

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Tài liệu mô tả thiết kế phần mềm

Đề tài: Hệ thống thuê xe EcobikeRental

Môn: Thiết kế và xây dựng phần mềm

Giảng viên hướng dẫn: TS.Nguyễn Thị Thu Trang

Nhóm 1

Lê Minh Tú – 20183651

Nguyễn Tiến Mạnh - 20180132

Đào Minh Dũng - 20183503

Hà Nội, 01/2022

Mục lục

1	Giới thiệu	4
1.1	Mục đích.....	4
1.2	Phạm vi.....	4
1.3	Từ điển thuật ngữ	5
1.4	Tài liệu tham khảo.....	5
2	Mô tả tổng quan	6
2.1	Mô tả chung	6
2.2	Giả định/ ràng buộc/ rủi ro	6
2.2.1	Giả định	6
2.2.2	Ràng buộc.....	6
2.2.3	Rủi ro.....	6
3	Hệ thống kiến trúc và thiết kế kiến trúc	7
3.1	Mô hình kiến trúc (Architectural Patterns).....	7
3.2	Biểu đồ tương tác.....	9
3.2.1	Biểu đồ trình tự Xem bãi và xem xe.....	9
3.2.2	Biểu đồ trình tự Chuyển Barcode	10
	10
3.2.3	Biểu đồ trình tự Thuê xe	10
3.2.4	Biểu đồ trình tự Trả xe	11
	11
3.3	Biểu đồ lớp phân tích	11
3.3.1	Biểu đồ lớp Xem bãi xe và xem xe	11
3.3.2	Biểu đồ lớp Chuyển Barcode	12
3.3.3	Biểu đồ lớp Thuê xe	12

3.3.4	Biểu đồ lớp Trả xe	13
3.4	Biểu đồ lớp phân tích kết hợp.....	14
3.5	Kiến trúc bảo mật phần mềm.....	14
4	Thiết kế chi tiết	15
4.1	Thiết kế giao diện người dùng	15
4.1.1	Chuẩn hóa cấu hình màn hình	15
4.1.2	Sơ đồ dịch chuyển màn hình	16
4.1.3	Đặc tả màn hình	17
4.2	Mô hình hóa dữ liệu	26
4.2.1	Mô hình hóa dữ liệu khái niệm.....	26
4.2.2	Thiết kế Cơ sở dữ liệu	26
4.3	Thiết kế lớp.....	33
4.3.1	Thiết kế lớp tổng quát.....	33
4.3.2	Biểu đồ lớp.....	34
4.3.3	Thiết kế lớp	38
5	Đánh giá thiết kế	57
5.1	Mục tiêu và định hướng.....	57
5.2	Chiến lược kiến trúc	57
5.3	Coupling và Cohesion	57
5.3.1	Coupling	57
5.3.2	Cohesion	58
5.4	Nguyên lý thiết kế	61
5.4.1	Single Responsibility	61
5.4.2	Open/Closed	61
5.4.3	Liskov Substitution	61
5.4.4	Interface Segregation.....	61
5.4.5	Dependency Inversion	61

5.5	Mẫu thiết kế Design pattern	62
5.5.1	Strategy cho chức năng tính phí trả xe của các loại xe.....	62
5.5.2	Factory	63

1 Giới thiệu

1.1 Mục đích

Tài liệu này đưa ra mô tả chi tiết cho nhóm người dùng và các chức năng người dùng có thể sử dụng được cũng như các hệ thống ngoài (ứng dụng, ngân hàng) và chức năng chúng cung cấp cho người sử dụng. Tài liệu cũng mô tả mục đích và các tính năng của hệ thống, các giao diện và ràng buộc của hệ thống cần thực hiện để phản ứng với các tác động bên ngoài.

Tài liệu dành cho các bên liên quan (stakeholder) và các nhà phát triển phần mềm.

1.2 Phạm vi

Mục đích của ứng dụng là cho phép người sử dụng có thể thực hiện các công việc xem thông tin xe, thuê xe, trả xe và thanh toán xe trong khu đô thị. Để có thể sử dụng phần mềm, đầu tiên người dùng cần phải đăng ký một tài khoản, chứa các thông tin cơ bản như mã người dùng, tên người dùng và mã thẻ, sử dụng trong quá trình thanh toán.

Người dùng có thể sử dụng ứng dụng để xem thông tin về các bãi đỗ xe (địa chỉ, diện tích, số lượng xe...) cũng như thông tin về các xe đang đỗ trong bãi đó (loại xe, trạng thái thuê xe, thời gian sử dụng...). Để thuê xe, phần mềm cung cấp chức năng quét mã vạch của xe; sau khi quét, phần mềm sẽ cung cấp thông tin về chiếc xe đó. Người sử dụng sẽ phải đặt cọc lượng tiền tương ứng với thông tin được quét từ mã vạch của xe để hoàn tất thủ tục và được hệ thống cho phép thuê xe. Khi trả xe, người dùng sẽ cho xe vào bãi và khóa lại, sau đó vào hệ thống để xem chi tiết thông tin thanh toán, bao gồm tiền đã đặt cọc, phí thuê xe và số tiền còn thiếu/được hoàn lại. Sau đó hệ thống sẽ thực hiện thanh toán số tiền còn thiếu/cần hoàn lại, và lưu lại lịch sử thuê xe.

Để đơn giản trong quá trình thiết kế, người dùng sẽ chỉ trả tiền thông qua đúng 1 thẻ tín dụng, và tại một thời điểm sẽ chỉ thuê 1 xe.

1.3 Từ điển thuật ngữ

Thuật ngữ	Chú giải
Thẻ tín dụng	Là thẻ cho phép chủ thẻ thực hiện giao dịch thẻ trong phạm vi hạn mức tín dụng đã được cấp theo thỏa thuận với tổ chức phát hành thẻ, hay đơn giản là mua hàng trước và thanh toán lại cho ngân hàng sau. [1]

1.4 Tài liệu tham khảo

[1] Cách sử dụng thẻ tín dụng, HSBC

<https://www.hsbc.com.vn/credit-cards/how-do-credit-cards-work/>

2 Mô tả tổng quan

2.1 Mô tả chung

Phần mềm được thiết kế theo kiến trúc 3-layers, sử dụng trên máy tính. Thiết kế này không thích hợp với người dùng trong thực tế (vì không ai mang PC hay laptop để đi thuê xe đạp, đồng thời chúng cũng không có chức năng quét barcode và được kết nối mạng liên tục như smartphone). Trong khuôn khổ của project này, phần mềm được thiết kế để phù hợp cho việc demo

2.2 Giả định/ ràng buộc/ rủi ro

2.2.1 Giả định

- Đã cài đặt hệ thống và chạy được trên máy tính
- Cần xây dựng trước database trong cơ sở dữ liệu MySQL
- Có thể sử dụng script SQL trong thư mục DataModeling để khởi tạo cơ sở dữ liệu (Đã trình bày trong SDD)

2.2.2 Ràng buộc

- Cần có kết nối mạng
- Cần có thông tin thẻ hợp lệ

2.2.3 Rủi ro

Không có

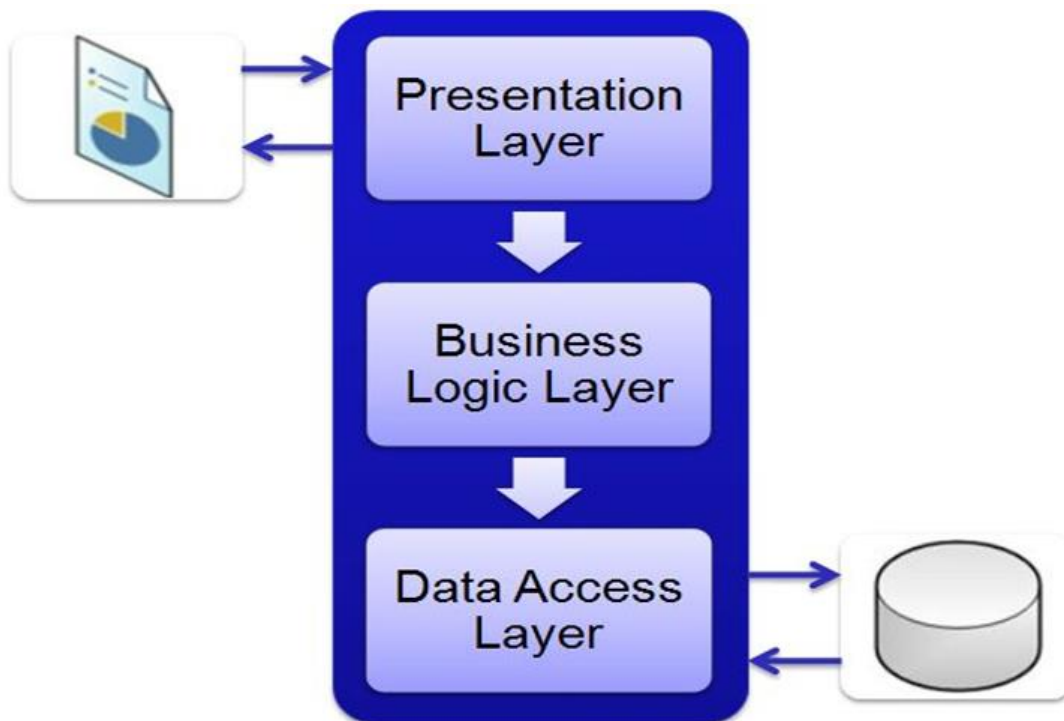
3 Hệ thống kiến trúc và thiết kế kiến trúc

3.1 Mô hình kiến trúc (Architectural Patterns)

Nhóm lựa chọn kiến trúc 3-layers cho việc thiết kế và xây dựng ứng dụng này.

Chức năng của từng tầng:

- *Presentation Layers*: Lớp này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả/dữ liệu thông qua các thành phần trong giao diện người sử dụng.
- *Business Logic Layers*: Đây là layer xử lý chính các dữ liệu trước khi được đưa lên hiển thị trên màn hình hoặc xử lý các dữ liệu trước khi chuyển xuống Data Access Layer để lưu dữ liệu xuống cơ sở dữ liệu. Đây cũng là nơi để kiểm tra ràng buộc, các yêu cầu nghiệp vụ, tính toán, xử lý các yêu cầu và lựa chọn kết quả trả về cho Presentation Layers.
- *Data Access Layers*: Layer này thực hiện các nghiệp vụ liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng như đọc, lưu, cập nhật cơ sở dữ liệu.

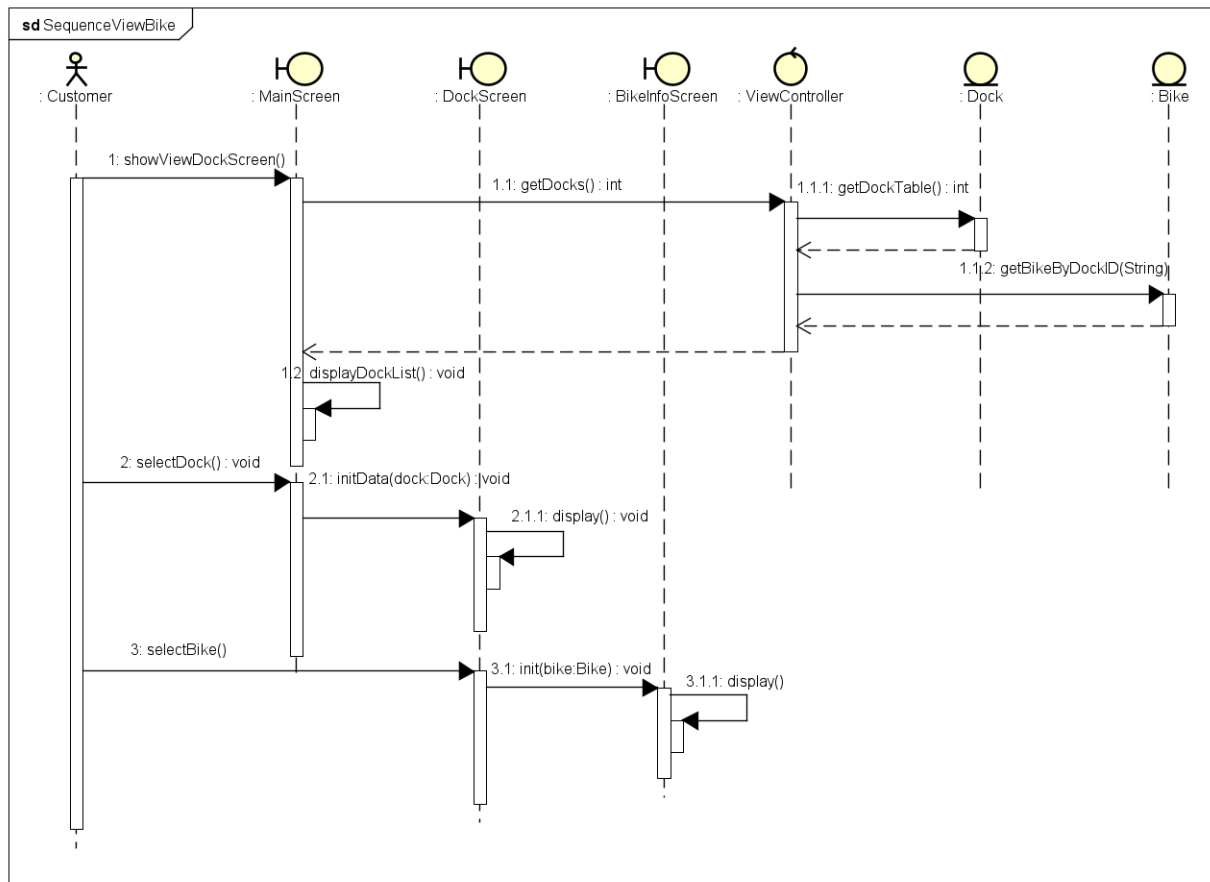


* Quy trình vận hành của mô hình:

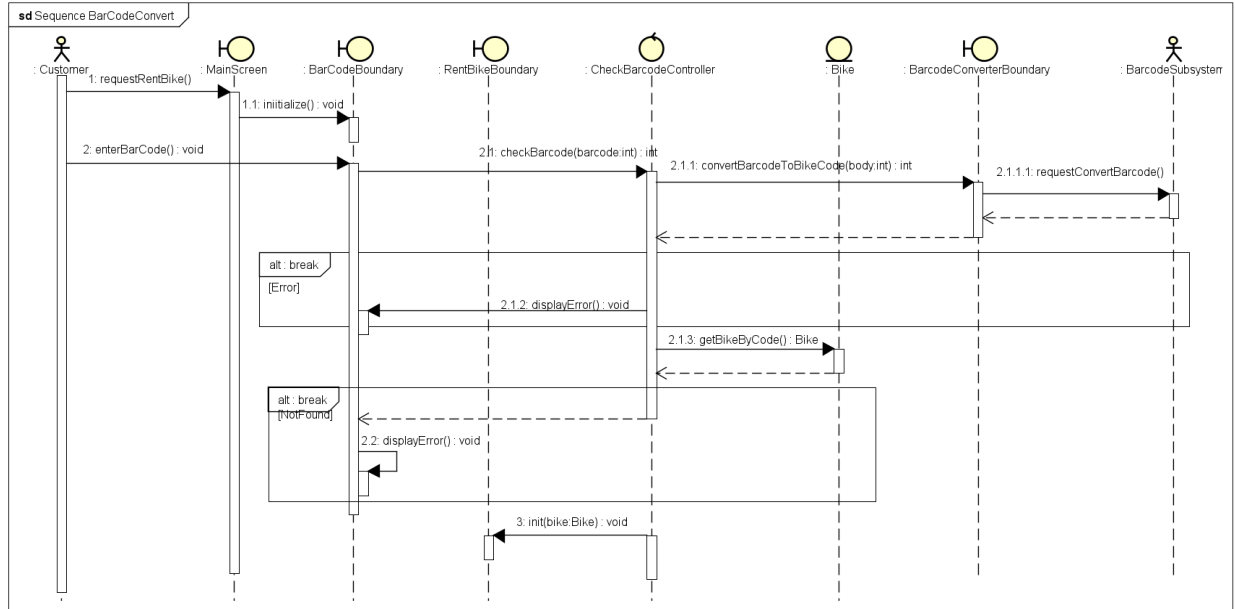
- Đầu tiên, người dùng giao tiếp với lớp *presentation layer*, sau khi tiếp xúc với giao diện, người dùng gửi đi các yêu cầu cho hệ thống và kèm theo là các thông tin. Các thông tin này sẽ được ở xử lý, kiểm tra, sau đó được gửi kèm theo yêu cầu tới lớp thứ 2 là *Business Logic Layer*.
- Tại lớp *Business Logic Layer*, các thông tin sẽ được xử lý theo yêu cầu của người dùng. Kết quả sẽ được trả về và hiển thị cho người dùng. Nếu dữ liệu cần thiết phải truy cập database thì dữ liệu sẽ được chuyển xuống lớp thứ 3 là *Data Access Layer*.
- Tại lớp *Data Access Layer*, hệ thống sẽ thực hiện các thao tác với database có thể là lưu trữ, lấy thông tin,...và chuyển lại lên tầng *Business Logic Layer*. *Business Logic Layer* kiểm tra và gửi nó lên *Presentation Layer* để hiển thị cho người dùng.

3.2 Biểu đồ tương tác

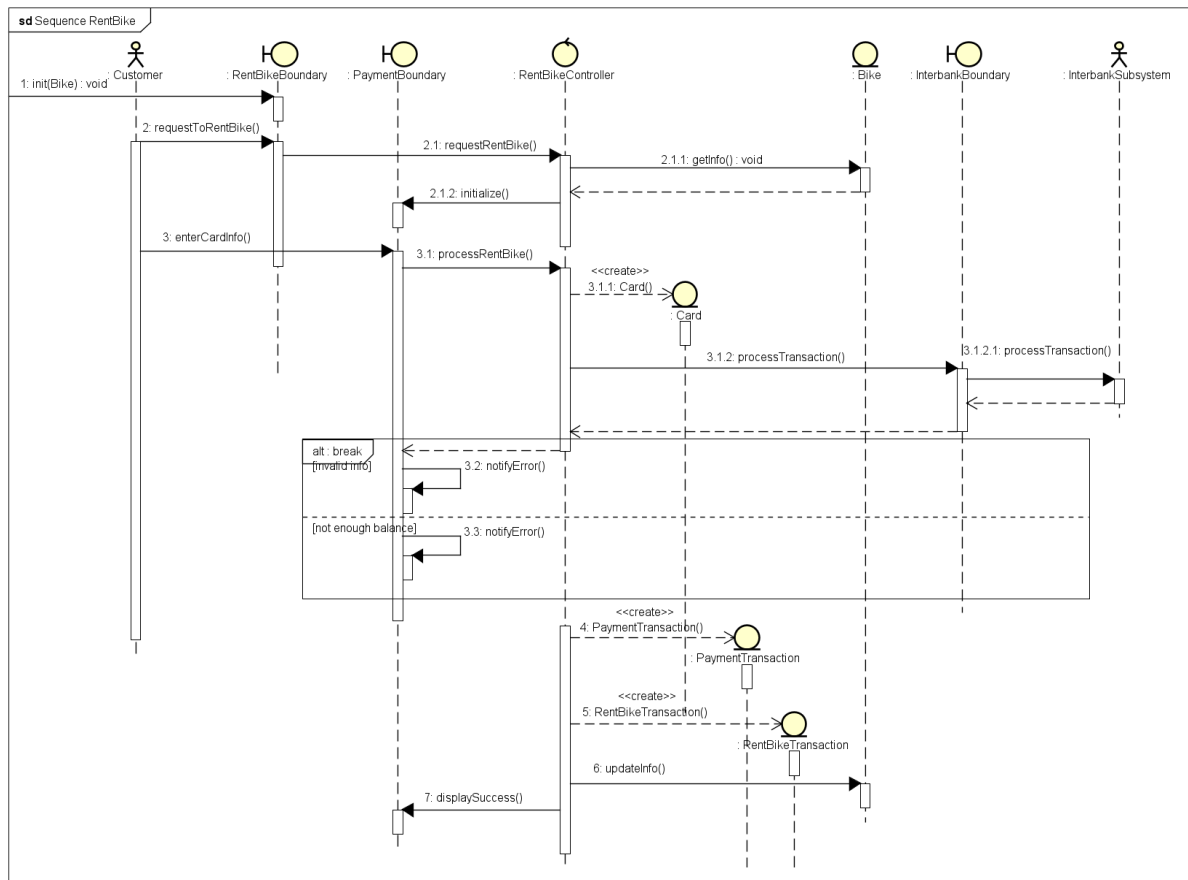
3.2.1 Biểu đồ trình tự Xem bãi và xem xe



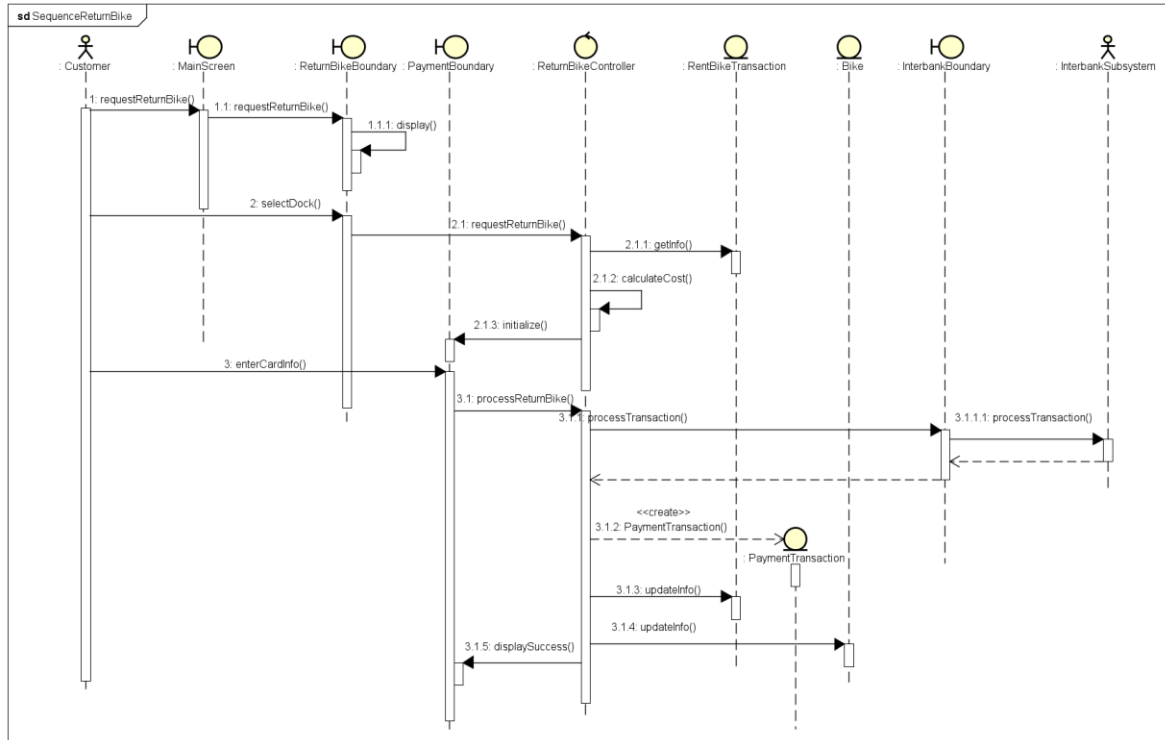
3.2.2 Biểu đồ trình tự Chuyển Barcode



3.2.3 Biểu đồ trình tự Thuê xe

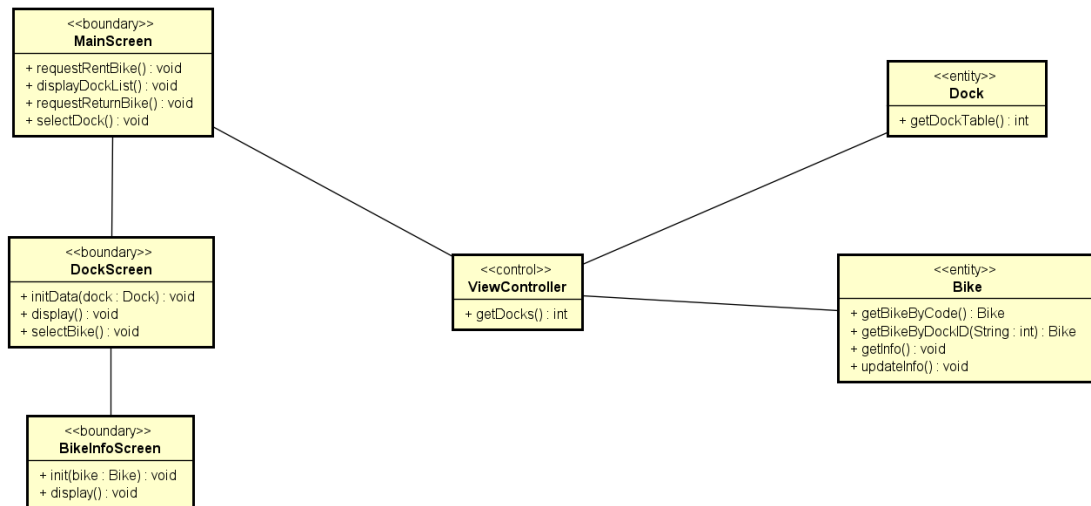


3.2.4 Biểu đồ trình tự Trả xe

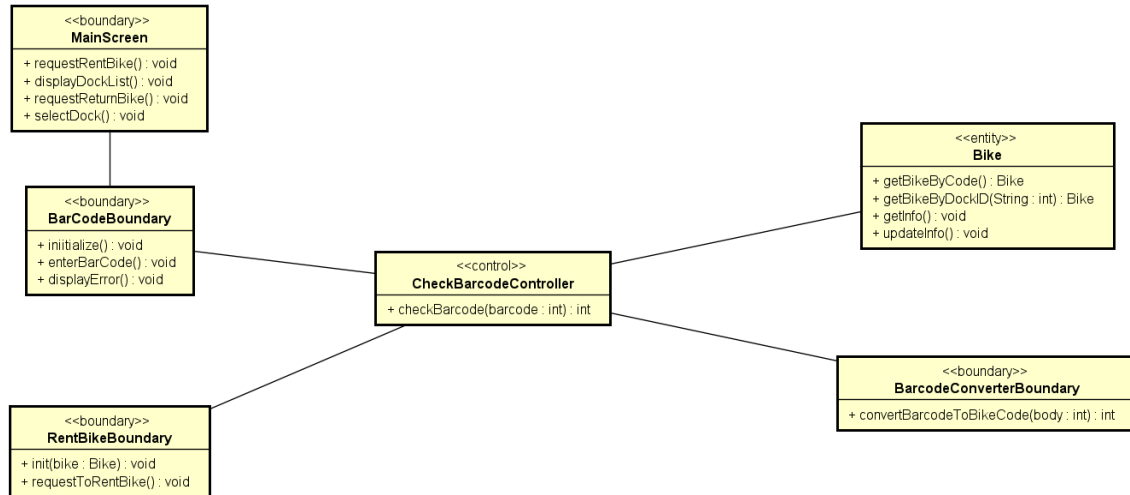


3.3 Biểu đồ lớp phân tích

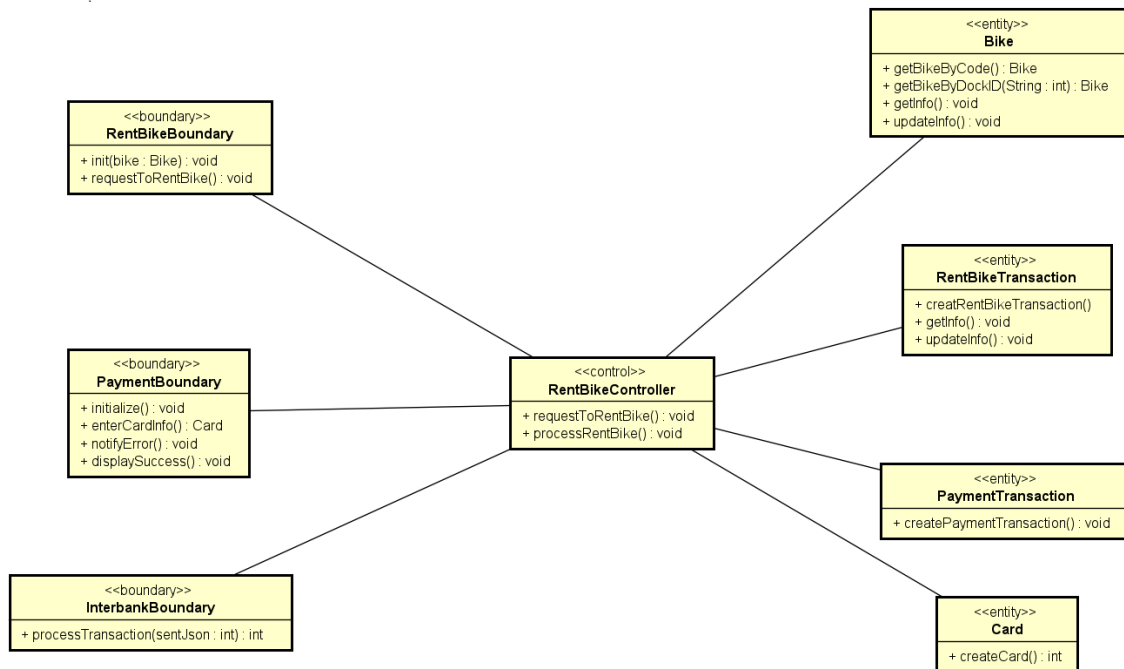
3.3.1 Biểu đồ lớp Xem bãi xe và xem xe



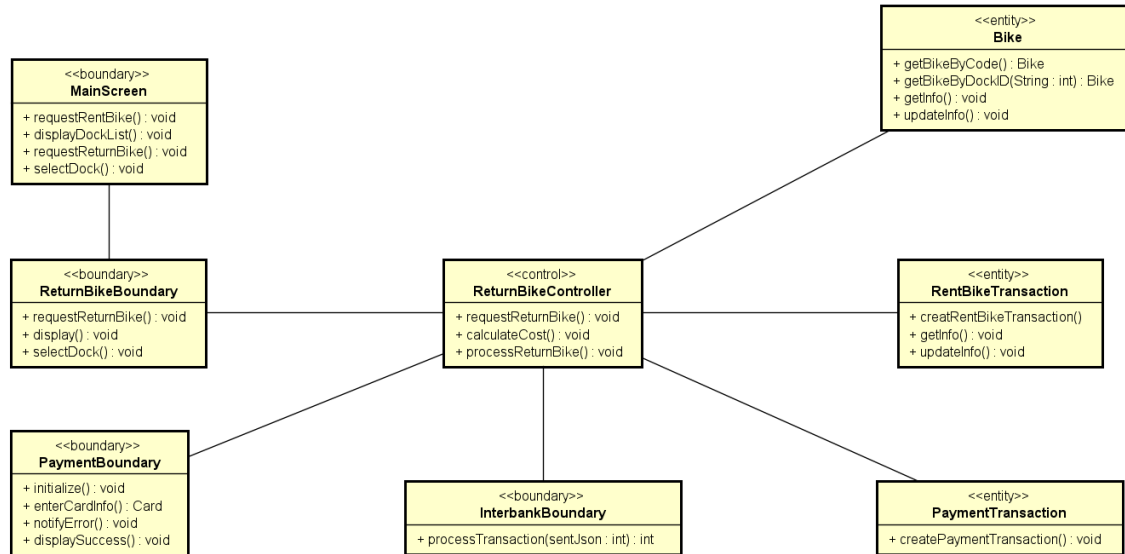
3.3.2 Biểu đồ lớp Chuyển Barcode



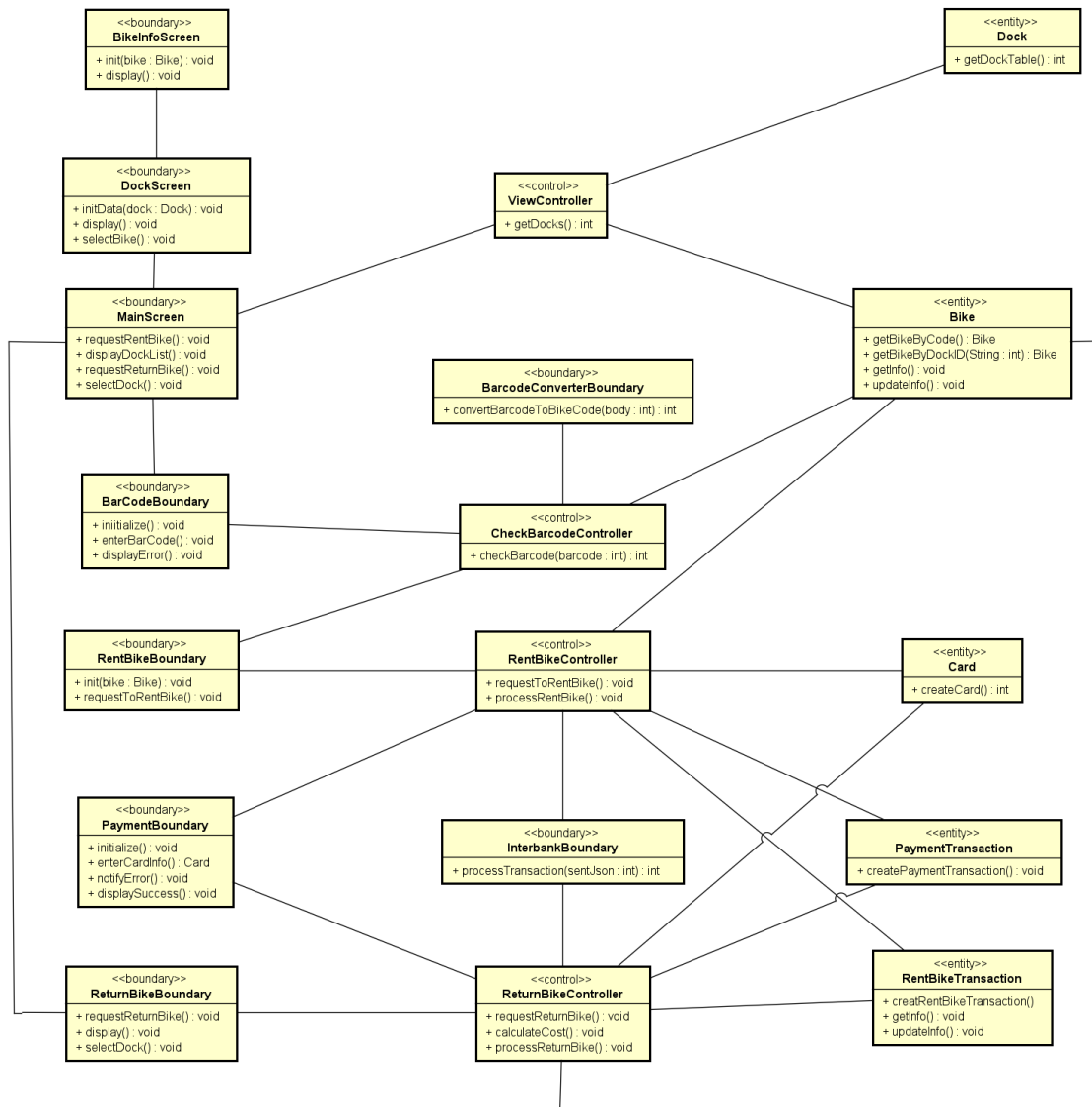
3.3.3 Biểu đồ lớp Thuê xe



3.3.4 Biểu đồ lớp Trả xe



3.4 Biểu đồ lớp phân tích kết hợp



3.5 Kiến trúc bảo mật phần mềm

<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.>

4 Thiết kế chi tiết

4.1 Thiết kế giao diện người dùng

4.1.1 Chuẩn hóa cấu hình màn hình

Chuẩn hóa cấu hình màn hình

Display:

- Số màu hỗ trợ: 16,777,216 màu
- Độ phân giải màn hình: 900x600 pixel

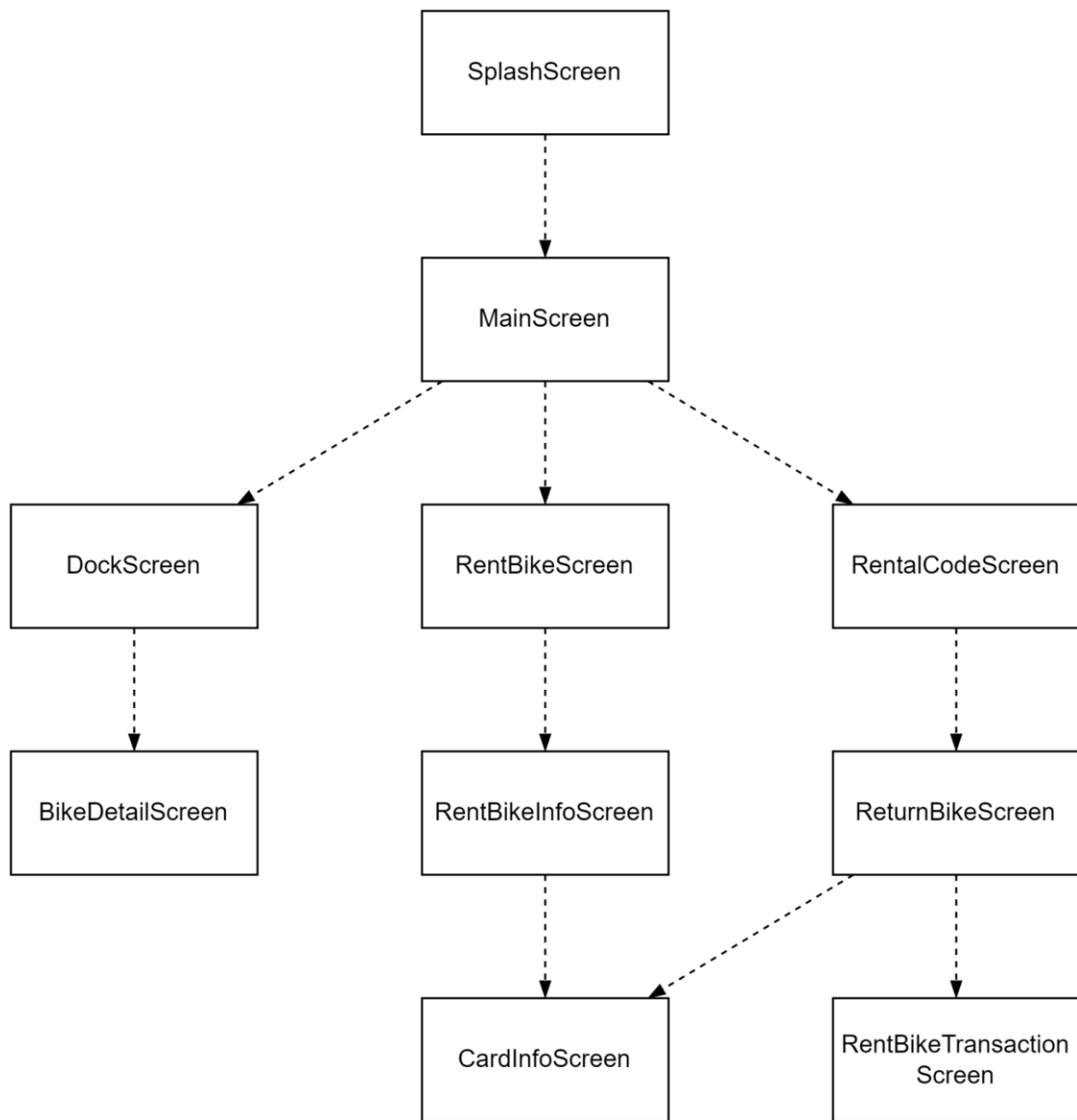
Screen:

- Vị trí của button: Ở dưới cùng theo chiều dọc và ở giữa theo chiều ngang
- Vị trí message: ở vị trí trung tâm màn hình
- Vị trí screen title: ở góc trên bên trái màn hình cùng với logo
- Màu chủ đề của ứng dụng là xanh lá : #01a13e

Text:

- Font: System
- Cỡ chữ: screen title:36px, text:24px
- Màu: title theo màu chủ đề, còn text thường là đen: #000000

4.1.2 Sơ đồ dịch chuyển màn hình

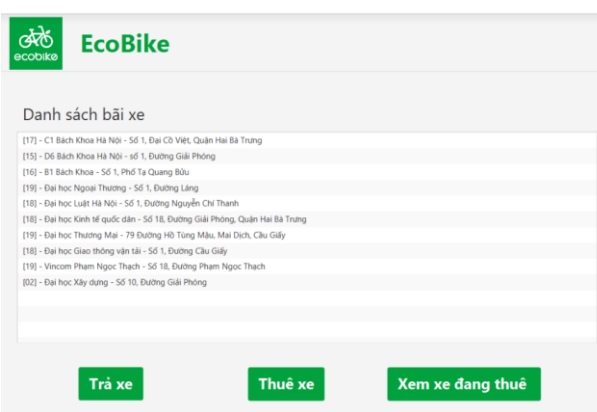


4.1.3 Đặc tả màn hình

4.1.3.1 Màn hình chờ Main



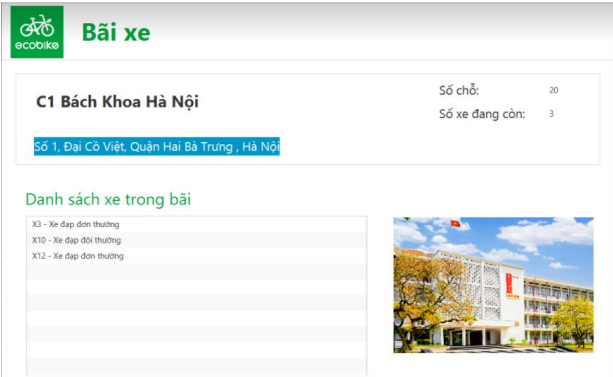
4.1.3.2 Màn hình chính

EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	MainScreen				Lê Minh Tú
		Control	Operation	Function	
		Vùng hiển thị danh sách bãi xe	Khởi tạo	Hiện thị bãi xe để người dùng chọn xem	
		Vùng hiển thị danh sách bãi xe	Double click	Xem chi tiết bãi xe	
		Nút Trả xe	click	Người dùng muốn trả xe	
		Nút Thuê xe	click	Người dùng muốn thuê xe	

	Nút Xem xe đang thuê	click	Người dùng muốn xem thông tin xe đang thuê
--	----------------------	-------	--

Screen name	MainScreen		
Item name	Type	Field attribute	Remarks
Số chỗ trống	Số	Đen	Căn trái
Tên bãi xe	String	Đen	Căn trái
Địa chỉ	String	Đen	Căn trái

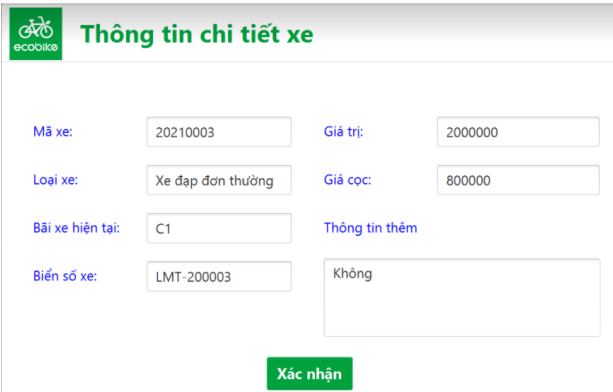
4.1.3.3 Màn hình thông tin bãi xe

EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	DockScreen				Lê Minh Tú
		Control	Operation	Function	
		Khu vực hiển thị thông tin bãi xe	Khởi tạo	Hiển thị thông tin bãi xe	
		Danh sách xe trong bãi	Khởi tạo	Hiển thị các xe có trong bãi	
		Danh sách xe trong bãi	Double click	Xem thông tin xe	

Screen name	DockScreen		
-------------	------------	--	--

Item name	Type	Field attribute	Remarks
Tên bãi xe	String	Đen	Căn trái
Địa chỉ	String	Đen	Căn trái
Số chỗ để xe	Số	Đen	Căn trái
Số xe đang còn	Số	Đen	Căn trái

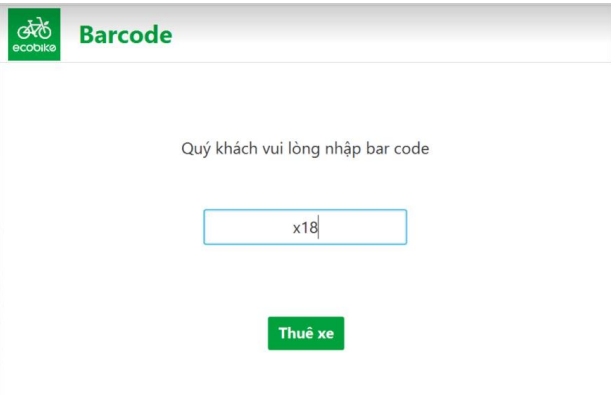
4.1.3.4 Màn hình thông tin xe

EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	BikeDetailScreen				Lê Minh Tú
		Control	Operation	Function	
		Khu vực hiển thị thông tin xe	Khởi tạo	Hiển thị thông tin xe	
		Nút Xác nhận	Click	Quay về trang trước	

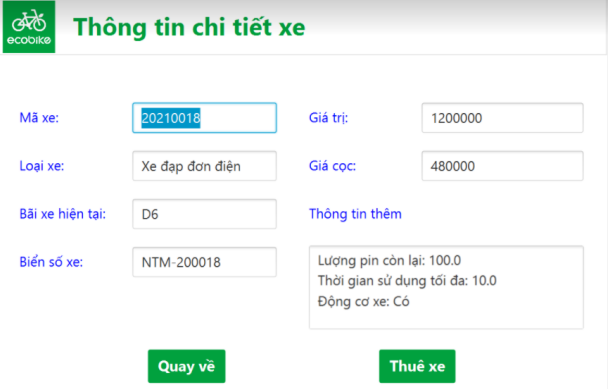
Screen name	BikeDetailScreen		
Item name	Type	Field attribute	Remarks

Mã xe	String	Đen	Căn trái
Loại xe	String	Đen	Căn trái
Bãi xe hiện tại	String	Đen	Căn trái
Biển số xe	String	Đen	Căn trái
Giá trị	Số	Đen	Căn trái
Giá cọc	Số	Đen	Căn trái
Thông tin thêm	String	Đen	Căn trái

4.1.3.5 Màn hình nhập barcode


EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	RentBikeScreen				
	Control	Operation	Function		
	Khu vực nhập barcode	Nhập thông tin	Nhập barcode để thuê xe		
	Nút Thuê xe	Click	Xác nhận mã barcode		

4.1.3.6 Màn hình thuê xe

EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	RentBikeInfo Screen				
		Control	Operation	Function	
		Khu vực hiển thị thông tin xe	Khởi tạo	Hiển thị thông tin xe	
		Nút Xác nhận	Click	Quay về trang trước	
		Nút Quay về	Click	Hủy thuê xe	

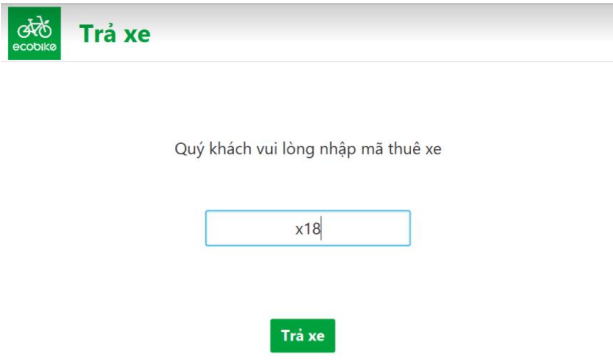
Screen name	BikeDetailScreen		
Item name	Type	Field attribute	Remarks
Mã xe	String	Đen	Căn trái
Loại xe	String	Đen	Căn trái
Bãi xe hiện tại	String	Đen	Căn trái
Biển số xe	String	Đen	Căn trái
Giá trị	Số	Đen	Căn trái
Giá cọc	Số	Đen	Căn trái
Thông tin thêm	String	Đen	Căn trái

4.1.3.7 Màn hình Thanh toán xe


EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	CardInfoScreen				
		Control	Operation	Function	
		Khu vực hiển thị thông tin thẻ	Người dùng nhập	Nhập thông tin thẻ vào	
		Nút Thanh toán	Click	Xác nhận thanh toán	
		Nút Hủy	Click	Hủy bỏ việc thanh toán	

Screen name	CardInfoScreen		
Item name	Type	Field attribute	Remarks
Số thẻ	String	Đen	Căn trái
Chủ thẻ	String	Đen	Căn trái
Mã bảo mật	String	Đen	Căn trái
Ngày hết hạn	String	Đen	Căn trái

4.1.3.8 Màn hình Nhập mã thuê để trả xe

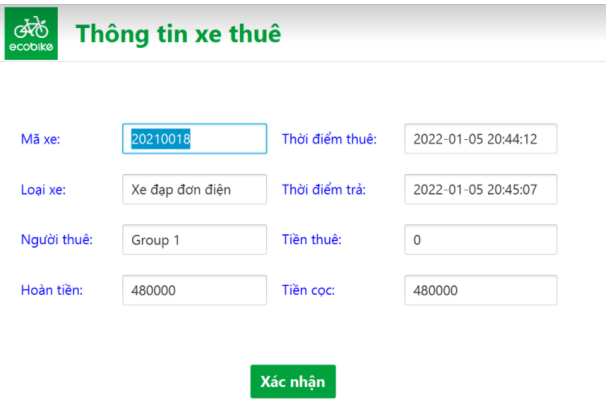
EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	Screen				
		Control	Operation	Function	
		Khu vực nhập mã thuê	Nhập thông tin	Nhập mã thuê để trả xe	
		Nút Trả xe	Click	Xác nhận trả xe	

4.1.3.9 Màn hình Trả xe

EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	ReturnBikeScreen				
		Control	Operation	Function	
		Khu vực hiển thị danh sách bãi xe	Khởi tạo	Hiển thị danh sách bãi xe	
		Khu vực danh sách bãi xe	Double click	Chọn bãi xe để trả	

Screen name	ReturnBikeScreen		
Item name	Type	Field attribute	Remarks
Số chỗ trống	Số	Đen	Căn trái
Tên bãi xe	String	Đen	Căn trái
Địa chỉ	String	Đen	Căn trái

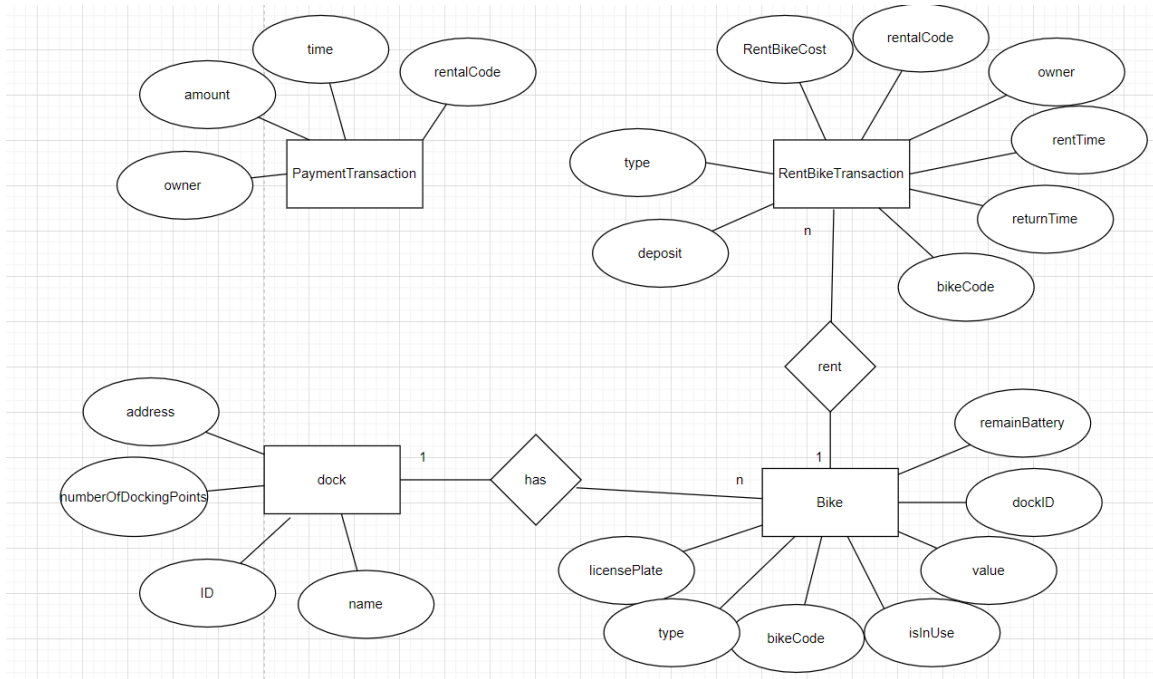
4.1.3.10 Màn hình Thông tin thuê xe

EcoBike		Date of creation	Approved by	Review by	Person in change
Screen specification	RentBike TransactionScreen				
		Control	Operation	Function	
		Khu vực hiển thị thông tin thuê	Khởi tạo	Hiển thị về việc thuê xe	
		Nút xác nhận	Click	Kết thúc	

Screen name	RentBike TransactionScreen		
Item name	Type	Field attribute	Remarks
Mã xe	String	Đen	Căn trái
Loại xe	String	Đen	Căn trái
Người thuê	String	Đen	Căn trái
Hoàn tiền	Số	Đen	Căn trái
Thời điểm thuê	DateTime	Đen	Căn trái
Thời điểm trả	DateTime	Đen	Căn trái
Tiền thuê	Số	Đen	Căn trái
Tiền cọc	Số	Đen	Căn trái

4.2 Mô hình hóa dữ liệu

4.2.1 Mô hình hóa dữ liệu khái niệm

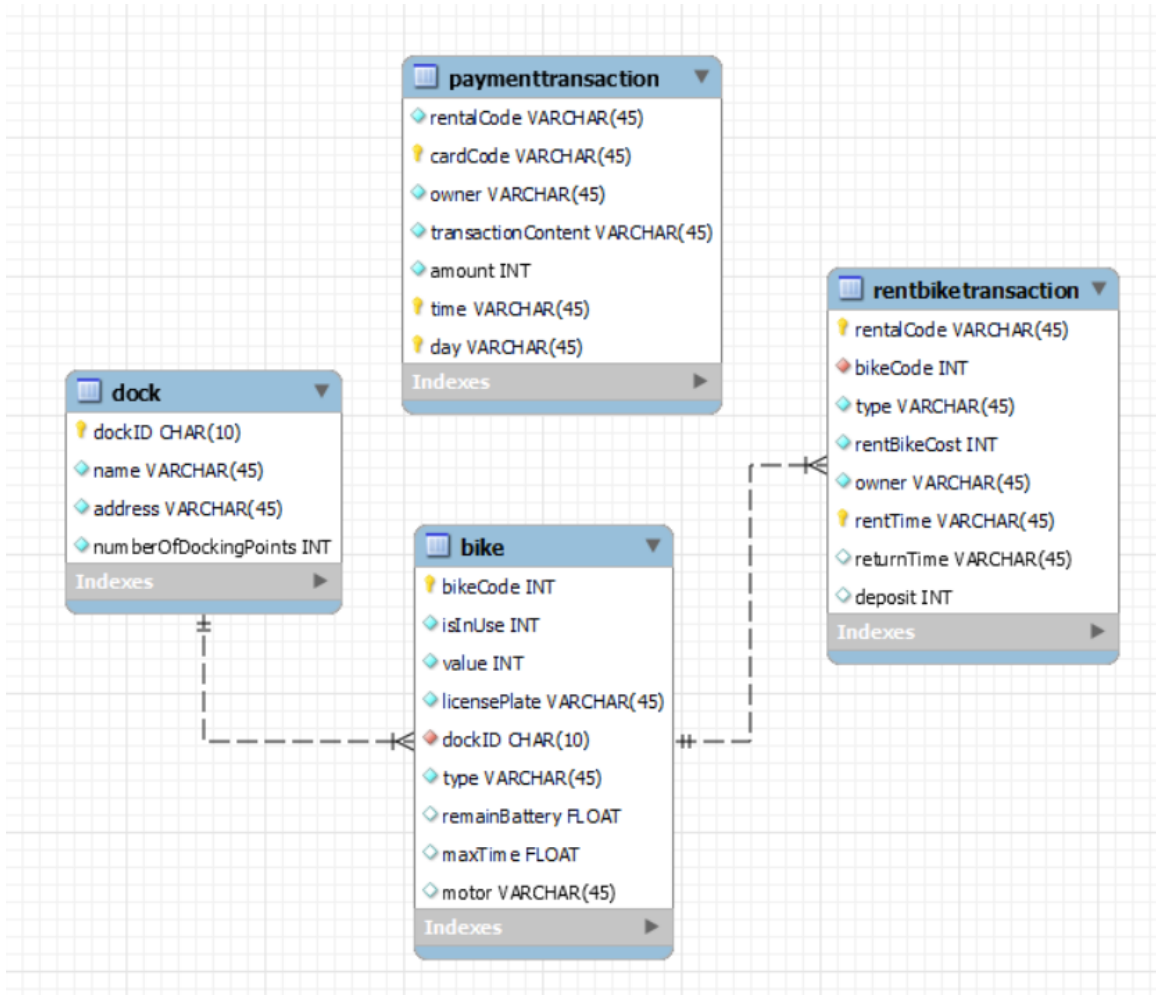


4.2.2 Thiết kế Cơ sở dữ liệu

4.2.2.1 Hệ quản trị cơ sở dữ liệu

Sử dụng CSDL là MySQL, dùng giao diện MySQL WorkBench để dễ làm việc với CSDL

4.2.2.2 Mô hình dữ liệu logic



4.2.2.3 Mô hình dữ liệu vật lý

dock					
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc
1	X		dockID	CHAR(10)	Có
2			name	VARCHAR(45)	Có
3			address	VARCHAR(45)	Có
4			numberOfDockingPoints	INT	Có

bike						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		bikeCode	INT	Có	mã bãi xe
2			isInUse	INT	Có	bằng 1 nếu đang sử dụng và bằng 0 nếu không sử dụng
3			type	VARCHAR(45)	Có	loại xe
4			value	INT	Có	giá xe
5			remainBattery	FLOAT	Không	lượng pin còn lại (với xe điện)
6			maxTime	FLOAT	Không	thời gian sử dụng tối đa (với xe điện)
7			licensePlate	VARCHAR(45)	Có	biển số xe
8		X	dockID	CHAR(10)	Có	vị trí bãi xe của xe hiện tại (chỉ có ý nghĩa với xe đang không được sử dụng)
9			motor	<u>VARCHAR(45)</u>	Không	Thông tin motor (đối với xe điện)

Payment transaction						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1			rentalCode	VARCHAR(45)	Có	mã thuê xe
2	X		cardCode	VARCHAR(45)	Có	mã thẻ
3			owner	VARCHAR(45)	Có	chủ thẻ
4			transactionContent	VARCHAR(45)	Có	nội dung giao dịch
5			amount	INT	Có	lượng tiền giao dịch
6	X		time	VARCHAR(45)	Có	thời gian giao dịch (hh-mm-ss)
7	X		day	VARCHAR(45)	Có	ngày giao dịch (yyyy-MM-dd)

Rent bike transaction						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		rentalCode	VARCHAR(45)	Có	mã thuê xe
2		X	bikeCode	INT	Có	mã xe được thuê
3			type	VARCHAR(45)	Có	loại xe
4			rentBikeCost	INT	Không	chi phí thuê xe (khi chưa trả xe thì đặt là -1)
5			owner	VARCHAR(45)	Có	người thuê
6	X		rentTime	VARCHAR(45)	Có	thời điểm thuê
7			returnTime	VARCHAR(45)	Không	thời điểm trả (khi chưa trả thì đặt là null)
8			deposit	INT	Có	tiền đặt cọc

SQL Script:

```
CREATE SCHEMA IF NOT EXISTS `ecobikedatabase` DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci ;
```

```
USE `ecobikedatabase` ;
```

```
-----
```

```
-- Table `ecobikedatabase`.`dock`
```

```
-----
```

```
DROP TABLE IF EXISTS `ecobikedatabase`.`dock` ;
```

```
CREATE TABLE IF NOT EXISTS `ecobikedatabase`.`dock` (
```

```
  `dockID` CHAR(10) NOT NULL,
```

```
  `name` VARCHAR(45) NOT NULL,
```

```
  `address` VARCHAR(45) NOT NULL,
```

```
  `numberOfDockingPoints` INT(11) NOT NULL,
```

```
  PRIMARY KEY (`dockID`))
```

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

-- Table `ecobikedatabase`.`bike`

DROP TABLE IF EXISTS `ecobikedatabase`.`bike` ;

CREATE TABLE IF NOT EXISTS `ecobikedatabase`.`bike` (

 `bikeCode` INT(11) NOT NULL,

 `isInUse` INT(10) NOT NULL,

 `value` INT(11) NOT NULL,

 `licensePlate` VARCHAR(45) NOT NULL,

 `dockID` CHAR(10) NOT NULL,

 `type` VARCHAR(45) NOT NULL,

 `remainBattery` float(11) NULL,

 `maxTime` FLOAT(11) NULL,

 `motor` VARCHAR(45) NULL,

 PRIMARY KEY (`bikeCode`),

 CONSTRAINT `dockID`

 FOREIGN KEY (`dockID`)

 REFERENCES `ecobikedatabase`.`dock` (`dockID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

```

-- Table `ecobikedatabase`.`rentbiketransaction`
-----

DROP TABLE IF EXISTS `ecobikedatabase`.`rentbiketransaction` ;

CREATE TABLE IF NOT EXISTS `ecobikedatabase`.`rentbiketransaction` (
  `rentalCode` VARCHAR(45) NOT NULL,
  `bikeCode` INT(11) NOT NULL,
  `type` VARCHAR(45) NOT NULL,
  `rentBikeCost` INT(11) NOT NULL,
  `owner` VARCHAR(45) NOT NULL,
  `rentTime` VARCHAR(45) NOT NULL,
  `returnTime` VARCHAR(45) NULL DEFAULT NULL,
  `deposit` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`rentalCode`),
  CONSTRAINT `bikeCode`
  FOREIGN KEY (`bikeCode`)
  REFERENCES `ecobikedatabase`.`bike` (`bikeCode`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----

-- Table `ecobikedatabase`.`paymenttransaction`
-----

DROP TABLE IF EXISTS `ecobikedatabase`.`paymenttransaction` ;

CREATE TABLE IF NOT EXISTS `ecobikedatabase`.`paymenttransaction` (

```



```

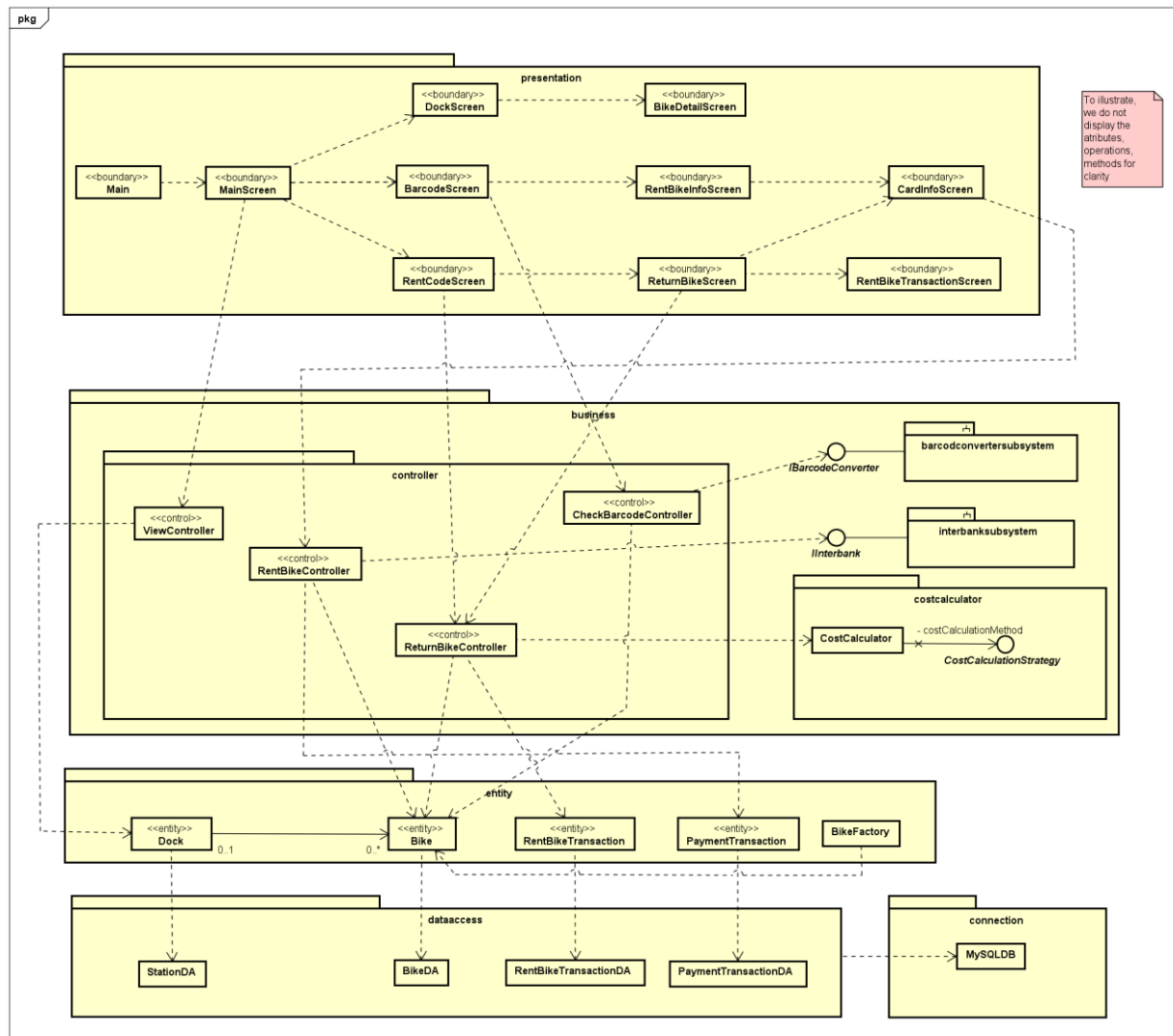
`rentalCode` VARCHAR(45) NOT NULL,
`cardCode` VARCHAR(45) NOT NULL,
`owner` VARCHAR(45) NOT NULL,
`transactionContent` VARCHAR(45) NOT NULL,
`amount` INT(11) NOT NULL,
`time` VARCHAR(45) NOT NULL,
`day` varchar(45) NOT NULL,
PRIMARY KEY (`time`,`day`,`cardCode`),
CONSTRAINT `rentalCode`
    FOREIGN KEY (`rentalCode`)
    REFERENCES `ecobikedbatabase`.`rentbiketransaction` (`rentalCode`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

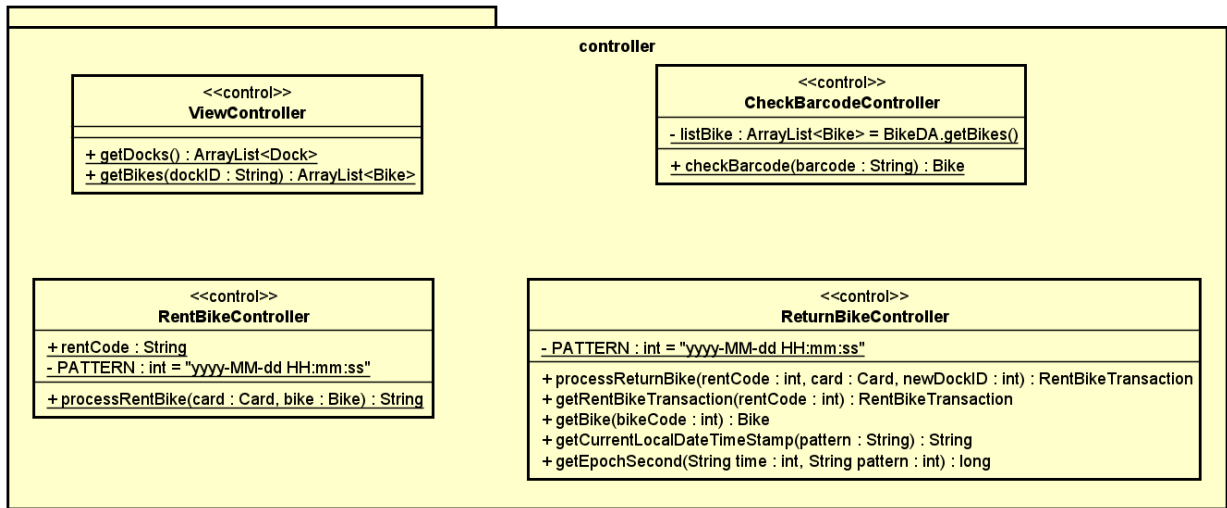
4.3 Thiết kế lớp

4.3.1 Thiết kế lớp tổng quát

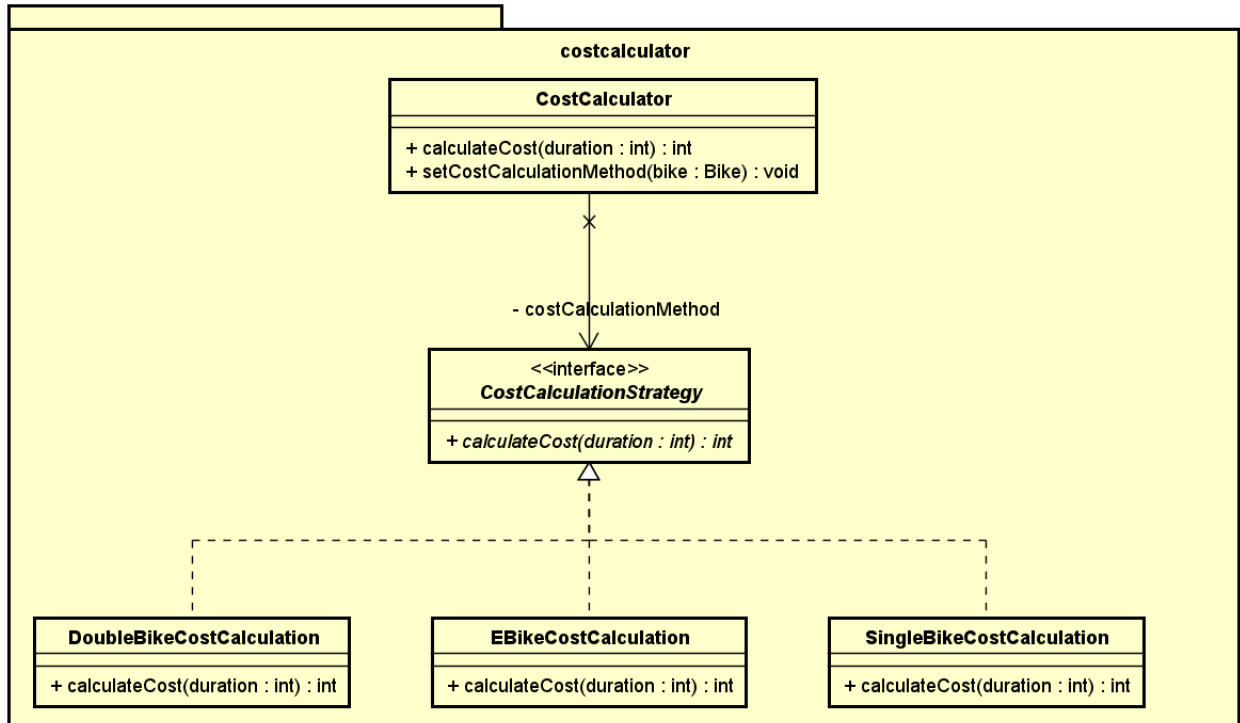


4.3.2 Biểu đồ lớp

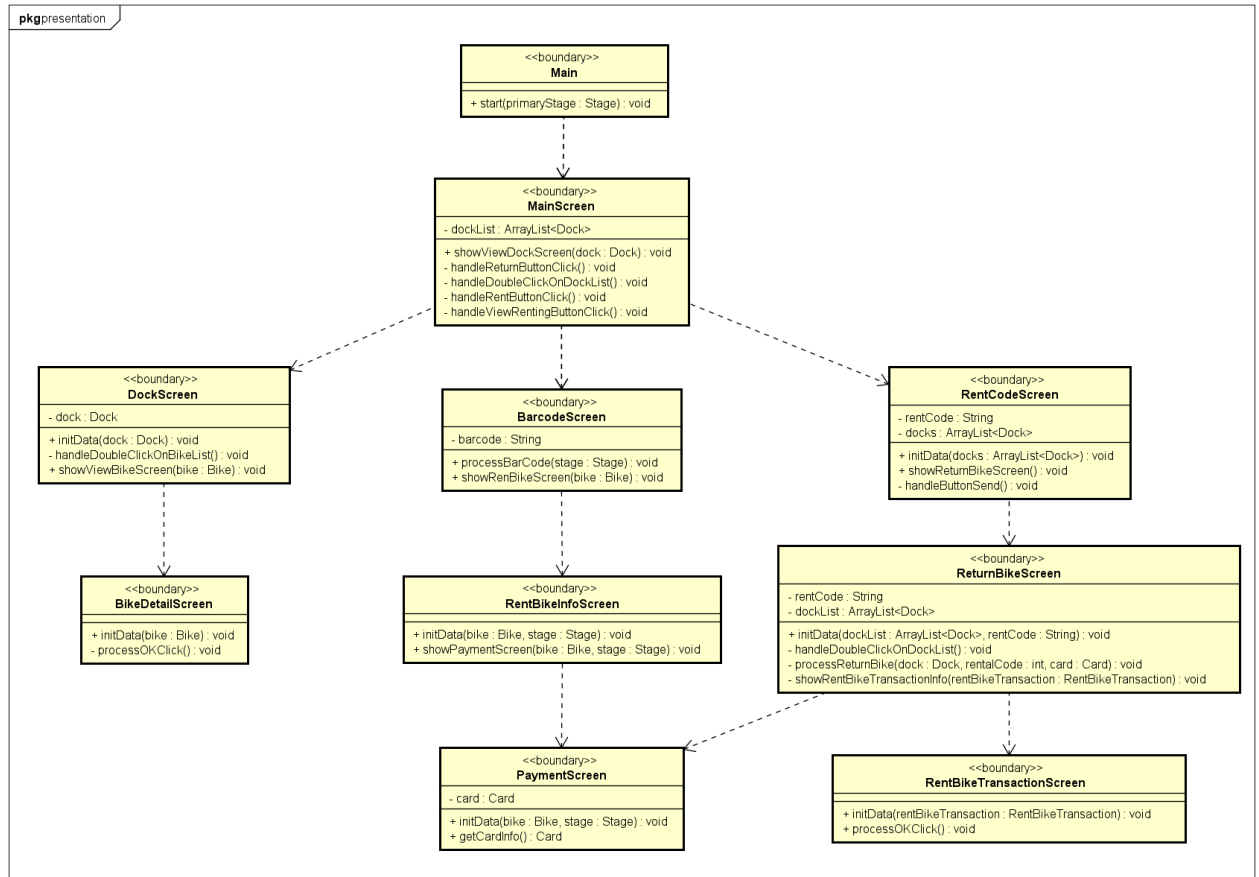
4.3.2.1 Biểu đồ lớp gói controller



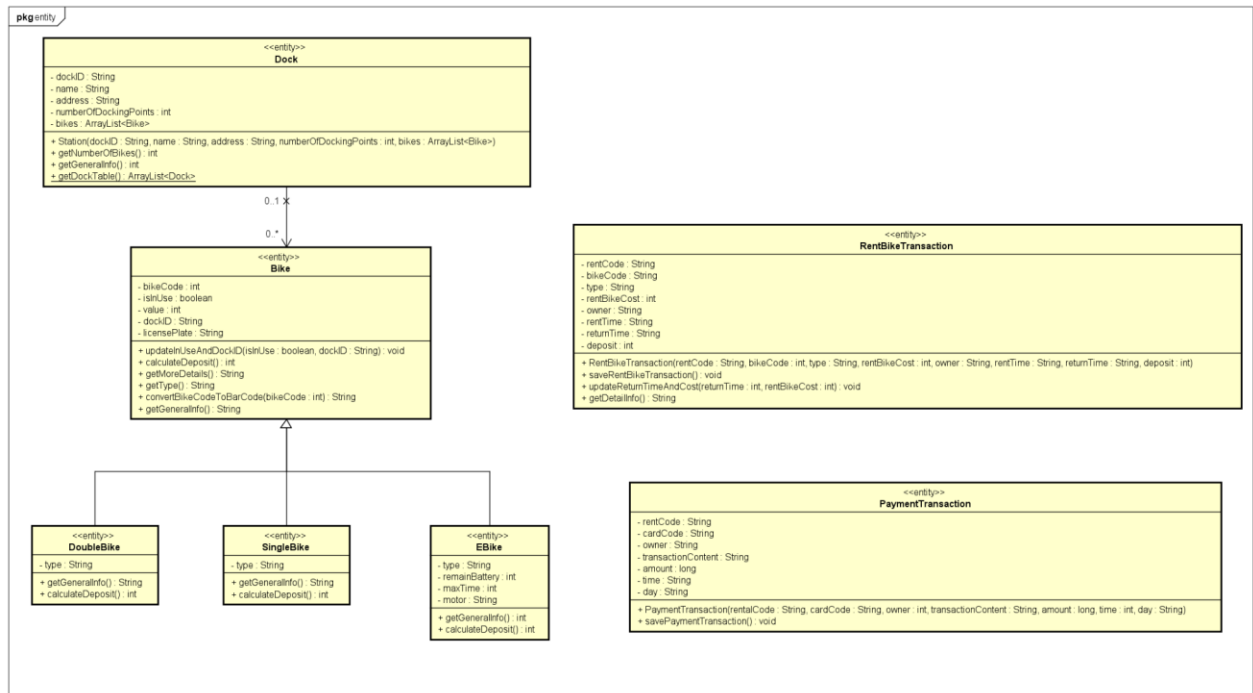
4.3.2.2 Biểu đồ lớp gói costcalculator



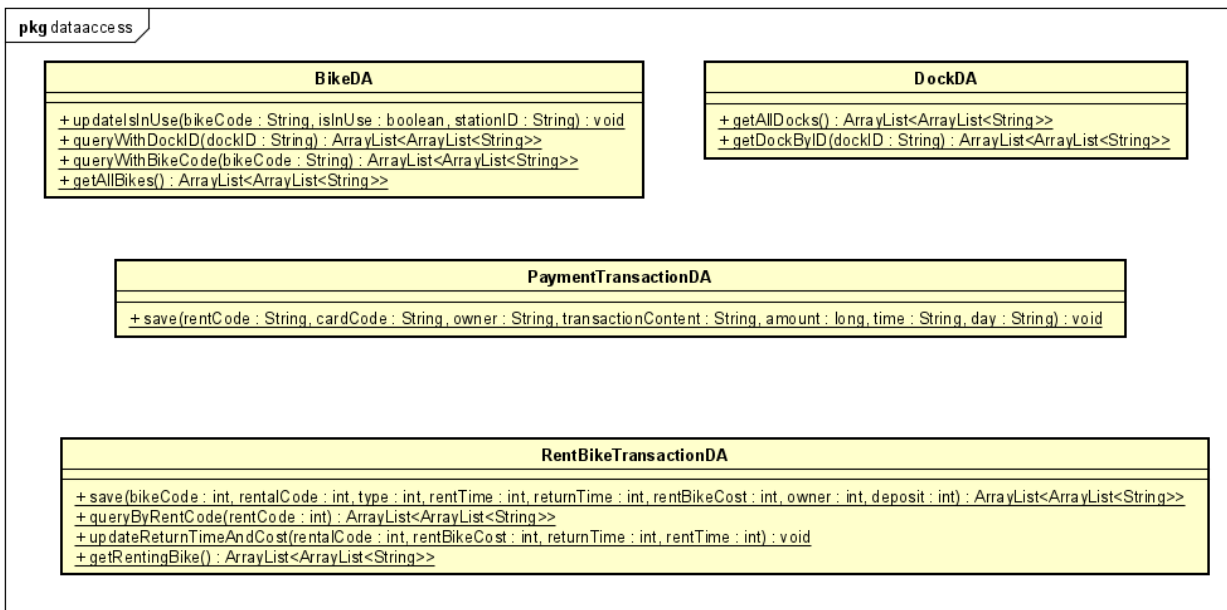
4.3.2.3 Biểu đồ lớp gói presentation



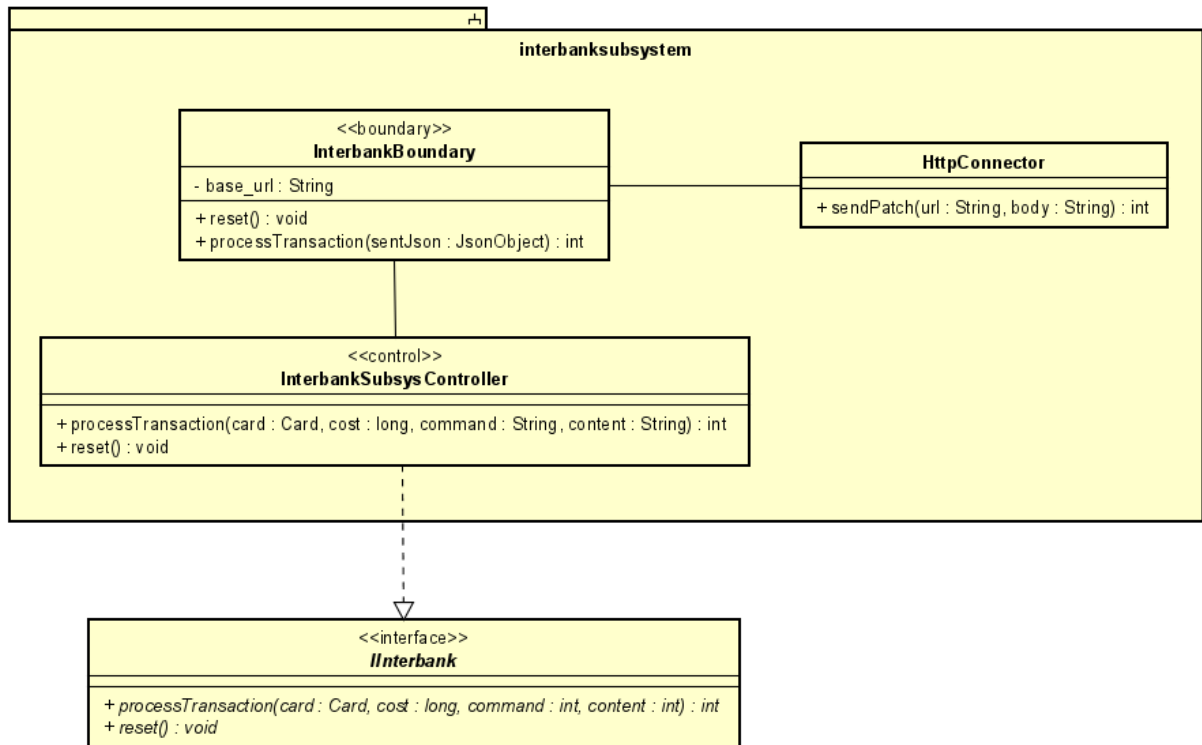
4.3.2.4 Biểu đồ lớp gói entity



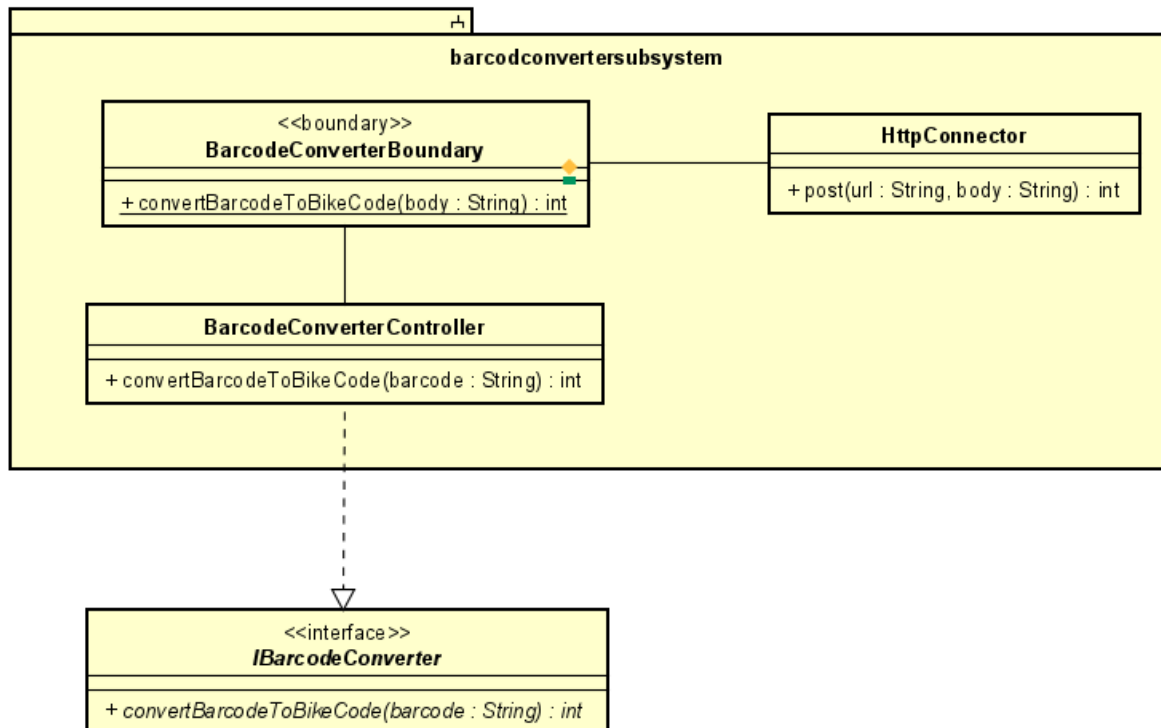
4.3.2.5 Biểu đồ lớp gói dataaccess



4.3.2.6 Biểu đồ lớp subsystem Interbank

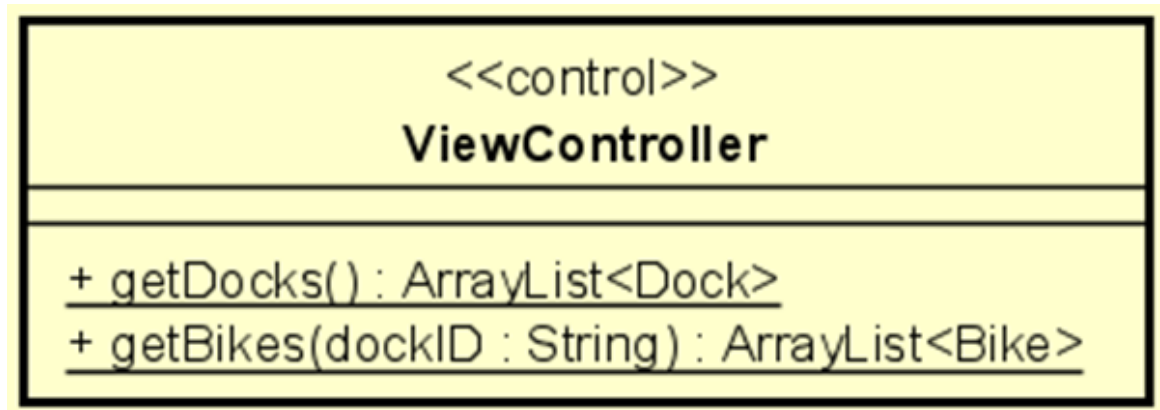


4.3.2.7 Biểu đồ lớp subsystem BarcodeConverter



4.3.3 Thiết kế lớp

4.3.3.1 Class ViewController



Attribute

Không

Operation

#	Name	Return type	Description (purpose)
1	getDocks	ArrayList<Station>	Trả về danh sách các bãi gửi xe
2	getBikes	ArrayList<Bike>	Trả về danh sách xe của một bãi nào đó

Parameter:

- dockID: mã của bãi xe được chọn để hiển thị danh sách xe bên trong

Exception:

Không

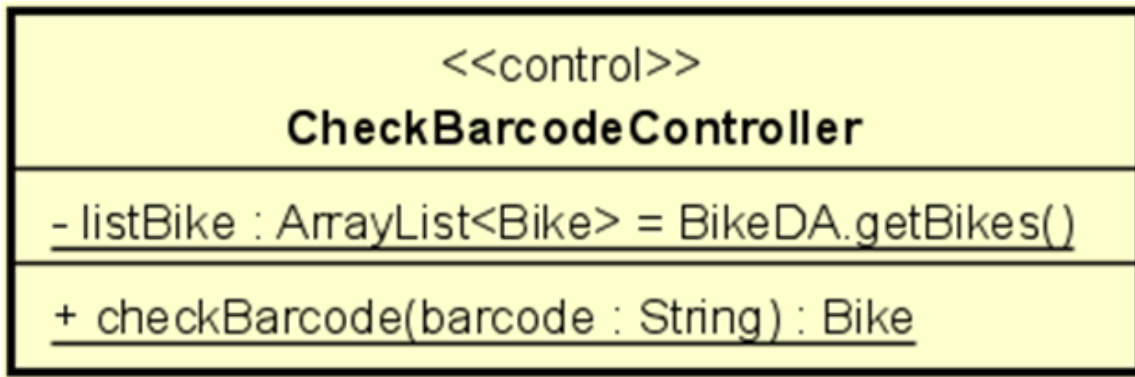
Method

Không

State

Không

4.3.3.2 Class BarCodeController



Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	listBike	ArrayList<Bike>	BikeDA.getBikes()	Danh sách tất cả các xe

Operation

No.	Tên	Kiểu dữ liệu	Mô tả
1	checkBarcode	Bike	Trả xe được truy vấn từ barcode

Parameter:

- barCode: mã barcode được truyền vào

Exception:

- InvalidBarCodeException

Method

Không

State

Không

4.3.3.3 Class RentBikeController



Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	rentCode	String	NULL	Mã của lịch sử thuê xe RentBikeTransaction
2	PATTERN	String	"yyyy-MM-dd HH:mm:ss"	Định dạng thời gian được dùng trong hệ thống

Operation

No.	Tên	Kiểu dữ liệu	Mô tả
1	processRentBike	String	Thực hiện việc thanh toán cọc để thuê xe và trả về mã giao dịch

Parameter:

- card: thẻ dùng để thanh toán
- bike: xe được yêu cầu thuê

Exception: Không

Method: Không

State: Không

4.3.3.4 Class ReturnBikeController

<<control>> ReturnBikeController	
- PATTERN : int = "yyyy-MM-dd HH:mm:ss"	
+ processReturnBike(rentCode : int, card : Card, newDockID : int) : RentBikeTransaction + getRentBikeTransaction(rentCode : int) : RentBikeTransaction + getBike(bikeCode : int) : Bike + getCurrentLocalDateTimeStamp(pattern : String) : String + getEpochSecond(String time : int, String pattern : int) : long	

Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	PATTERN	String	"yyyy-MM-dd HH:mm:ss"	Định dạng thời gian được dùng trong hệ thống

No.	Tên	Kiểu dữ liệu	Mô tả
1	processReturnBike	RentBikeTransaction	Xử lý yêu cầu trả xe và trả về thông tin lịch sử thuê xe (cập nhật thêm thời gian trả xe và phí)
2	getRentBikeTransaction	RentBikeTransaction	Thực hiện giao dịch trả tiền cọc
3	getBike	Bike	Lấy thông tin xe được trả
4	getCurrentLocalDateTimeStamp	String	Thông tin ngày tháng theo Pattern
5	getEpochSecond	long	Trả về thời gian tính bằng giây của String ngày giờ thuê xe và trả xe

Parameter:

- rentCode: mã thuê xe

- card: thẻ dùng để nhận tiền hoàn
- newDockId: bãi đỗ nơi trả xe
- bikeCode: mã xe được trả
- pattern: định dạng ngày giờ
- time: chuỗi ngày giờ (trả xe và thuê xe)

Exception: Không

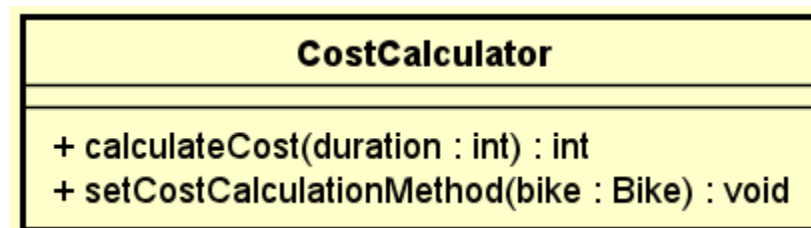
Method

Không

State

Không

4.3.3.5 Class CostCalculator



Attribute

Không

Operation

#	Name	Return type	Description (purpose)
1	calculateCost	int	Tính chi phí thuê dựa vào thời gian thuê
2	setCostCalculationMethod	void	Thiết lập cách tính tiền với loại xe nhất định

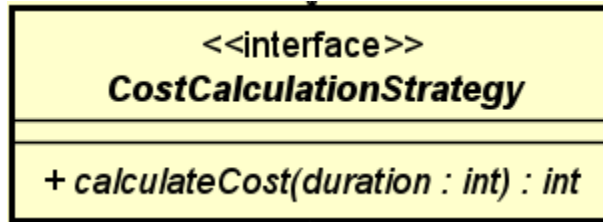
Parameter:

- duration: thời gian dùng xe tính bằng phút
- bike: xe cần tính chi phí

Exception: Không

Method: Không

4.3.3.6 Interface CostCalculationStrategy



Attribute

Không

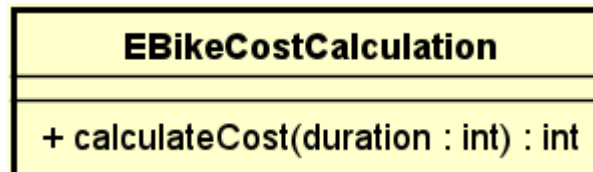
Operation

#	Name	Return type	Description (purpose)
1	calculateCost	int	Tính chi phí thuê dựa vào thời gian thuê

Parameter:

- duration: thời gian dùng xe tính bằng phút

4.3.3.7 Class EBikeCostCalculation



Attribute

Không

Operation

#	Name	Return type	Description (purpose)
1	calculateCost	int	Tính chi phí thuê dựa vào thời gian thuê

Parameter:

- duration: thời gian dùng xe tính bằng phút

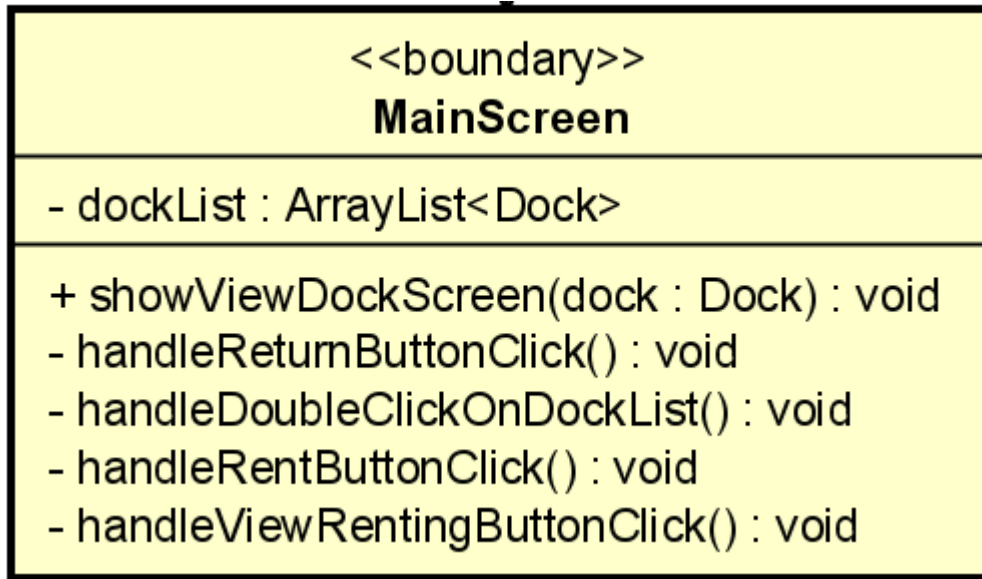
Exception: Không

Method: Không

State: Không

Các class **DoubleCostCalculation** và **SingleCostCalculation** cũng tương tự, đều implement Interface **CostCalculationStrategy**

4.3.3.8 Class mainScreen



Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	dockList	ArrayList<Dock>	NULL	Danh sách các bãi xe có trong hệ thống

Operation

#	Name	Return type	Description (purpose)
1	showViewDockScreen	void	Hiển thị màn hình DockScreen với bãi xe được chọn
2	handleReturnButtonClick	void	Xử lý yêu cầu trả xe
3	handleDoubleClickOnDockList	void	Xử lý yêu cầu chọn bãi xe để hiện thị
4	handleRentButtonClick	void	Xử lý yêu cầu thuê xe
5	handleViewRentingButtonClick	void	Xử lý yêu cầu xem xe đang thuê

Parameter:

- dock: bãi xe được chọn để hiển thị

Exception:

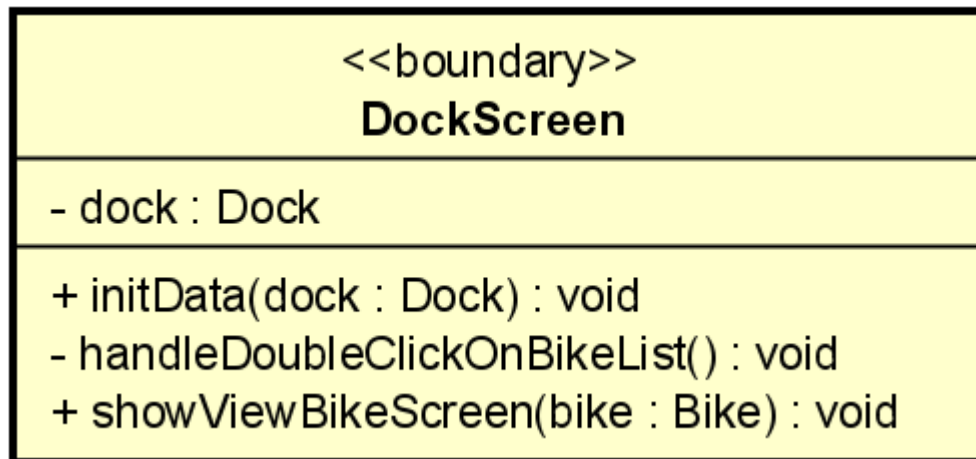
Không

Method

Không

State

Không

4.3.3.9 Class DockScreen**Attribute**

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	dock	Dock	NULL	Bãi xe được chọn để hiển thị

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Truyền giá trị vào cho class
2	handleDoubleClickOnBikeList	void	Xử lý yêu cầu chọn xe trong bãi
3	showViewBikeScreen	void	Hiển thị xe được chọn

Parameter:

- dock: bãi xe được chọn để hiển thị
- bike: xe được chọn để hiển thị

Exception:

Không

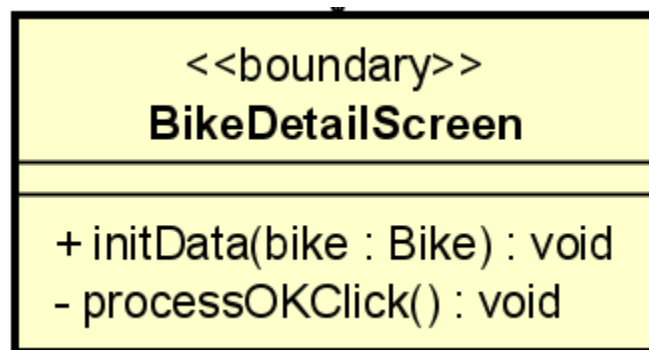
Method

Không

State

Không

4.3.3.10 Class BikeDetailScreen



Attribute

Không

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Hiển thị thông tin của xe được chọn
2	processOKClick	void	Người dùng xác nhận xem xong thông tin

Parameter:

- bike: xe được chọn

Exception:

Không

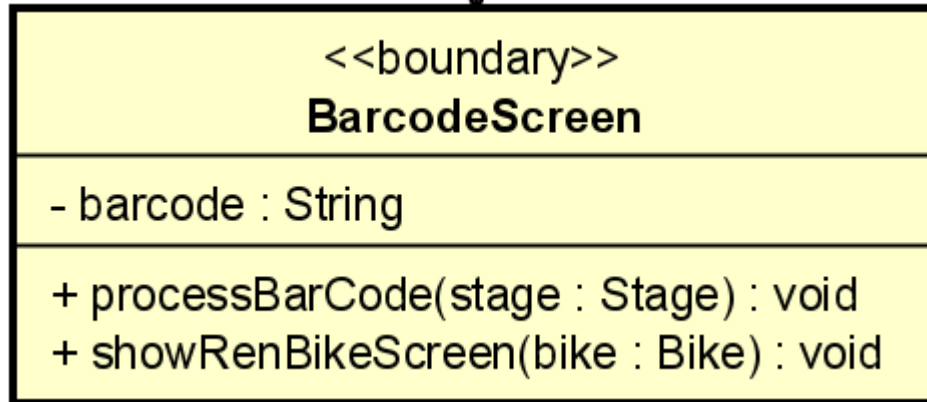
Method

Không

State

Không

4.3.3.11 Class BarcodeScreen



Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	barcode	Dock	NULL	Mã barcode người dùng nhập vào

Operation

#	Name	Return type	Description (purpose)
1	procesBarcode	void	Truyền giá trị vào cho class
2	processOKClick	void	Người dùng xác nhận đã xem xong thông tin

Parameter:

- bike: xe được chọn để hiển thị

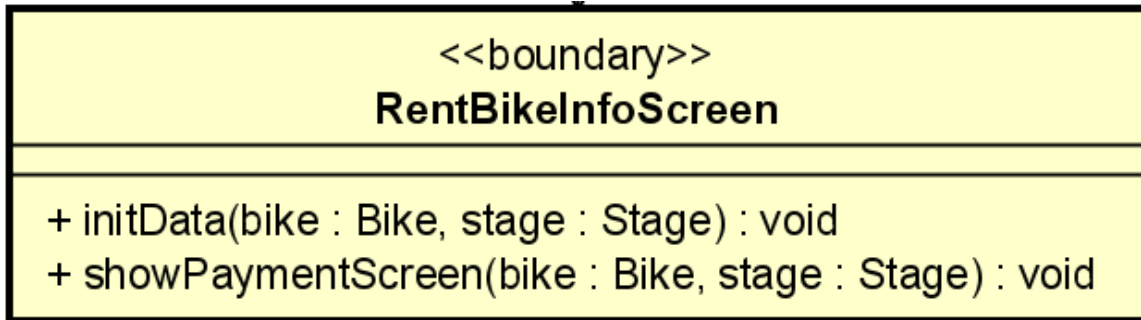
Exception:

Không

Method: Không

State: Không

4.3.3.12 Class RentBikeInfoScreen



Attribute

Không

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Truyền giá trị vào cho class
2	showPaymentScreen	void	Hiển thị màn hình thanh toán

Parameter:

- bike: xe được chọn để thuê (xác định từ barcode)

Exception:

Không

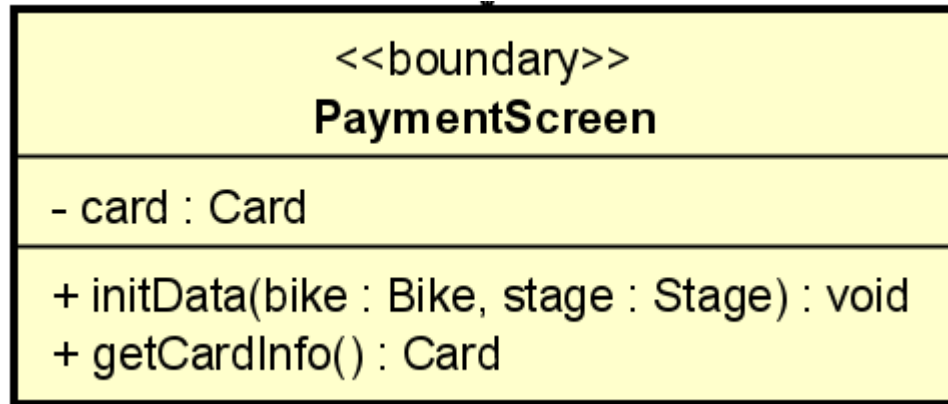
Method

Không

State

Không

4.3.3.13 Class PaymentScreen



Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	card	Card	NULL	Thẻ được dùng để thanh toán

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Truyền giá trị vào cho class
2	getCardInfo	Card	Trả về thông tin thẻ

Parameter:

- bike: xe cần đặt cọc thanh toán

Exception:

Không

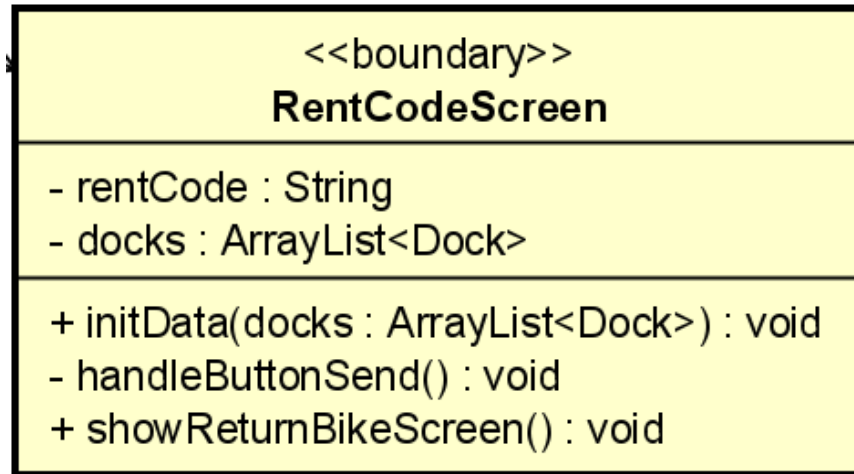
Method

Không

State

Không

4.3.3.14 Class RentCodeScreen



Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	rentCode	String	NULL	Mã thuê xe cần xử lý
2	docks	ArrayList<Dock>	NULL	Danh sách các bãi xe

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Truyền giá trị vào cho class
2	handleButtonSend	void	xử lý mã thuê xe người dùng nhập vào
3	showReturnBikeScreen	void	Hiển thị màn hình trả xe

Parameter:

- docks : danh sách các bãi xe

Exception:

Không

Method

Không

State: Không

4.3.3.15 Class RentCodeScreen

<<boundary>> ReturnBikeScreen	
- rentCode : String - dockList : ArrayList<Dock>	
+ initData(dockList : ArrayList<Dock>, rentCode : String) : void - handleDoubleClickOnDockList() : void - processReturnBike(dock : Dock, rentalCode : int, card : Card) : void - showRentBikeTransactionInfo(rentBikeTransaction : RentBikeTransaction) : void	

Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	rentCode	String	NULL	Mã thuê xe cần xử lý
2	docks	ArrayList<Dock>	NULL	Danh sách các bãi xe

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Truyền giá trị vào cho class
2	handleDoubleClickOnDockList	void	xử lý thao tác chọn bãi xe để trả
3	processReturnBike	void	Xử lý việc trả xe
4	showRentBikeTransactionInfo	void	Hiển thị thông tin chi tiết của việc thuê xe

Parameter:

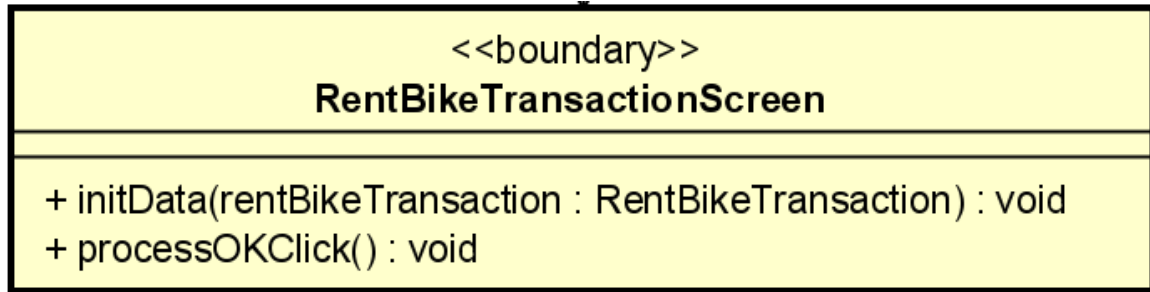
- docks : danh sách các bãi xe
- rentCode: mã thuê xe RentBikeTransaction
- dock: bãi xe được chọn để trả
- card: thẻ được dùng để nhận cọc
- rentBikeTransaction: thông tin thuê xe

Exception: Không

Method: Không

State: Không

4.3.3.16 Class RentBikeTransactionScreen



Attribute

Không

Operation

#	Name	Return type	Description (purpose)
1	initData	void	Truyền giá trị vào cho class
2	processOKClick	void	Người dùng xác nhận đọc xong thông tin

Parameter:

- rentBikeTransaction: thông tin thuê xe

Exception:

Không

Method:

Không

State:

Không

4.3.3.17 Class Dock

<<entity>> Dock	
- dockID : String - name : String - address : String - numberOfDockingPoints : int - bikes : ArrayList<Bike>	
+ Station(dockID : String, name : String, address : String, numberOfDockingPoints : int, bikes : ArrayList<Bike>) + getNumberOfBikes() : int + getGeneralInfo() : int <u>+ getDockTable() : ArrayList<Dock></u>	

Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	dockID	String	NULL	Mã bãi xe
2	name	String	NULL	Tên bãi xe
3	address	String	NULL	Địa chỉ
4	number Point	int	NULL	Số chỗ trống
5	bikes	ArrayList<Bike>	NULL	Danh sách xe trong bãi

Operation

#	Name	Return type	Description (purpose)
1	getNumberOfBikes	int	số xe trong bãi
2	getGeneralInfo	String	Thông tin bãi xe
3	getDockTable	ArrayList<Dock >	danh sách các bãi xe

4.3.3.18 Class Bike

<<entity>> Bike	
- bikeCode : int - isInUse : boolean - value : int - dockID : String - licensePlate : String	
+ updateInUseAndDockID(isInUse : boolean, dockID : String) : void + calculateDeposit() : int + getMoreDetails() : String + getType() : String + convertBikeCodeToBarCode(bikeCode : int) : String + getGeneralInfo() : String	

Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	bikeCode	int	NULL	Mã xe
2	isInUse	boolean	NULL	Trạng thái sử dụng
3	value	int	NULL	Giá trị xe
4	dockID	String	NULL	Mã bãi xe
5	license	String	NULL	Biển số xe

Operation

#	Name	Return type	Description (purpose)
1	updateInUseAndDockID	void	thiết lập trạng thái xe và bãi
2	calculateDeposit	int	Tính phí đặt cọc
3	getMoreDetail	String	Thông tin thêm
4	getType	String	loại xe
5	convertBikeCodeToBarCode	String	chuyển mã xe thành barcode
6	getGeneralInfo	String	thông tin chung của xe

4.3.3.19 Class RentBikeTransaction

<<entity>> RentBikeTransaction	
- rentCode : String - bikeCode : String - type : String - rentBikeCost : int - owner : String - rentTime : String - returnTime : String - deposit : int	
+ RentBikeTransaction(rentCode : String, bikeCode : int, type : String, rentBikeCost : int, owner : String, rentTime : String, returnTime : String, deposit : int) + saveRentBikeTransaction() : void + updateReturnTimeAndCost(returnTime : int, rentBikeCost : int) : void + getDetailInfo() : String	

Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	rentCode	String	NULL	Mã thuê xe
2	bikeCode	int	NULL	Xe thuê
3	type	String	NULL	Loại xe
4	rentBikeCost	int	NULL	Chi phí thuê xe
5	owner	String	NULL	Người thuê
6	rentTime	String	NULL	Thời điểm thuê
7	returnTime	String	NULL	Thời điểm trả
8	deposit	int	NULL	Tiền cọc

Operation

#	Name	Return type	Description (purpose)
1	saveRentBikeTransaction	void	lưu lại lịch sử thuê xe
2	updateReturnTimeAndCost	void	Cập nhật thời điểm trả xe và tiền thuê (thực hiện khi trả)
3	getGeneralInfo	String	Thông tin thuê xe

4.3.3.20 Class PaymentTransaction

<<entity>> PaymentTransaction	
- rentCode : String - cardCode : String - owner : String - transactionContent : String - amount : long - time : String - day : String	
+ PaymentTransaction(rentalCode : String, cardCode : String, owner : int, transactionContent : String, amount : long, time : int, day : String) + savePaymentTransaction() : void	

Attribute

No.	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	rentCode	String	NULL	Mã thuê xe
2	cardCode	String	NULL	Số thẻ
3	owner	String	NULL	Người thuê
4	transaction Content	String	NULL	Nội dung giao dịch
5	amount	int	NULL	Số tiền giao dịch
6	time	String	NULL	Giờ giao dịch
7	day	String	NULL	Ngày giao dịch

Operation

#	Name	Return type	Description (purpose)
1	savePaymentTransaction	void	lưu lại lịch sử giao dịch

5 Đánh giá thiết kế

5.1 Mục tiêu và định hướng

Mục tiêu: - Mang đến trải nghiệm tốt với người dùng

Định hướng:

- Thiết kế theo nguyên lý hướng đối tượng
- Tránh lặp code
- Giải thích code đầy đủ và chi tiết, viết java doc cho các hàm và các lớp phục vụ cho việc bảo trì, mở rộng sau này

5.2 Chiến lược kiến trúc

Hệ thống được thiết kế với ba mục tiêu chính: dễ mở rộng, dễ bảo trì và tái sử dụng mã nguồn.

Các công nghệ được sử dụng gồm có:

- Ngôn ngữ lập trình: Java- version 11
- Hệ quản trị cơ sở dữ liệu: MySQL
- IDE: IntelliJ

Trong khi thiết kế và cài đặt, việc tối ưu hóa bộ nhớ và hiệu năng cũng được xem xét chi tiết và đầy đủ

5.3 Coupling và Cohesion

5.3.1 Coupling

5.3.1.1 Common Coupling

Đây là trường hợp 2 module cùng chia sẻ chung dữ liệu, 2 global structure có thể cùng vào chỉnh sửa, truy cập hoặc có chung những khối mã nguồn thì sẽ vi phạm common coupling.

Tuy nhiên, object oriented programming không có common data. Tất cả data đều thuộc về lớp.

Do đó, project hiện tại không vi phạm common coupling.

5.3.1.2 **Control Coupling**

Một module vi phạm control coupling khi nó truyền tham số điều khiển cho các module khác thông qua việc gọi method. Điều này không tốt vì thành phần được gọi sẽ biết được cấu trúc bên trong thành phần gọi và khi cấu trúc này bị thay đổi thì thành phần được gọi sẽ phải thay đổi theo.

Capstone vi phạm control coupling ở những lớp điều khiển screen.

Cụ thể là có 1 số lớp điều khiển screen truyền tham số điều khiển vào phương thức display của lớp NotificationBox

5.3.1.3 **Stamp coupling**

Hai lớp được coi là vi phạm stamp coupling nếu một lớp gửi một collection hoặc một object dưới dạng tham số và chỉ một vài phần data được sử dụng ở lớp thứ hai.

Capstone có loại coupling này ở lớp RentBikeController khi truyền cả đối tượng Bike nhưng không sử dụng toàn bộ thuộc tính của nó, điều này vẫn chấp nhận được để cho phương thức không bị quá nhiều tham số truyền vào, sẽ gây rối.

5.3.1.4 **Data coupling**

Đây là level mà chúng ta hướng đến.

Ngoài các lớp vi phạm kể trên thì những lớp còn lại ở mức độ này.

5.3.1.5 **Uncoupled**

Capstone không có loại này

5.3.2 **Cohesion**

5.3.2.1 **Coincidental cohesion**

Các sub component đặt trong 1 component vì tính ngẫu nhiên

Capstone không có loại này

5.3.2.2 *Logical cohesion*

Các thành phần trong 1 module có liên quan đến nhau nhưng về mặt logic chứ không phải chức năng.

Lớp ViewController, có 2 phương thức getDocks, getBikes lấy ra danh sách bãi xe và xe để hiển thị ra màn hình, chúng ở chung lớp vì có liên quan đến việc hiển thị ra màn hình nhưng rõ ràng chức năng của chúng là khác nhau.

5.3.2.3 *Temporal cohesion*

Các sub component đặt trong một component vì chúng liên quan đến nhau về mặt thời gian chứ không phải về mặt chức năng.

Lớp Main thực hiện chức năng hiển thị SplashScreen trước rồi mới hiển thị mainScreen.

5.3.2.4 *Procedural cohesion*

Các thành phần đặt trong 1 module vì nó có quan hệ chặt chẽ với nhau theo một thứ tự nào đó chứ không liên hệ với nhau về mặt chức năng.

Lớp ReturnBikeController phải kiểm tra thẻ tín dụng hợp lệ trước rồi mới tính tiền.

5.3.2.5 *Communicational cohesion*

Các thành phần đặt trong cùng một module vì cùng thực hiện trên cùng một dữ liệu.

Lớp ReturnBikeController có các phương thức thực hiện chức năng khác nhau nhưng đều thực hiện trên cùng dữ liệu là rentalCode

5.3.2.6 *Sequential cohesion*

Trong một module, output của thành phần này là input của thành phần kia.

Ví dụ tính phí đặt cọc của xe trong lớp Bike bằng 40% giá trị thực trước, sau đó truyền kết quả này vào hàm tạo hóa đơn đặt cọc thuê xe.

5.3.2.7 *Informational cohesion*

Các operation có tính độc lập (có input và output riêng nhưng chúng có thể thao tác trên một tập dữ liệu chung là attribute của lớp đó)

Loại cohesion này ở trong các lớp entity như là Bike hay Dock.

5.3.2.8 *Functional cohesion*

Mỗi một subcomponent thực hiện một công việc nào đó và hướng đến mục đích chung của component đó.

Những lớp tính tiền được tách riêng ra cho từng loại xe thỏa mãn mức độ cohesion này.

5.4 Nguyên lý thiết kế

5.4.1 Single Responsibility

Một lớp chỉ nên chịu trách nhiệm về một nhiệm vụ cụ thể nào đó mà thôi. Một lớp quá quá nhiều chức năng sẽ trở nên cồng kềnh, khó đọc, khó bảo trì, nên chỉ có duy nhất một lý do để thay đổi một lớp, không nên có 2 lý do trở lên.

Vi phạm nguyên tắc này ở ReturnBikeController vì thực hiện nhiều chức năng khác nhau phục vụ cho việc trả xe.

5.4.2 Open/Closed

Theo nguyên lý này, mỗi khi chúng ta thêm mới chức năng chúng ta nên viết class mới extend từ class đã có chứ không nên chỉnh sửa trực tiếp nội dung trên class đã viết.

Nguyên tắc này thỏa mãn ở modul tính tiền vì nếu cần đổi cách tính tiền thì ta viết 1 lớp khác rồi cho implements interface CostCalculation là được.

5.4.3 Liskov Substitution

Nguyên tắc này nói rằng, các đối tượng của class con có thể thay thế cho lớp cha ở mọi tính huống mà không gây ra lỗi, hoặc nếu không, chúng ta có sự trừu tượng sai. Hệ thống phân cấp kế thừa ở class Bike đã tuân theo nguyên lý này vì các lớp con của Bike đều thực hiện được các chức năng của lớp Bike

5.4.4 Interface Segregation

Nguyên lý này nói rằng, thay vì một sử dụng một interface quá lớn, quá nhiều phương thức thì chúng ta sẽ tách nhỏ ra thành các interface con với mục đích cụ thể. Vì khi để một interface quá to, các lớp implement sẽ phải implement các phương thức mà bản thân nó không cần dùng đến.

Capstone không vi phạm nguyên tắc này vì không có interface nào quá nhiều phương thức.

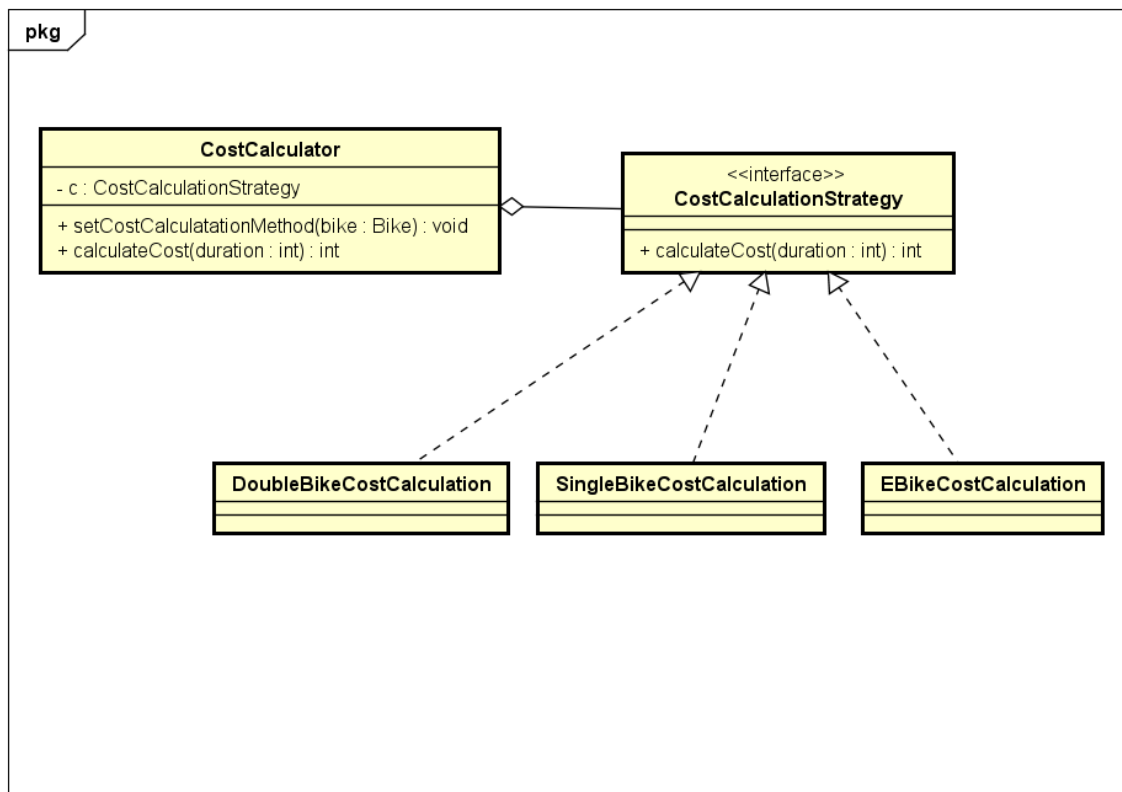
5.4.5 Dependency Inversion

Có thể hiểu nguyên lý này như sau: những thành phần trong một chương trình chỉ nên phụ thuộc vào những cái trừu tượng. Những thành phần trừu tượng không nên phụ thuộc vào một thành phần mang tính cụ thể mà nên ngược lại.

Hiện tại, các lớp hiển thị màn hình như mainScreen hay ReturnBikeScreen ... phụ thuộc khá chặt vào lớp Dock -> vi phạm nguyên tắc này

5.5 Mẫu thiết kế Design pattern

5.5.1 Strategy cho chức năng tính phí trả xe của các loại xe



Ý nghĩa : tách rời phần xử lý một chức năng cụ thể ra khỏi đối tượng

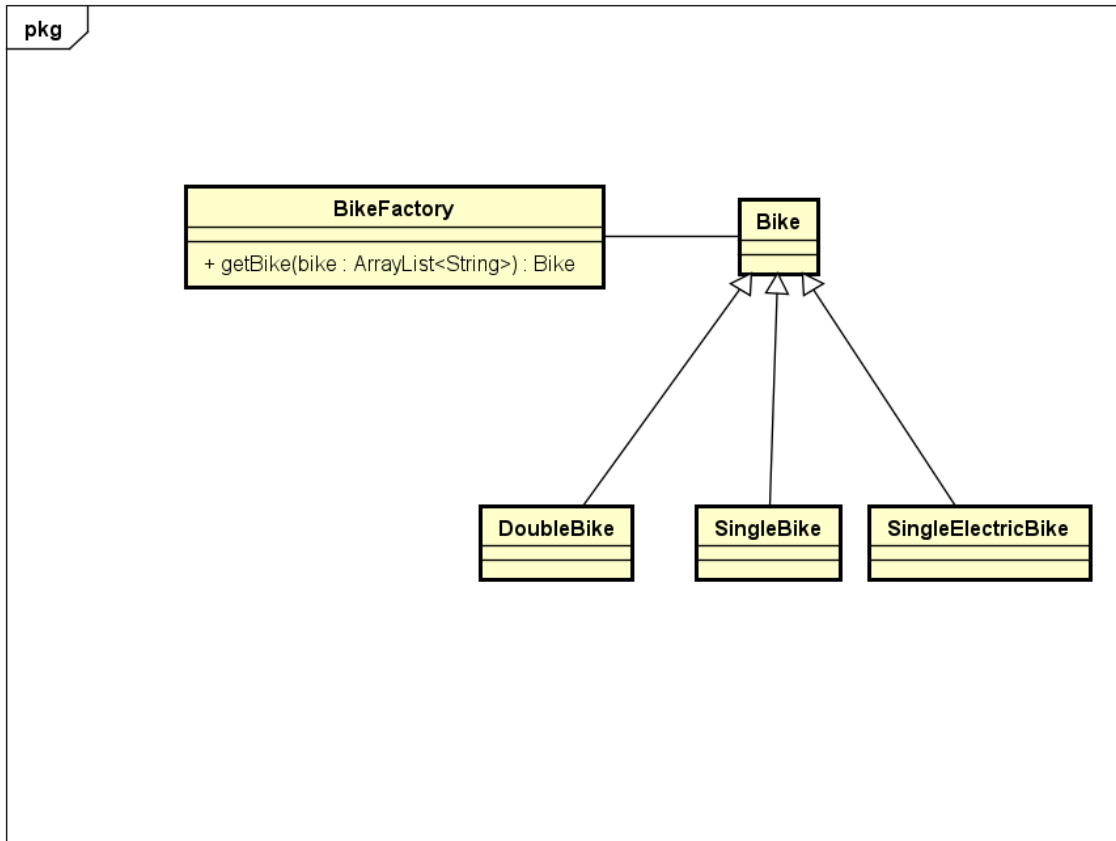
Sử dụng khi :

- Khi muốn có thể thay đổi các thuật toán được sử dụng bên trong một đối tượng tại thời điểm run-time.
- Khi có một đoạn mã thường thay đổi, và muốn tách chúng ra khỏi chương trình chính để dễ dàng bảo trì.

Lợi ích : Đảm bảo Single responsibility principle

Đảm bảo Open/Closed Principle

5.5.2 Factory



Sử dụng khi:

Có một lớp cha (Bike) với nhiều lớp con và dựa trên đầu vào, chúng ta cần trả về một class con. Mô hình này giúp chúng ta đưa trách nhiệm của việc khởi tạo một lớp từ phía người dùng (client) sang lớp Factory.

Lợi ích

Giảm sự phụ thuộc giữa các module

Dễ bổ sung thêm lớp con trong tương lai

Khởi tạo các Objects mà che giấu đi xử lý logic của việc khởi tạo đấy. Người dùng không biết logic thực sự được khởi tạo bên dưới phương thức factory.

