	Station	void	Constructor
4	getter	void	Get all attribute in acronym
5	setter	void	Set value for each attribute in acronym
6	addBike	void	Add bike to the station if user return bike
7	removeBike	void	Remove bike in the station if user rent bike

Parameter: same like attribute

Method: none

State: none

4.4.3.21 Class "Card"



Attribute

#	Name	Data type	Default value	Description
1	name	String	NULL	Name of the owner of the credit card
2	pin	String	NULL	Pin code of the credit card
3	bankName	String	NULL	The name of the bank that provides the credit card
4	securityCode	String	NULL	Security code of the credit card
5	expirationDate	Timestamp	NULL	The expiration date of the card, in form MM/YYYY
6	Balance	double	0.00	The balance of the card

Operation

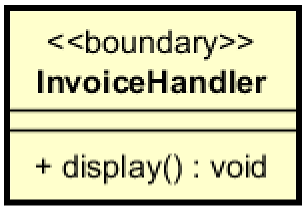
#	Name	Return type	Description (purpose)
1	Card	Card	Constructor
2	getter	void	Get all attribute in acronym
3	setter	void	Set value for each attribute in acronym

Parameter: same as attributes

State: None

Method: None

4.4.3.22 Class "InvoiceHandler"



Attribute

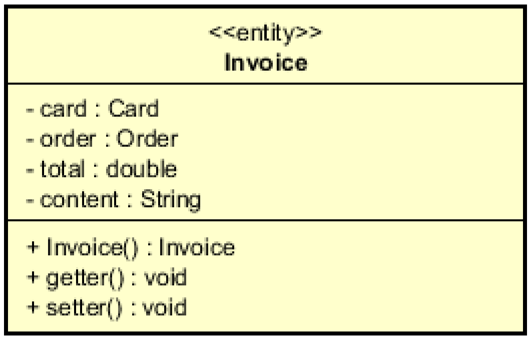
Operation

#	Name	Return type	Description (purpose)
1	display	void	Display the invoice screen

State: None

Method: None

4.4.3.23 Class "Invoice"



Attribute

#	Name	Data type	Default value	Description
1	card	Card	NULL	The credit card that was used for this transaction/invoice
2	order	Order	NULL	The order that was used for this transaction/invoice
3	total	double	0.00	The amount of money that was transferred in the transaction/The amount of money that was deducted from the credit card in the transaction
4	content	String	NULL	The details of the transaction/invoice

Operation

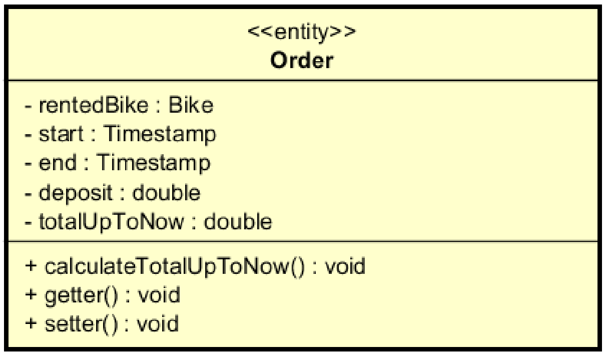
#	Name	Return type	Description (purpose)
1	Invoice	Invoice	Constructor
2	getter	void	Get all attribute in acronym
3	setter	void	Set value for each attribute in acronym

Parameter: same as attributes

State: None

Method: None

4.4.3.24 Class "Order"



Attribute

#	Name	Data type	Default value	Description
1	rentedBike	Bike	NULL	The bike that was rented by user
2	start	Timestamp	Time when the bike was rented	The timestamp which user rented the bike
3	end	Timestamp	Current Time	The timestamp which user returns bike (in case this order is for returning bike), or current time (in case this order is for renting bike)
4	deposit	double	0.00	The deposit when renting the bike
5	totalUpToNow	double	0.00	Total renting money up to now (not include deposit)

Operation

#	Name	Return type	Description (purpose)
1	calculateTotalUpToNow	void	Calculate the renting amount up to now (not include deposit)
2	getter	void	Get all attribute in acronym

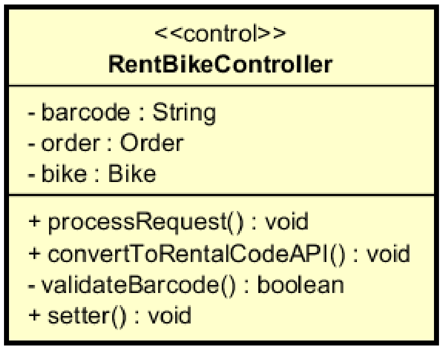
3	setter	void	Set value for each attribute in acronym

Parameter: same as attributes

State: None

Method: None

4.4.3.25 Class "RentBikeController"



Attribute

#	Name	Data type	Default value	Description
1	barcode	String	NULL	The barcode of the rented bike
2	order	Order	NULL	The order is made when renting bike
3	bike	Bike	NULL	The rented bike

Operation

#	Name	Return type	Description (purpose)
1	processRequest	void	Process the request of renting bike, to see that if the request comes from user entering the barcode or user choosing the bike
2	validateBarcode	boolean	In case the user entering the barcode (the barcode attribute is not empty), then we should check if the barcode is valid. If it is, call the setter of the bike

			attribute. If it is not, an error is displayed in the barcode screen.
3	convertToRentalCodeAPI	void	Call the API to convert the barcode into rental code
4	setter	void	Set value for each attribute in acronym

Parameter

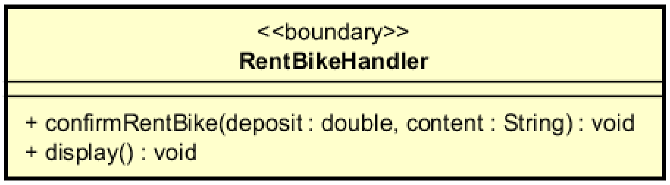
Exception:

- RentBikeException: if responded with error that rent bike request is failed

State: None

Method: None

4.4.3.26 Class "RentBikeHandler"



<<boundary>>
RentBikeHandler
+ confirmRentBike(deposit : double, content : String) : void
+ display() : void

Attribute

Operation

#	Name	Return type	Description (purpose)
1	display	void	Display the rent bike screen
2	confirmRentBike	void	Display the payment screen, send the deposit amount and the transaction content to the payment controller

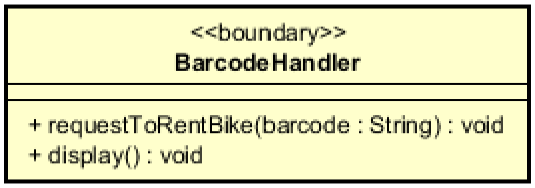
Parameter:

- deposit - the deposit amount that user has to pay when renting bike
- content - the details of the transaction when sending to payment

State: None

Method: None

4.4.3.27 Class "BarCodeHandler"



Attribute

Operation

#	Name	Return type	Description (purpose)
1	display	void	Display the barcode screen
2	requestToRentBike	void	After the user inputs the barcode, this function sends the barcode to controller to process the rent-bike request

Parameter:

- barcode: the code that user inputs when he/she wants to rent bike

State: None

Method: 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*

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 .