HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

Hệ thống EcoBikeRental

Subject: Thiết kế xây dựng phần mềm

Group 11

Nguyễn Thị Thắm 20183984

Phạm Thị Vân 20184015

Phạm Thị Duyên 20180067

*Hanoi**, 12/2021*

*<All notations inside the angle bracket are not part of this document, for its purpose is for extra instruction. When using this document, please erase all these notations and/or replace them with corresponding content as instructed>*

*<This document, written by Asst. Prof. NGUYEN Thi Thu Trang, is used as a case study for student with related courses. Any modifications and/or utilization without the consent of the author is strictly forbidden>*

Table of Contents

Table of Contents 1

1 Introduction 3

1.1 Objective 3

1.2 Scope 3

1.3 Glossary 3

1.4 References 3

2 Overall Description 4

2.1 General Overview 4

2.2 Assumptions/Constraints/Risks 4

2.2.1 Assumptions 4

2.2.2 Constraints 4

2.2.3 Risks 5

3 System Architecture and Architecture Design 6

3.1 Architectural Patterns 6

3.2 Interaction Diagrams 6

3.3 Analysis Class Diagrams 6

3.4 Unified Analysis Class Diagram 6

3.5 Security Software Architecture 6

4 Detailed Design 7

4.1 User Interface Design 7

4.1.1 Screen Configuration Standardization 7

4.1.2 Screen Transition Diagrams 7

4.1.3 Screen Specifications 7

4.2 Data Modeling 7

4.2.1 Conceptual Data Modeling 7

4.2.2 Database Design 7

4.3 Non-Database Management System Files 11

4.4 Class Design 12

4.4.1 General Class Diagram 12

4.4.2 Class Diagrams 13

4.4.3 Class Design 16

5 Design Considerations 33

5.1 Goals and Guidelines 33

5.2 Architectural Strategies 33

5.3 Coupling and Cohesion 34

5.4 Design Principles 34

5.5 Design Patterns 34

**List of Figures**

**List of Tables**

No table of figures entries found.

# Introduction

*<The following subsections of the Software Design Document (SDD) document should provide an overview of the entire SDD.>*

## Objective

<*Identify the purpose of this SDD and its intended audience. In this subsection, describe the purpose of the SDD and specify the intended audience for the SDD*>

## Scope

<*In this subsection:*

1. *Identify the software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here*

*Note that this will be similar to what was written in the SRS.*

>

## Glossary

*<Listing and explaining the terms appearing in the software’s profession and this document. Any assumption of the reader’s prior knowledge or experience on the subject is ill advised>*

## 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

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

# Overall Description

<*This section describes the principles and strategies to be used as guidelines when designing and implementing the system.>*

## General Overview

*<Briefly introduce the system context and the basic design approach or organization. Provide a brief overview of the system and software architectures and the design goals. Include the high-level context diagram(s) for the system and subsystems provided in previous documents like SRS (e.g., general use case diagram, lower-level use case diagrams, activity diagrams), updated as necessary to reflect any changes that have been made based on more current information or understanding. If the high-level context diagram has been updated, identify the changes that were made and why>*

## Assumptions/Constraints/Risks

### Assumptions

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

### Constraints

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

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

*>*

### Risks

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

# System Architecture and Architecture Design

<*Briefly describe the architectural design steps*>

## Architectural Patterns

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

## Interaction Diagrams

## Analysis Class Diagrams

## Unified Analysis Class Diagram

## Security Software Architecture

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

# Detailed Design

## User Interface Design

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

### Screen Configuration Standardization

### Screen Transition Diagrams

### Screen Specifications

*<Screen images should be included in the screen specifications>*

## Data Modeling

### Conceptual Data Modeling

*<E-R Diagram image and description of entities and relationships>*

### Database Design

#### Database Management Systems

*<Specify what is the decision of Database Management System (DBMS) and give some description of the DBMS>*

#### Logical Data Model

<

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

*>*

#### Physical Data Model

**RENT\_INFO**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | integer | No | Yes | Id of rentInfo |
| 2 |  |  | startTime | Date | No | Yes | Start time |
| 3 |  |  | endTime | Date | No | Yes | End time |
| 4 |  |  | rentType | Varchar (20) | No | Yes | Type of rent |
| 5 |  |  | rentPeriod | int | No | Yes | Time to rent |
| 6 |  |  | depositAmount | int | No | Yes | Amount of deposit |
| 7 |  |  | rentAmount | int | No | Yes | Amount rent |
| 8 |  |  | isComplete | boolean | No | Yes | True if complete |
| 9 |  | x | bikeId | int | No | Yes | Bike id |

**CARD**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | integer | No | Yes | Id, auto increment |
| 2 |  |  | cardCode | Varchar(15) | No | Yes | Card code |
| 3 |  |  | owner | Varchar(45) | No | Yes | Name of owner |
| 4 |  |  | dateExpired | Varchar(4) | No | Yes | Date: mmyy |

**TRANSACTION**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | integer | No | Yes | Id of transaction |
| 2 |  |  | method | Varchar(20) | No | Yes | Payment method |
| 3 |  |  | content | Varchar(50) | No | Yes | content |
| 4 |  |  | createAt | Date | No | Yes | Time to create |
| 5 |  |  | cardNumber | Varchar(20) | No | Yes | Card number |
| 6 |  | x | rentInfoId | int | No | Yes | rentInfo id |

**DOCK**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. | x |  | id | int | x | ID, auto increment |
| 2. |  |  | name | varchar (45) | x | name of dock |
| 3. |  |  | numCell | int | x | number of cells in dock |
| 4. |  |  | area | float | x | area of dock |
| 5. |  |  | address | varchar (255) | x | address of dock |

**Bike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. | x |  | id | int | x | ID, auto increment |
| 2. |  |  | licensePlate | varchar (45) | x | license plate of bike |
| 3. |  |  | type | String | x | type of bike |
| 4. |  |  | isAvailable | boolean | x | check bike has been rented |

**Twin\_EBike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. |  |  | remainingBattery | int | x | remaining battery of twin electric bike |
| 2. |  |  | BIKEcode | float | x | code of bike to caculate deposit amount |

**Twin\_Bike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. |  |  | BIKEcode | float | x | code of bike to caculate deposit amount |

**STANDARD\_EBike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. |  |  | remainingBattery | int | x | remaining battery of twin electric bike |
| 2. |  |  | BIKEcode | float | x | code of bike to caculate deposit amount |

**STANDARD \_Bike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. |  |  | BIKEcode | float | x | code of bike to caculate deposit amount |

**CellBike**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **PK** | **FK** | **Column** | **Data type** | **Mandatory** | **Description** |
| 1. | x |  | id | int | x | ID, auto increment |
| 2. |  |  | type | varchar (45) | x | type of cell |
| 3. |  | x | dockId | int | x | id of dock |
| 4. |  | x | bikeId | int |  | id of bike |

*You may add indexing, trigger, view, etc.*

*Give the database script*>

## Non-Database Management System Files

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

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

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

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

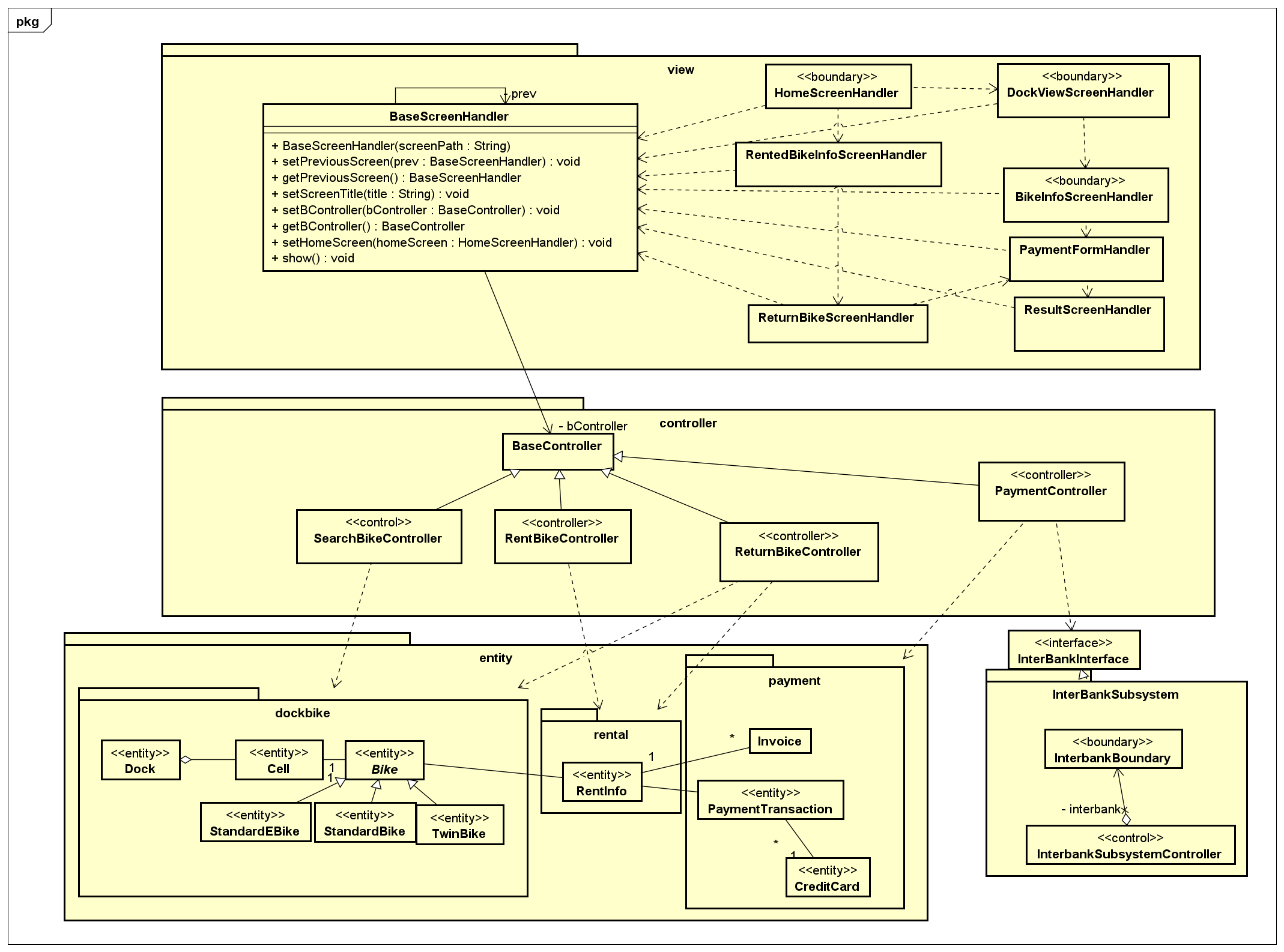
*• Estimate of the file size or volume of data within the file, including overhead resulting from file access methods*

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

*• Backup and recovery specifications>*

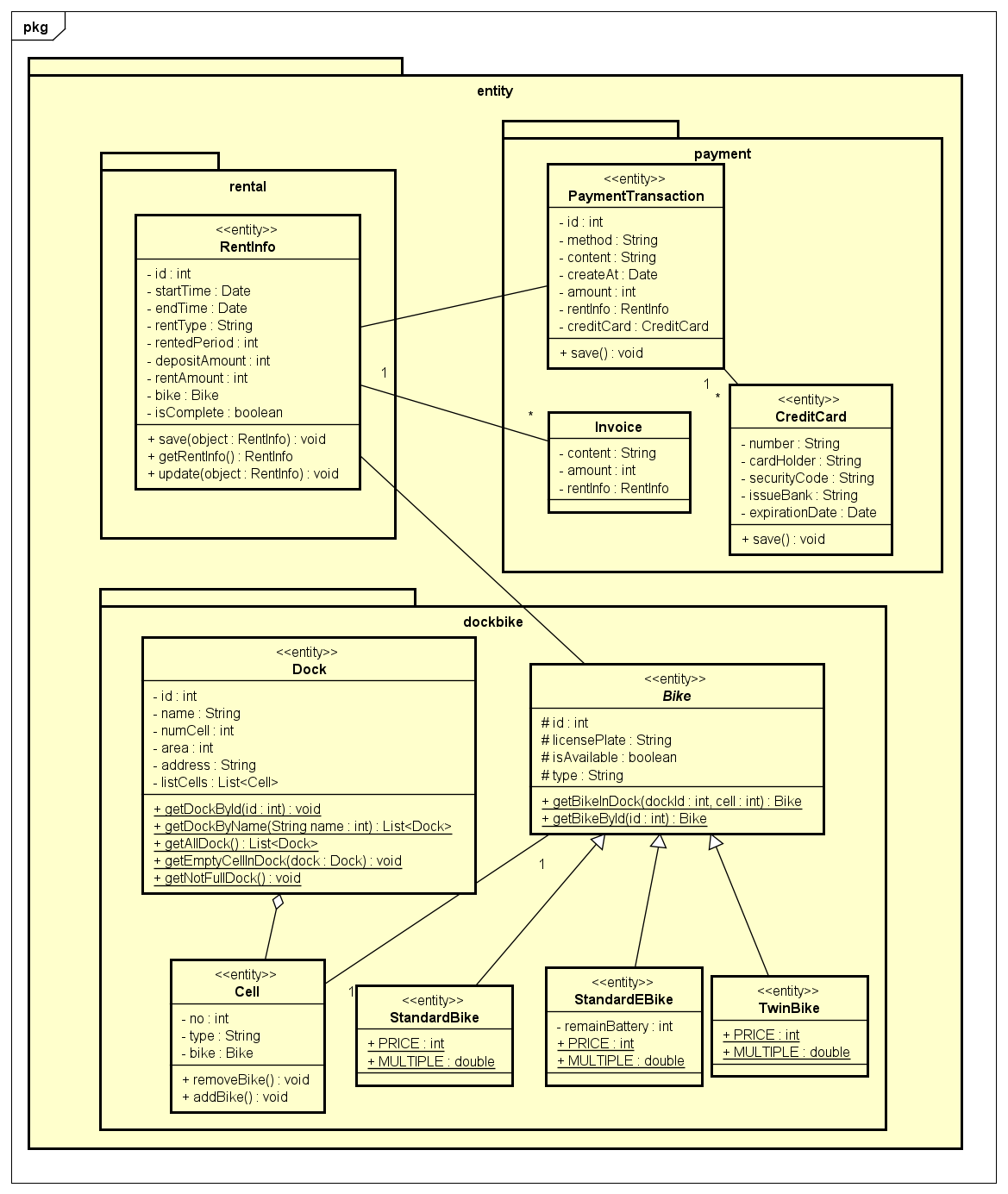
## Class Design

### General Class Diagram

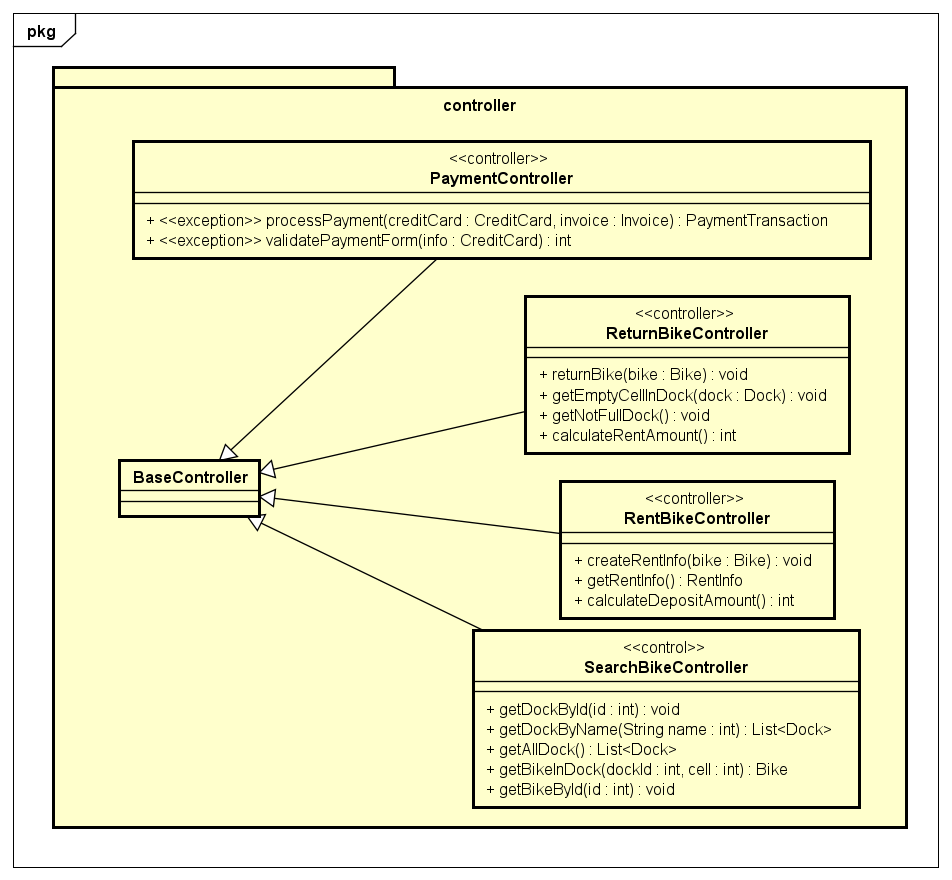


### Class Diagrams

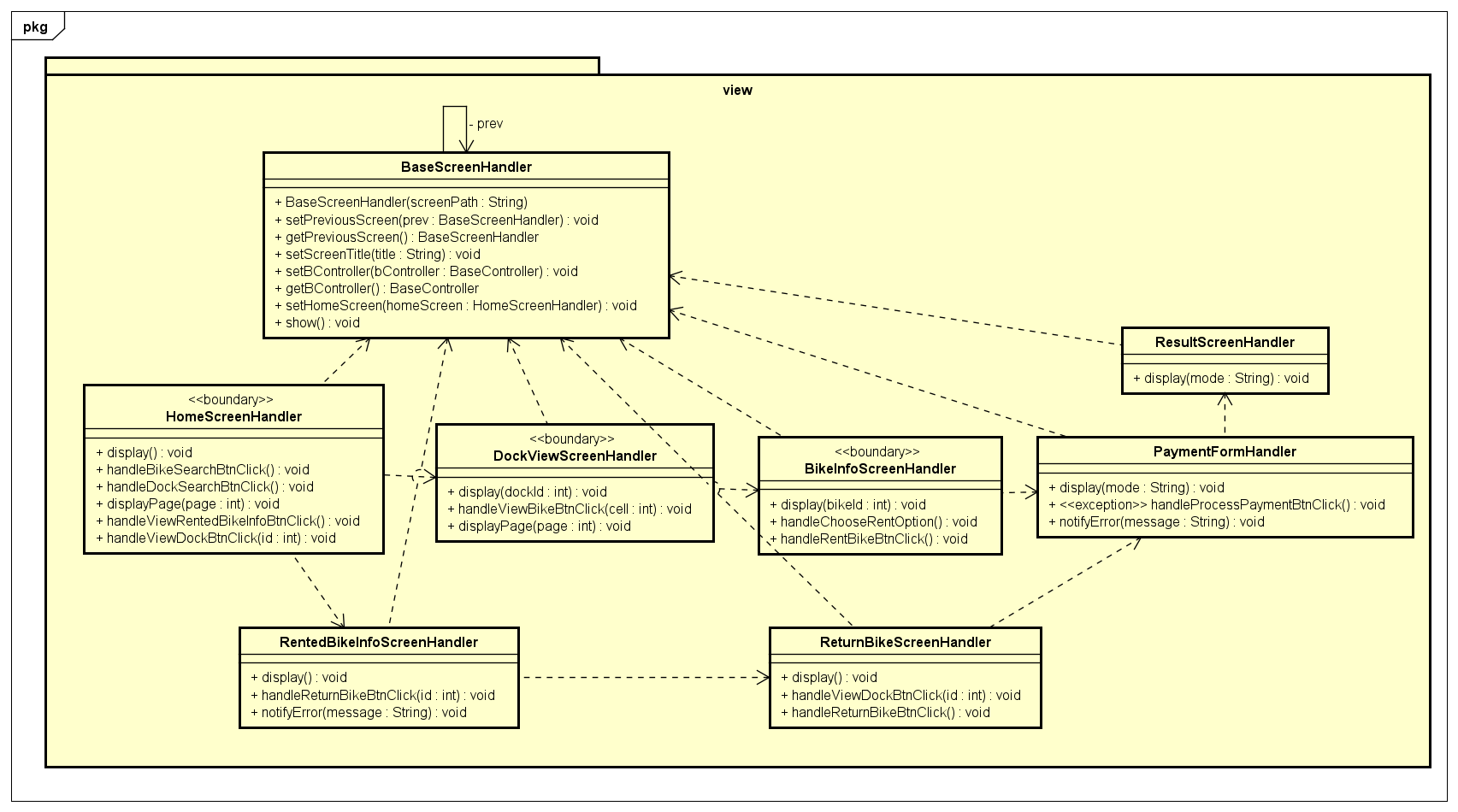
#### Class Diagram for Package Entity



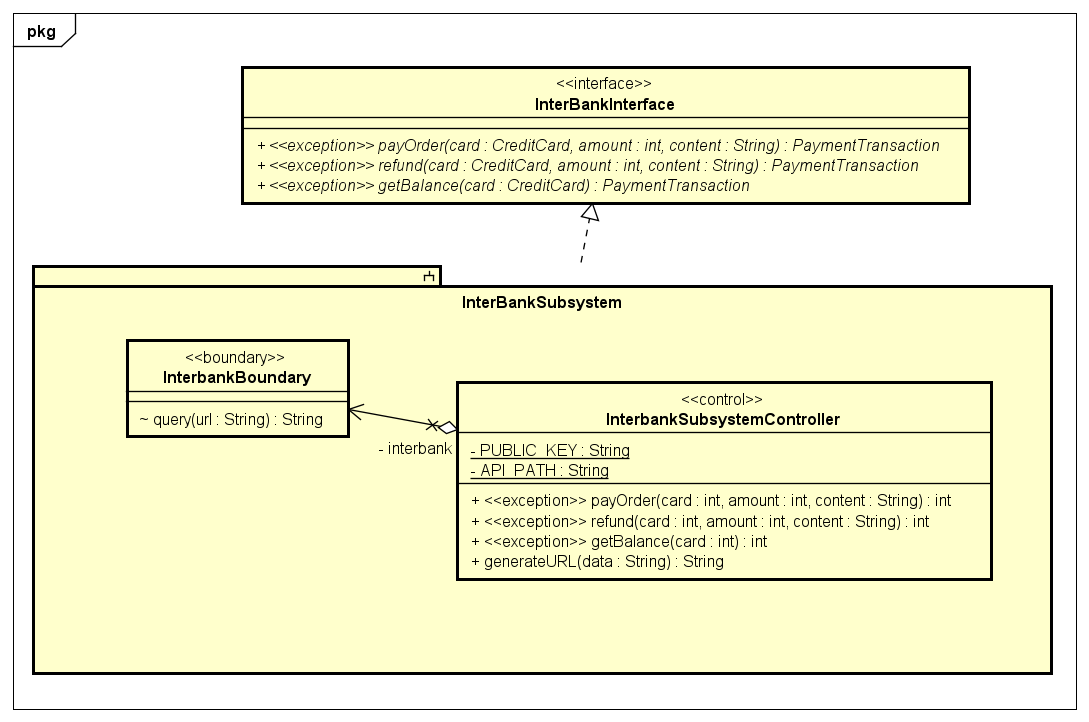
#### Class Diagram for Package Controller



#### Class Diagram for Package View



#### Class Diagram for Subsystem Interbank



### Class Design

#### Class “Bike”

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | int |  | Code of bike |
| 2 | licensePlate | String |  | License plate number of bike |
| 3 | isAvailable | boolean |  | Show if that bike is available for renting |
| 4 | type | String |  | One of following value: “StandardBike”, “StandardEBike”, “TwinBike”, “TwinEBike” |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getBikeInDock | Bike | Find detail information of bike when knowing its dockId and its No. of cell |
| 2 | getBikeById | Bike | Find detail information of bike when knowing its code |

*Parameter*:

* dockId: id of Dock
* cell: No. of cell
* id: id of Bike

*Exception*:

* EntityNotFoundException: get data of an entity but not exist data

**Method:** Không

**State:** Không

#### Class “StandardBike”

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Super Class**

Bike

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | PRICE | int | 400000 | Price of bike |
| 2 | MULTIPLE | double | 1.0 | The multiple of rental fee when comparing to the fee of renting standard bike |

**Operation**

Không

*Parameter*:

Không

*Exception*:

* EntityNotFoundException: get data of an entity but not exist data

**Method:** Không

**State:** Không

#### Class “TwinBike”

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Super Class**

Bike

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | PRICE | int | 550000 | Price of bike |
| 2 | MULTIPLE | double | 1.5 | The multiple of rental fee when comparing to the fee of renting standard bike |

**Operation**

Không

*Parameter*:

Không

*Exception*:

* EntityNotFoundException: get data of an entity but not exist data

**Method:** Không

**State:** Không

#### Class “StandardEBike”

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Super Class**

Bike

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | remainBattery | int | 0 | The remaining percentage of bike battery |
| 2 | PRICE | int | 700000 | Price of bike |
| 3 | MULTIPLE | float | 1.5 | The multiple of rental fee when comparing to the fee of renting standard bike |

**Operation**

Không

*Parameter*:

Không

*Exception*:

* EntityNotFoundException: get data of an entity but not exist data

**Method:** Không

**State:** Không

#### Class “BikeInfoScreenHandler”

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | display | void | Display BikeInfoScreen |
| 2 | handleChooseRentOption | void | Display suitable form right after user chooses rent option |
|  | handleRentBikeBtnClick | void | handle after user clicks “Rent Bike” button |

*Parameter*:

bikeId: code of bike

*Exception*: Không

**Method:** Không

**State:** Không

#### Class “PaymentFormHandler”

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | display | void | Display PaymentForm |
| 2 | handleProcessPaymentBtnClick | void | handle after user clicks “Process Payment” button |
| 3 | notifyError | void | Display error notification if information submitted is invalid |

*Parameter*:

mode: either “pay deposite” or “pay rental fee”

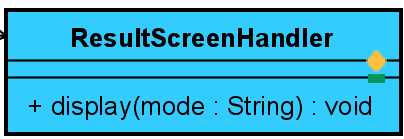
message: error detail

*Exception*: Không

**Method:** Không

**State:** Không

#### Class “ResultScreenHandler”



**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | display | void | Display ResultScreen notifying user the result got from InterBank |

*Parameter*:

mode: Either “success” or “fail”

*Exception*: Không

**Method:** Không

**State:** Không

#### Class “Invoice”

Ảnh có chứa bàn

Mô tả được tạo tự động

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | content | String |  | Have format: “The <rental fee/ deposit> for renting <bike type> <bike code> at <dock Id>” |
| 2 | amount | int |  | The amount of money for transfering |
| 3 | rentInfo | RentInfo |  | Store information of rental turn |

**Operation**

Không

*Parameter*:

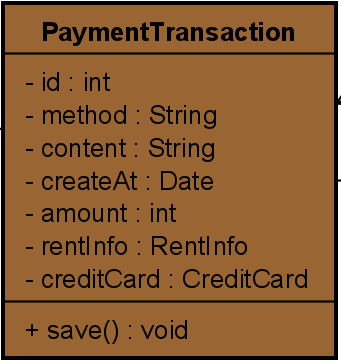
Không

*Exception*: Không

**Method:** Không

**State:** Không

#### Class “PaymentTransaction”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | int |  | Id of transaction |
| 2 | method | String | “credit card” | Current system has only one payment method: “credit card” |
| 3 | content | String |  | Have format: “The <rental fee/ deposit> for renting <bike type> <bike code> at <dock Id>” |
| 4 | amount | int |  | The amount of money for transfering |
| 5 | rentInfo | RentInfo |  | Store information of rental turn |
| 6 | creditCard | CreditCard |  | Detail information of credit card used for payment |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | save | void | Save its object in database |

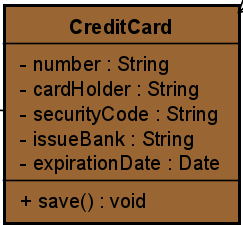
*Parameter*: Không

*Exception*: Không

**Method:** Không

**State:** Không

#### Class “CreditCard”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | number | String |  | Number of card |
| 2 | cardHolder | String |  | Name of card holder |
| 3 | securityCode | String |  | Security code of card |
| 4 | issueBank | String |  | The bank issues this card |
| 5 | expirationDate | String |  | Four digits show date and month when card expires |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | save | void | Save its object in database |

*Parameter*:

Không

*Exception*: Không

**Method:** Không

**State:** Không

* + - 1. ***Class “PaymentController”***

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | processPayment | PaymentTransaction | Forward payment request to InterBank and receive its result as payment transaction |
| 2 | validatePaymentForm | int | Validate input entered in payment form |

*Parameter*:

* Invoice: the invoice containing payment content
* creditCard: card used for payment
* info: card information submited by user

*Exception*:

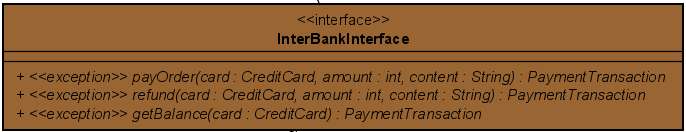
* InvalidFormInputException
* PaymentException
* UnrecognizedException

**Method**

* Không

**State:** Không

* + - 1. ***Class “InterbankInterface”***

****

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | payOrder | PaymentTransaction | Pay order and return payment transaction |
| 2 | refund | PaymentTransaction | transfer money to the account of credit card |
| 3 | getBalance | PaymentTransaction | Get the balance in account |

*Parameter*:

* amount: money amount for transfering
* creditCard: card used for transaction
* content: Have format: “The <rental fee/ deposit> for renting <bike type> <bike code> at <dock Id>”

*Exception*:

* PaymentException
* UnrecognizedException

**Method:** Không

**State:** Không

* + - 1. ***Class “HomeScreenHandler”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute:** Không

**Operation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return data type** | **Description** |
| 1. | display | void | Display home screen – list dock |
| 2. | handleBikeSearchBtnClick | void | Click search bike button |
| 3. | handleDockSearchBtnClick | void | Click search dock button |
| 4. | displayPage | void | Display list dock in a page |
| 5. | handleViewDockBtnClick | void | Click view dock button |

**Parameter:**

* id – id of dock
* page – page number

**Exception:** No

**Method:** No

**State:** No

* + - 1. ***Class “DockViewScreenHandler”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute:** Không

**Operation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return data type** | **Description** |
| 1. | display | void | Display dock detail screen |
| 2. | handleViewBikeBtnClick | void | Click view bike button |

**Parameter:**

* dockId – id of dock
* cell – number of a cell in dock
* page – page number

**Exception:** No

**Method:** No

**State:** No

* + - 1. ***Class “SearchBikeController”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute:** Không

**Operation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return data type** | **Description** |
| 1. | getDockById | Dock | Get data of a dock by id |
| 2. | getDockByName | List<Dock> | Get data of list dock by name |
| 3. | getAllDock | List<Dock> | Get data of all dock |
| 4. | getBikeInDock | Bike | Get data of a cell in dock |
| 5. | getBikeById | Bike | Get data of a bike by id |

**Parameter:**

* id (getDockById), dockId – id of dock
* name – name of dock
* cell – number of a cell in dock
* id (getBikeById) – id of bike

**Exception:**

* EntityNotFoundException: get data of an entity but not found data

**Method:** No

**State:** No

* + - 1. ***Class “Dock”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Data type** | **Description** |
| 1. | id | int | id of dock |
| 2. | name | String | name of dock |
| 3. | numCell | int | number of cells in dock |
| 4. | area | float | area of dock |
| 5. | address | String | address of dock |
| 6. | listCell | List<Cell> | list cell in dock |

**Operation:** No

**Parameter:** No

**Exception:**

* EntityNotFoundException: get data of an entity but not exist data

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return data type** | **Description** |
| 1. | getDockById | Dock | Get data of a dock by id |
| 2. | getDockByName | List<Dock> | Get data of list dock by name |
| 3. | getAllDock | List<Dock> | Get data of all dock |
| 4. | getEmptyCellInDock | List<Cell> | Get data of list empty cell in dock |
| 5. | getNotFullDock | List<Dock> | Get data of list dock not full |

**State:** No

* + - 1. ***Class “RentedBikeInfoScreenHandler”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | display | void | Display the information of the bike which was rented |
| 2 | handleReturnBikeBtnClick | void | Transfer toReturnBikeScreen |
| 3 | notifyError | void | Notify error |

*Parameter*:

* Message: the message error

*Exception*:

* không

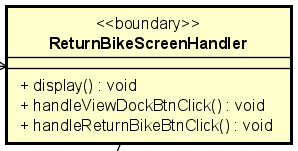
**Method**

Không

**State**

Không

* + - 1. ***Class “ReturnBikeScreenHandler”***



**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | display | void | Display info in screen |
| 2 | handleViewDockBtnClick | void | Process when user choose a dock |
| 3 | handleReturnBikeBtnClick | void | Process when user choose a cell |

*Parameter*:

*Exception*:

**Method**

không

**State:** Không

* + - 1. ***Class “RentBikeController”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | createRentInfo | void | Create rentInfo |
| 2 | getRentInfo | RentInfo | Get RentInfo for display |
| 3 | calculateDepositAmount | int | Get deposite for the bike |

*Parameter*:

* bike: bike which was rented

*Exception*:

* không

**Method**

Không

**State**

Không

* + - 1. ***Class “ReturnBikeController”***

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | rentInfo | RentInfo |  | RentInfo |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | returnBike | void | Process return bike |
| 2 | getEmptyCellInDock | List<Cell> | Get list of empty cell to return bike |
| 3 | getNotFullDock | List<Dock> | Get list of dock which has empty cell |
| 4 | calculateRentAmount | int | Calculate rent amount |

*Parameter*:

* rent
* dock: dock which was chosen

*Exception*:

* không

**Method**

Không

**State**

Không

* + - 1. ***Class “RentInfo”***

Ảnh có chứa bàn

Mô tả được tạo tự động

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | int |  | Id of rentinfo |
| 2 | startTime | Date |  | Time when user start renting bike |
| 3 | endTime | Date |  | End rent bike |
| 4 | rentType | String |  | Type of rent |
| 5 | rentedPeriod | int |  | How long user rent bike |
| 6 | depopsitAmount | int |  | The deposite to rent bike |
| 7 | bike | Bike |  | Bike was rented |
| 8 | isComplete | boolean |  | True If user returned bike |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | save | void | Save rentInfo into database |
| 2 | getRentInfo | RentInfo | Get rentInfo which was not complete |
| 3 | update | void | Update RentInfo |

*Parameter*:

* object: rentInfo need to update

*Exception*:

* EntityNotFoundException: get data of an entity but not found data

**Method**

Không

**State**

Không

# Design Considerations

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

## Goals and Guidelines

*<Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system and its software.*

*Examples of such goals might be: an emphasis on speed versus memory use; or working, looking, or “feeling” like an existing product.*

*Guidelines include coding guidelines and conventions.*

*For each such goal or guideline, describe the reason for its desirability unless it is implicitly obvious.*

*Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system (e.g., choice of which specific product to use)*>

## Architectural Strategies

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

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

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

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

*• Future plans for extending or enhancing the software*

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

*• Hardware and/or software interface paradigms*

*• Error detection and recovery*

*• Memory management policies*

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

*• Distributed data or control over a network*

*• Generalized approaches to control*

*• Concurrency and synchronization*

*• Communication mechanisms*

*• Management of other resources*

>

## Coupling and Cohesion

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

## Design Principles

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

## Design Patterns

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