RENT Class Design

1. Create Initial Design Classes

```
RENT
├── common
│   ├── exception
│   └── type
├── controller
│   ├── bike
│   ├── rent
│   │   ├── ○ PaymentConfirmationController
│   │   └── ○ PaymentScreenController
│   ├── return
│   ├── ○ BarcodeController
│   ├── ○ HomeScreenController
│   └── ○ SplashScreenController
├── interbank
├── model
│   ├── bike
│   ├── dock
│   ├── invoice
│   │   ├── ○ Invoice
│   │   └── ○ InvoiceHistory
│   ├── payment
│   │   ├── ○ CreditCard
│   │   └── ○ PaymentTransaction
│   └── session
│       ├── ○ Session
│       └── ○ SessionManager
└── view
    ├── bike
    ├── rent
    │   ├── ⊢○ PaymentConfirmationHandler
    │   └── ⊢○ PaymentScreenHandler
    ├── return
    ├── ⊢○ HomeScreenHandler
    └── ⊢○ SplashScreenHandler
```
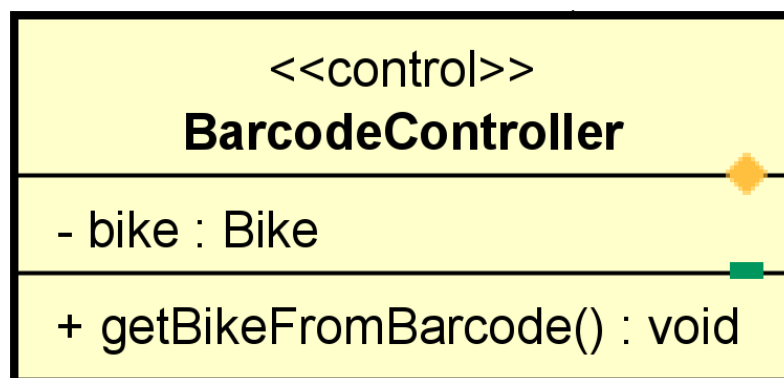
2. Define Relationships Between Classes
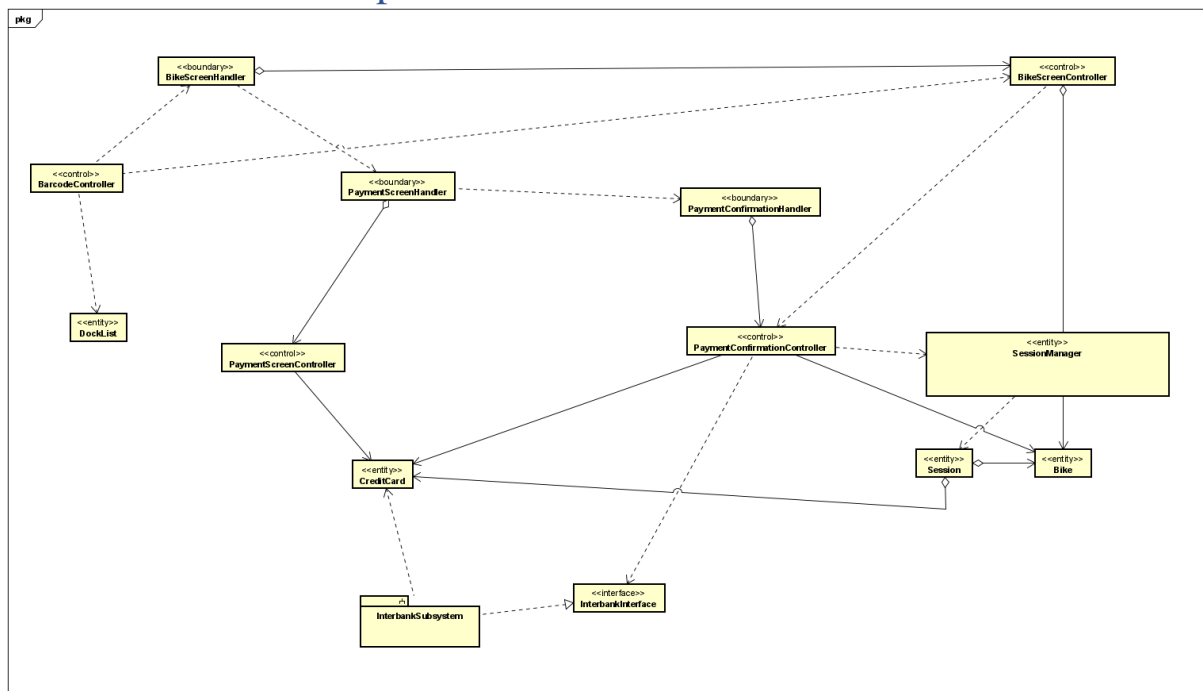
3. Class Design
3.1. Class "BarcodeController"

Attribute

#	Name	Data type	Default value	Description
1	bike	Bike	NULL	The result of barcode searching

Operation

#	Name	Return type	Desciption
1	getBikeFromBarcode	void	Get bike using barcode from DockList

Parameter:

    None

Exception:

    None

Method
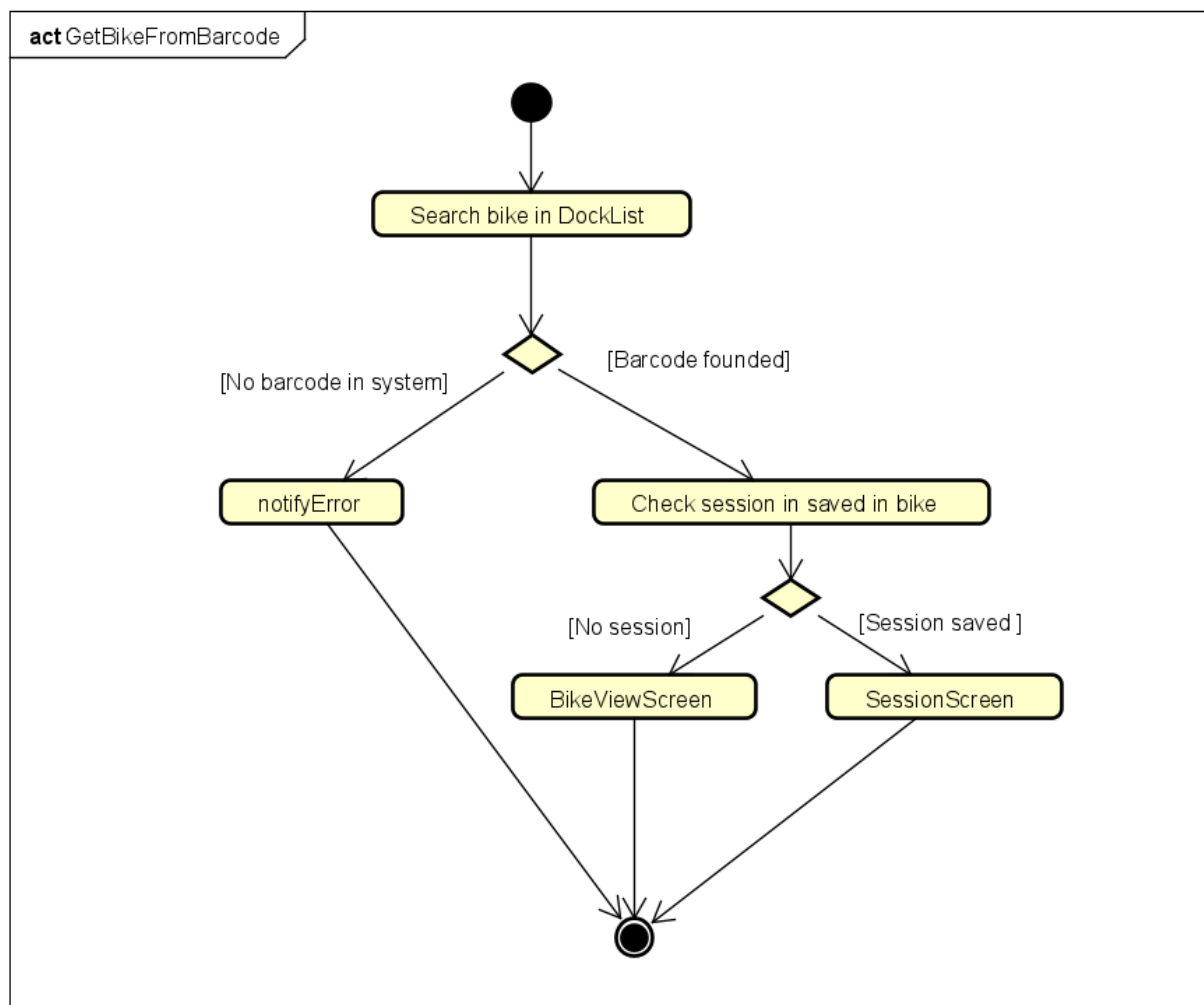
State

    None

3.2. Class "PaymentScreenHandler"
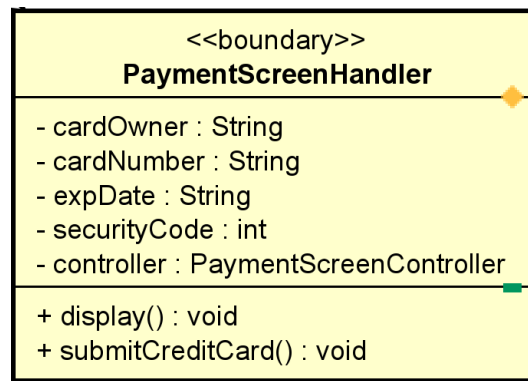
```
            <<boundary>>
        PaymentScreenHandler

- cardOwner : String
- cardNumber : String
- expDate : String
- securityCode : int
- controller : PaymentScreenController

+ display() : void
+ submitCreditCard() : void
```

Attribute

#	Name	Data type	Default value	Description
1	controller	PaymentScreenController	NULL	Controller for this screen
2	cardOwner	String	Empty	Card owner
3	cardNumber	String	Empty	Card number
4	expDate	String	Empty	Expired Date
5	securityCode	Int	Empty	Security Date

Operation

#	Name	Return type	Desciption
3	display	void	Display screen
4	submitCreditCard	void	Send card to next handler

Parameter:

    None

Exception:

    None

Method

    None

State

    None

3.3. Class "PaymentScreenController"

```
┌─────────────────────────────────────────────────────────────────────────┐
│                          <<control>>                                  ◆  │
│                      PaymentScreenController                             │
├─────────────────────────────────────────────────────────────────────────┤
│ - card : CreditCard                                                   ▬  │
├─────────────────────────────────────────────────────────────────────────┤
│ + createCard(cardOwner : String, cardNumber : String, expDate : String, securityCode : int) : void │
└─────────────────────────────────────────────────────────────────────────┘
```

Attribute

#	Name	Data type	Default value	Description
1	card	CreditCard	NULL	Card for payment

Operation

#	Name	Return type	Desciption
1	createCard	void	Create an instance of CreditCard

Parameter:

cardOwner – card owner

cardNumber – card number

expDate – expired date

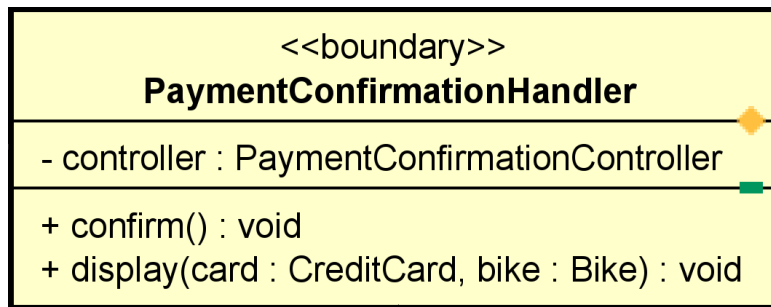securityCode – security code

Exception:

None

Method

None

State

None

3.4. Class "PaymentConfirmationHandler"

```
               ┌─────────────────────────────────────────────┐
               │          <<boundary>>                    ◆    │
               │    PaymentConfirmationHandler                 │
               ├─────────────────────────────────────────────┤
               │ - controller : PaymentConfirmationController  │
               │                                          ▬    │
               ├─────────────────────────────────────────────┤
               │ + confirm() : void                            │
               │ + display(card : CreditCard, bike : Bike) : void │
               └─────────────────────────────────────────────┘
```

Attribute

#	Name	Data type	Default value	Description
1	controller	PaymentConfirmationController	NULL	Controller for this screen

Operation

#	Name	Return type	Desciption
1	display	void	Display screen
2	confirm	void	Move to next screen

Parameter:

      None

Exception:

      None

Method

      None

State

      None

3.5. Class "PaymentConfirmationController"
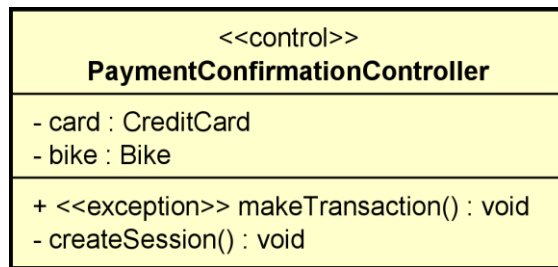
```
                <<control>>
        PaymentConfirmationController

    - card : CreditCard
    - bike : Bike

    + <<exception>> makeTransaction() : void
    - createSession() : void
```

Attribute

#	Name	Data type	Default value	Description
1	card	CreditCard	NULL	Card for payment
2	bike	Bike	NULL	Bike for renting

Operation

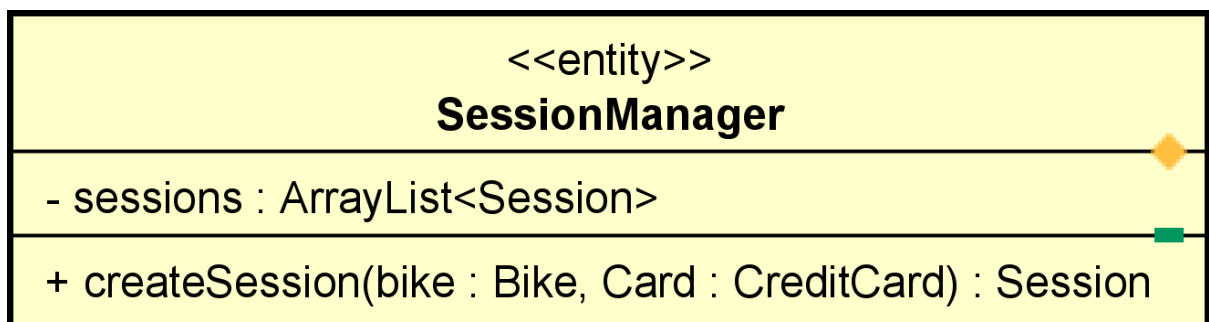| #	Name	Return type	Desciption
1	makeTransaction	void	Make transaction
2	createSession	void	Create new Session

Parameter:

    None

Exception:

    InvalidCardException

    NotEnoughBalanceException

    InternalServerErrorException

Method

    None

3.6. Class "SessionManager"


```
                <<entity>>
              SessionManager

    - sessions : ArrayList<Session>

    + createSession(bike : Bike, Card : CreditCard) : Session
```

Attribute

#	Name	Data type	Default value	Description
1	sessions	ArrayList<Session>	Empty ArrayList	List of sessions

Operation

#	Name	Return type	Desciption
1	createSession	Session	Create new session and add to list

Parameter:

      bike – the rented bike

      card – the used card

Exception:

      None

Method

      None

4. Class Diagram