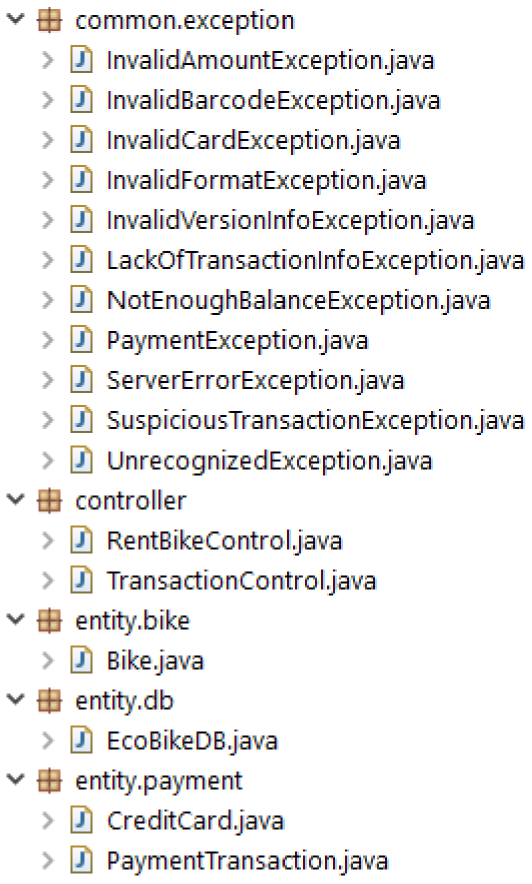
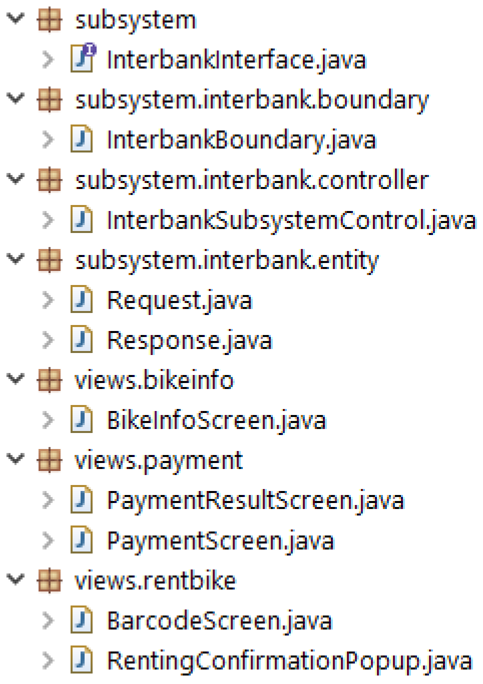
1. Package Group

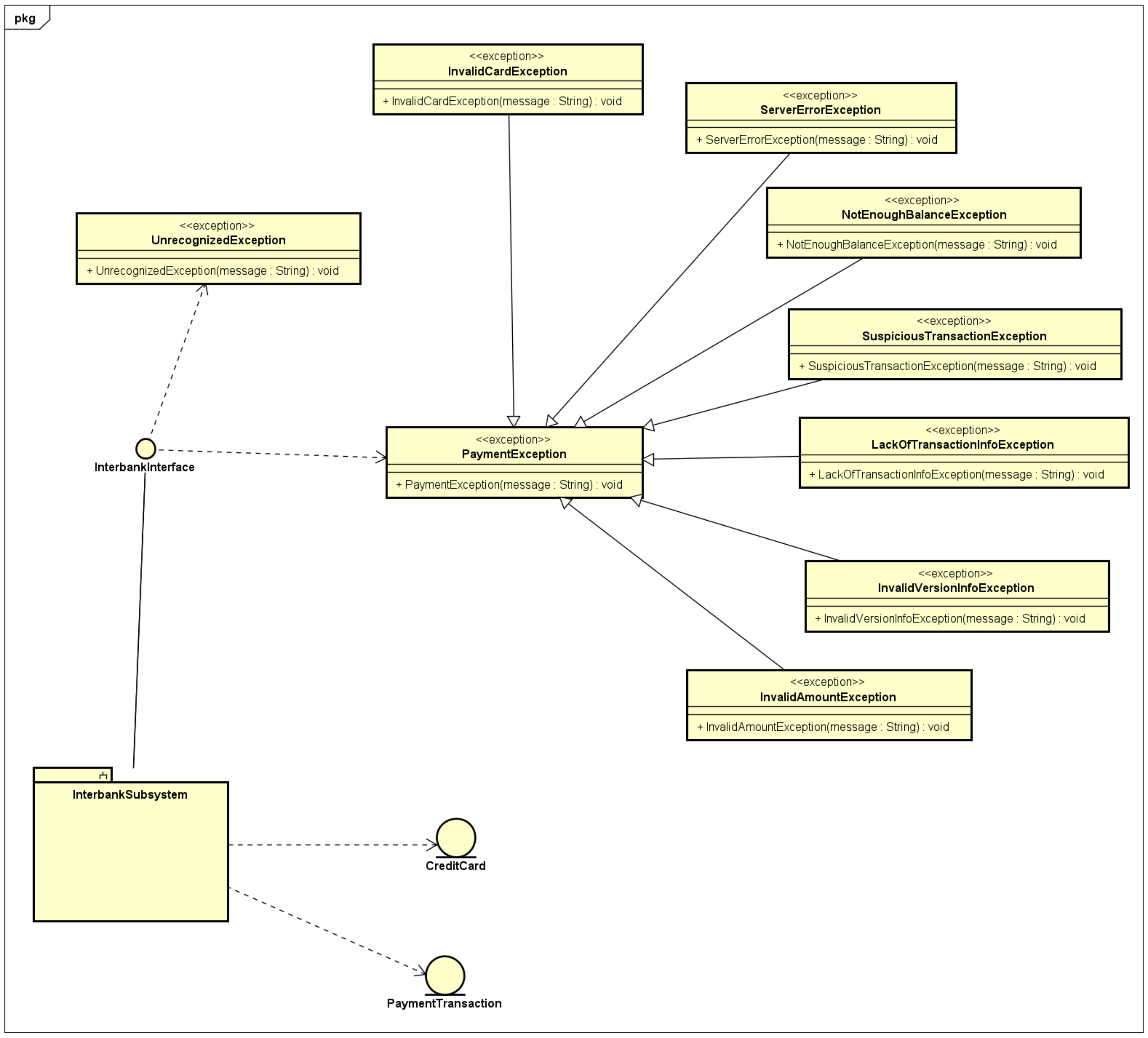


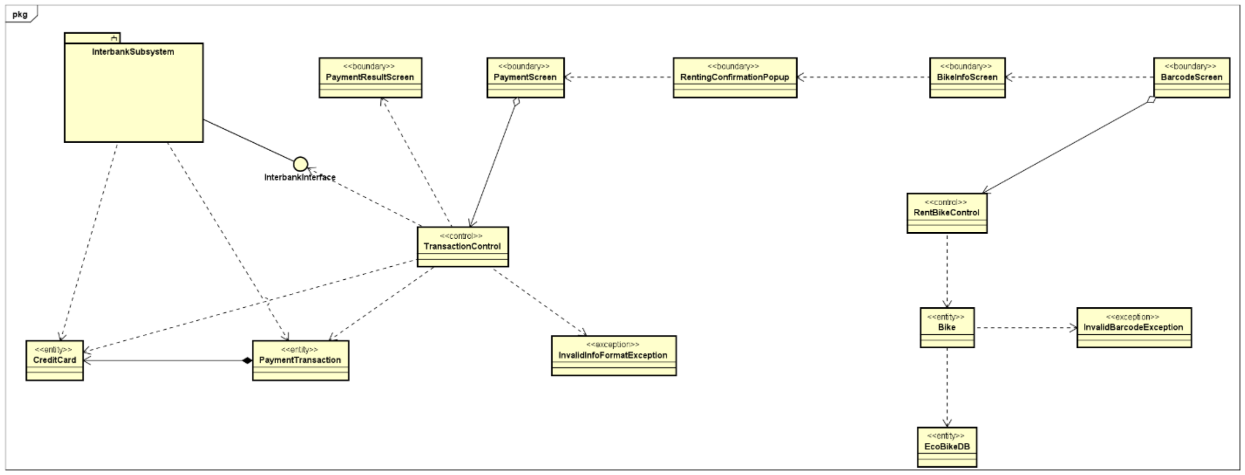
- common.exception
  - InvalidAmountException.java
  - InvalidBarcodeException.java
  - InvalidCardException.java
  - InvalidFormatException.java
  - InvalidVersionInfoException.java
  - LackOfTransactionInfoException.java
  - NotEnoughBalanceException.java
  - PaymentException.java
  - ServerErrorException.java
  - SuspiciousTransactionException.java
  - UnrecognizedException.java
- controller
  - RentBikeControl.java
  - TransactionControl.java
- entity.bike
  - Bike.java
- entity.db
  - EcoBikeDB.java
- entity.payment
  - CreditCard.java
  - PaymentTransaction.java



- subsystem
  - InterbankInterface.java
- subsystem.interbank.boundary
  - InterbankBoundary.java
- subsystem.interbank.controller
  - InterbankSubsystemControl.java
- subsystem.interbank.entity
  - Request.java
  - Response.java
- views.bikeinfo
  - BikeInfoScreen.java
- views.payment
  - PaymentResultScreen.java
  - PaymentScreen.java
- views.rentbike
  - BarcodeScreen.java
  - RentingConfirmationPopup.java

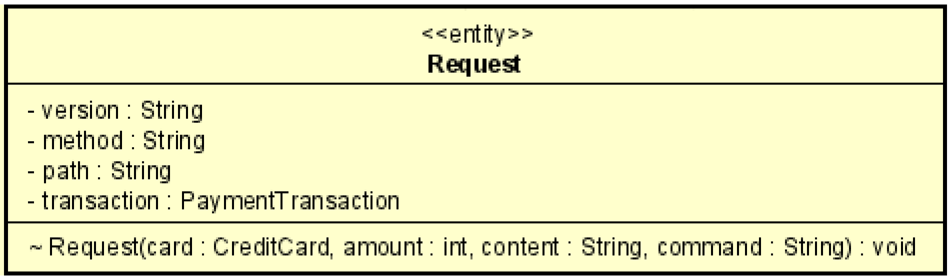
2. Class Diagram


pkg
<<exception>>
InvalidCardException
+ InvalidCardException(message : String) : void
<<exception>>
ServerErrorException
+ ServerErrorException(message : String) : void
<<exception>>
NotEnoughBalanceException
+ NotEnoughBalanceException(message : String) : void
<<exception>>
UnrecognizedException
+ UnrecognizedException(message : String) : void
<<exception>>
SuspiciousTransactionException
+ SuspiciousTransactionException(message : String) : void
InterbankInterface
<<exception>>
PaymentException
+ PaymentException(message : String) : void
<<exception>>
LackOfTransactionInfoException
+ LackOfTransactionInfoException(message : String) : void
<<exception>>
InvalidVersionInfoException
+ InvalidVersionInfoException(message : String) : void
<<exception>>
InvalidAmountException
+ InvalidAmountException(message : String) : void
InterbankSubsystem
CreditCard
PaymentTransaction



3. Class Design

3.1 Request


<<entity>>
Request
- version : String
- method : String
- path : String
- transaction : PaymentTransaction
~ Request(card : CreditCard, amount : int, content : String, command : String) : void

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	version	String	1.0.1	Version use to request Interbank
2	method	String	PATCH	Method use to request Interbank
3	path	String	NULL	Path to request Interbank
4	transaction	PaymentTransaction	NULL	The payment transaction to request Interbank

Operation

#	*Name*	*Description (purpose)*
1	Request	Constructor for class Request

Parameter:

- card – CreditCard used for the transaction
- amount – amount of money used for the transaction
- content – content of the transaction
- command – command for the Interbank (either “pay” or “refund”)

Exception:

- None

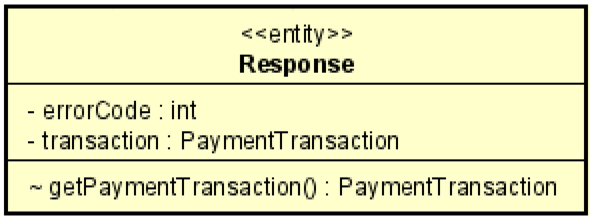
Method

None

State

None

3.2 Response


<<entity>>
Response
- errorCode : int
- transaction : PaymentTransaction
~ getPaymentTransaction() : PaymentTransaction

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	errorCode	int	NULL	Describe the result of the transaction
2	transaction	PaymentTransaction	NULL	The transaction that was requested

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	getPaymentTransaction	PaymentTransaction	To get the transaction that was requested

Parameter:

- None

Exception:

- PaymentException if the transaction is fail and errorCode is recognized
- UnrecognizedException if the transaction is fail and the errorCode is unrecognizable

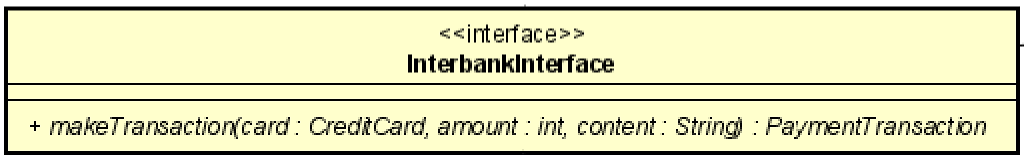
Method

None

State

None

3.3 InterbankInterface


<<interface>>
InterbankInterface
+ makeTransaction(card : CreditCard, amount : int, content : String) : PaymentTransaction

Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	makeTransaction	PaymentTransaction	Perform payment transaction

Parameter:

- card – the credit card that was used for the transaction
- amount – the amount of money for transact
- content – the content of the transaction

Exception:

- PaymentException if the transaction is fail and errorCode is recognized

- UnrecognizedException if the transaction is fail and the errorCode is unrecognizable

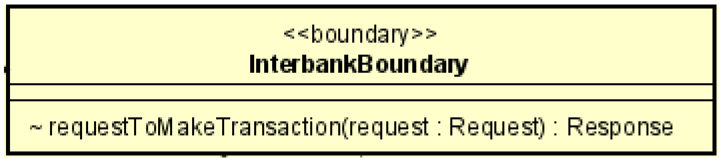
Method

None

State

None

3.4 InterbankBoundary



Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	requestToMakeTransaction	Response	Send the transaction request to Interbank

Parameter:

- request – the transaction request that need to be send to Interbank

Exception:

- None

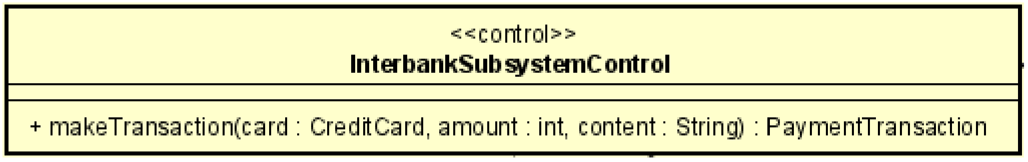
Method

None

State

None

3.5 InterbankSubsystemControl



<<control>>
InterbankSubsystemControl
+ makeTransaction(card : CreditCard, amount : int, content : String) : PaymentTransaction

Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*

1	makeTransaction	PaymentTransaction	Perform payment transaction

Parameter:

- card – the CreditCard use for the transaction
- amount – amount for transaction (positive number to pay and negative number to refund)
- content – the content of the transaction

Exception:

- PaymentException if the transaction is fail and errorCode is recognized
- UnrecognizedException if the transaction is fail and the errorCode is unrecognizable

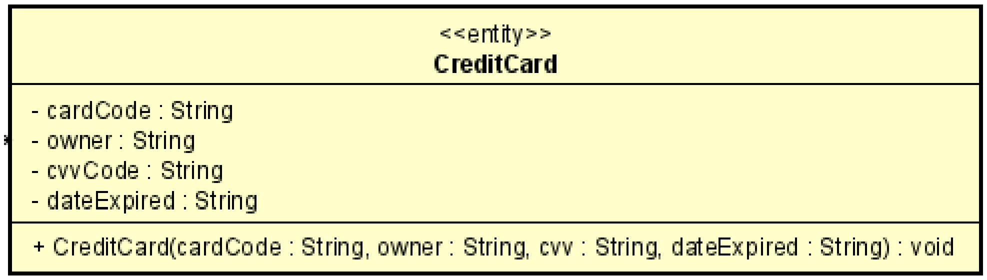
Method

None

State

None

3.6 CreditCard



Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	cardCode	String	NULL	The code of the card
2	owner	String	NULL	The owner of the card
3	cvvCode	String	NULL	The security code of the card
4	dateExpired	String	NULL	The expired date of the card in the format of mmyy

Operation

#	*Name*	*Description (purpose)*
1	CreditCard	The constructor for class CreditCard

Parameter:

- cardCode – the code of the card
- owner – the owner of the card
- cvv – the security code of the card

- dateExpired – the expired date of the card in the format of mmyy

Exception:

- None

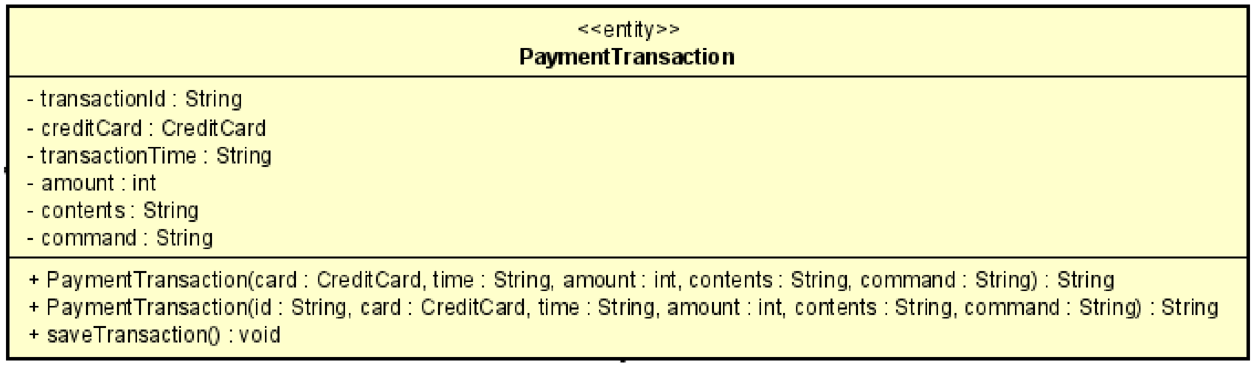
Method

None

State

None

3.7 PaymentTransaction



- transactionId : String
- creditCard : CreditCard
- transactionTime : String
- amount : int
- contents : String
- command : String

+ PaymentTransaction(card : CreditCard, time : String, amount : int, contents : String, command : String) : String
+ PaymentTransaction(id : String, card : CreditCard, time : String, amount : int, contents : String, command : String) : String
+ saveTransaction() : void

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	transactionId	String	NULL	The unique identification of the transaction
2	creditCard	CreditCard	NULL	The credit card used for the transaction
3	transactionTime	String	NULL	The time the transaction was made
4	amount	int	NULL	The amount used for transaction
5	contents	String	NULL	The content of the transaction
6	command	String	NULL	The command that was used for the transaction

Operation

#	*Name*	*Description (purpose)*
1	PaymentTransaction	The constructor for class PaymentTransaction using card, transaction time, amount of money in the transaction, transaction content and command used in the transaction

Parameter:

- card – the credit card used for the transaction
- time – the time the transaction was made

- amount – the amount used for transaction
- contents – the content of the transaction
- command – the command used in the transaction (either "pay" or "refund")

Exception:

- None

#	*Name*	*Description (purpose)*
2	PaymentTransaction	The constructor for class PaymentTransaction using transaction id, card, transaction time, amount of money in the transaction, transaction content and command used in the transaction

Parameter:

- Id – the identification of the transaction
- card – the credit card used for the transaction
- time – the time the transaction was made
- amount – the amount used for transaction
- contents – the content of the transaction
- command – the command used in the transaction (either "pay" or "refund")

Exception:

- None

Method

None

State

None

#	*Name*	*Return type*	*Description (purpose)*
3	saveTransaction	void	Record the transaction into the database

Parameter:

- None

Exception:

- None

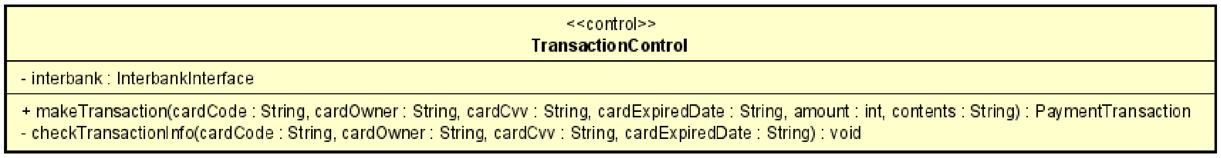
Method

None

State

None

3.8 TransactionControl



<<control>>
TransactionControl

- interbank : InterbankInterface

+ makeTransaction(cardCode : String, cardOwner : String, cardCvv : String, cardExpiredDate : String, amount : int, contents : String) : PaymentTransaction
- checkTransactionInfo(cardCode : String, cardOwner : String, cardCvv : String, cardExpiredDate : String) : void

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	interbank	InterbankInterface	New InterbankSubsystemControl()	Represent the interbank

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	makeTransaction	PaymentTransaction	Perform payment transaction

Parameter:

- cardCode – the code of the credit card used in the transaction
- cardOwner – the owner of the credit card used in the transaction
- cardCvv – the security code of the credit card used in the transaction
- cardExpiredDate – the expired date of the credit card used in the transaction
- amount – the amount of money in the transaction (positive to pay and negative to refund)
- contents – the contents of the transaction

Exception:

- PaymentException if the transaction fail and the error code is recognizable
- UnrecognizedException if the transaction fail and the error code is unrecognizable

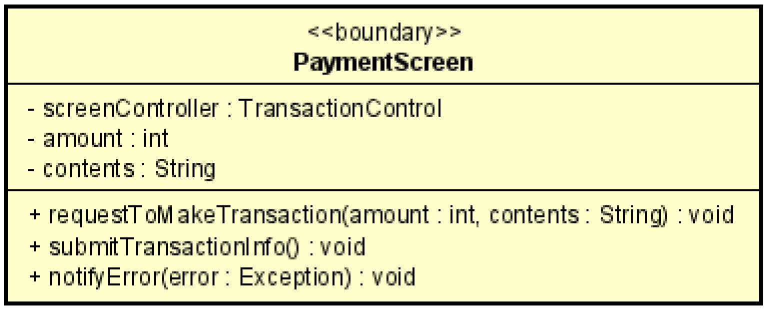
Method

- checkTransactionInfo: use to check the format of the transaction info, if the transaction info is not in the correct format then throw InvalidInfoFormatException. The correct format should be: cardCode is a string of digits, cardOwner is a string of letter and not null, cardCvv contain exactly 3 digits, cardExpiredDate is in format of mmyy.

State

None

3.9 PaymentScreen



<<boundary>>
PaymentScreen
- screenController : TransactionControl
- amount : int
- contents : String
+ requestToMakeTransaction(amount : int, contents : String) : void
+ submitTransactionInfo() : void
+ notifyError(error : Exception) : void

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	screenController	TransactionControl	New TransactionControl()	The controller of the screen
2	amount	int	NULL	The amount of money in the payment transaction
3	contents	String	NULL	The contents of the payment transaction

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	requestToMakeTransaction	void	Make this screen appear in order to make a payment transaction

Parameter:

- amount – the amount of money use in the transaction (positive for pay and negative for refund)
- contents – the contents of the transaction

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
2	submitTransactionInfo	void	Submit the transaction info filled in the screen to the controller in order to make payment transaction

Parameter:

- None

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*

3	notifyError	void	Notify the user error in case an exception happen

Parameter:

- error – The exception that need to inform user

Exception:

- None

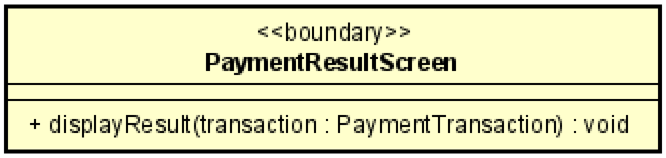
Method

None

State

None

3.10 PaymentResultScreen



Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	displayResult	void	Display the succesful payment transaction with the transaction info

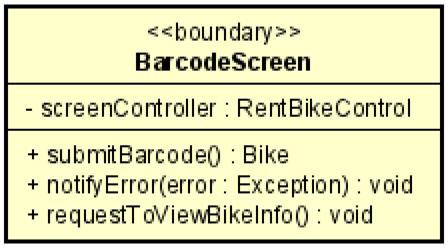
Parameter:

- transaction – The transaction that was successfully finish

Exception:

- None

3.11 BarcodeScreen


<<boundary>>
BarcodeScreen
- screenController : RentBikeControl
+ submitBarcode() : Bike
+ notifyError(error : Exception) : void
+ requestToViewBikeInfo() : void

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	screenController	RentBikeControl	New RentBikeControl()	The controller of the screen

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	submitBarcode	Bike	Submit the barcode that the user input to the controller in order to get the corresponding bike

Parameter:

- None

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
2	notifyError	void	Notify the user error in case an exception happen

Parameter:

- error – The exception that need to inform user

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
3	requestToViewBike	void	Show the user the info of the bike with the correspond barcode

Parameter:

- None

Exception:

- InvalidBarcodeException if there are no bike with the correspond barcode

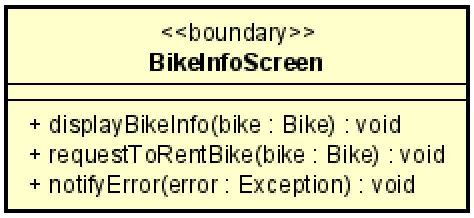
Method

None

State

None

3.12 BikeInfoScreen



<<boundary>>
BikeInfoScreen
+ displayBikeInfo(bike : Bike) : void
+ requestToRentBike(bike : Bike) : void
+ notifyError(error : Exception) : void

Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	displayBikeInfo	void	Show the information of a bike

Parameter:

- bike – The bike that we want to show information

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
2	requestToRentBike	void	Process to rent a bike

Parameter:

- bike – The bike you wish to rent

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
3	notifyError	void	Notify the user error in case an exception happen

Parameter:

- error – The exception that need to inform user

Exception:

- None

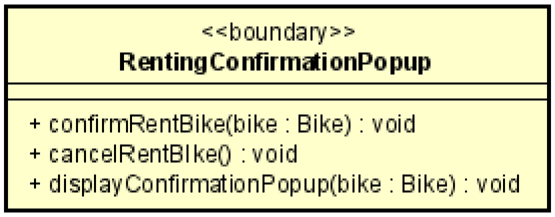
Method

None

State

None

3.13 Renting confirmation popup


<<boundary>>
RentingConfirmationPopup
+ confirmRentBike(bike : Bike) : void
+ cancelRentBIke() : void
+ displayConfirmationPopup(bike : Bike) : void

Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	confirmRentBike	void	Confirm to rent a bike

Parameter:

- bike – The bike you wish to rent

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
2	cancelRentBike	void	Cancel the rent bike process

Parameter:

- None

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
3	displayConfirmationPopup	void	Display this screen with the information of the bike you wish to rent

Parameter:

- bike – The bike you wish to rent

Exception:

- None

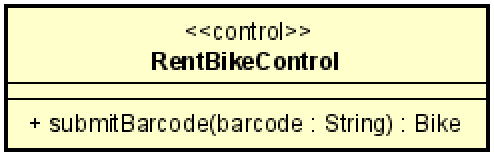
Method

None

State

None

3.14 RentBikeControl



Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	submitBarcode	Bike	Take the barcode into the database to get the bike with that corresponding barcode

Parameter:

- barcode - the barcode (identification) of the bike we would like to get from the database

Exception:

- InvalidBarcodeException if there no bike with the correspond barcode in the database

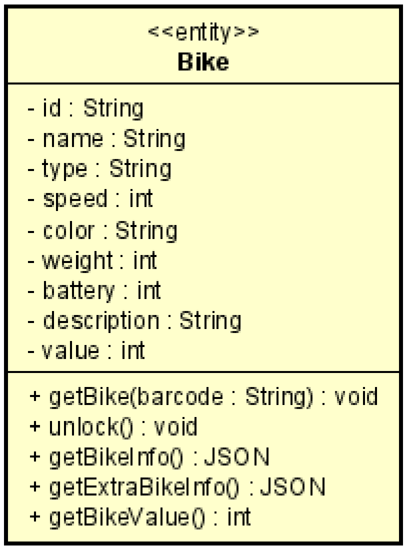
Method

None

State

None

3.15 Bike



Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	id	String	NULL	//
2	name	String	NULL	//
3	type	String	NULL	//
4	speed	int	NULL	//
5	color	String	NULL	//
6	weight	int	NULL	//
7	battery	int	NULL	//
8	description	String	NULL	//
9	value	int	NULL	//

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	getBike	void	Use the input barcode to change this class into the bike with that correspond barcode

Parameter:

- barcode - the barcode (identification) of the bike we would like to change this object into

Exception:

- InvalidBarcodeException if no bike with that barcode was found in the database

#	*Name*	*Return type*	*Description (purpose)*
2	unlock	void	Unlock this bike

Parameter:

- None

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
3	getBikeInfo	JSON	Get the bike attributes

Parameter:

- None

Exception:

- None

#	*Name*	*Return type*	*Description (purpose)*
4	getExtraBikeInfo	JSON	Get EBike info, only work with EBike

Parameter:

- None

Exception:

- NotEBikeException if bike is not EBike

#	*Name*	*Return type*	*Description (purpose)*
5	getBikeValue	int	Get the bike value

Parameter:

- None

Exception:

- None

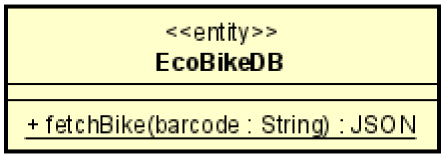
Method

None

State

None

3.16 EcoBikeDb



Attribute

None

Operation

#	*Name*	*Return type*	*Description (purpose)*
1	fetchBike	JSON	Using barcode to get the corresponding bike information from the database

Parameter:

- barcode – the barcode (identification) of the bike we would like to get from the database

Exception:

- InvalidBarcodeException if no bike with that barcode was found in the database

Method

None

State

None