Class Design - Model

1. Create Initial Design Classes

- 📁 **model**
  - ⊞ 📁 bike
  - 📁 db
    - ⊞ Ⓠ EBRDB
  - 📁 dock
    - ⊞ Ⓠ Dock
    - ⊞ Ⓠ DockManager
  - 📁 invoice
    - ⊞ Ⓠ Invoice
    - ⊞ Ⓠ InvoiceManager
  - 📁 payment
    - 📁 creditCard
      - ⊞ Ⓠ CreditCard
      - ⊞ Ⓠ CreditCardManager
    - 📁 transaction
      - ⊞ Ⓠ PaymentTransaction
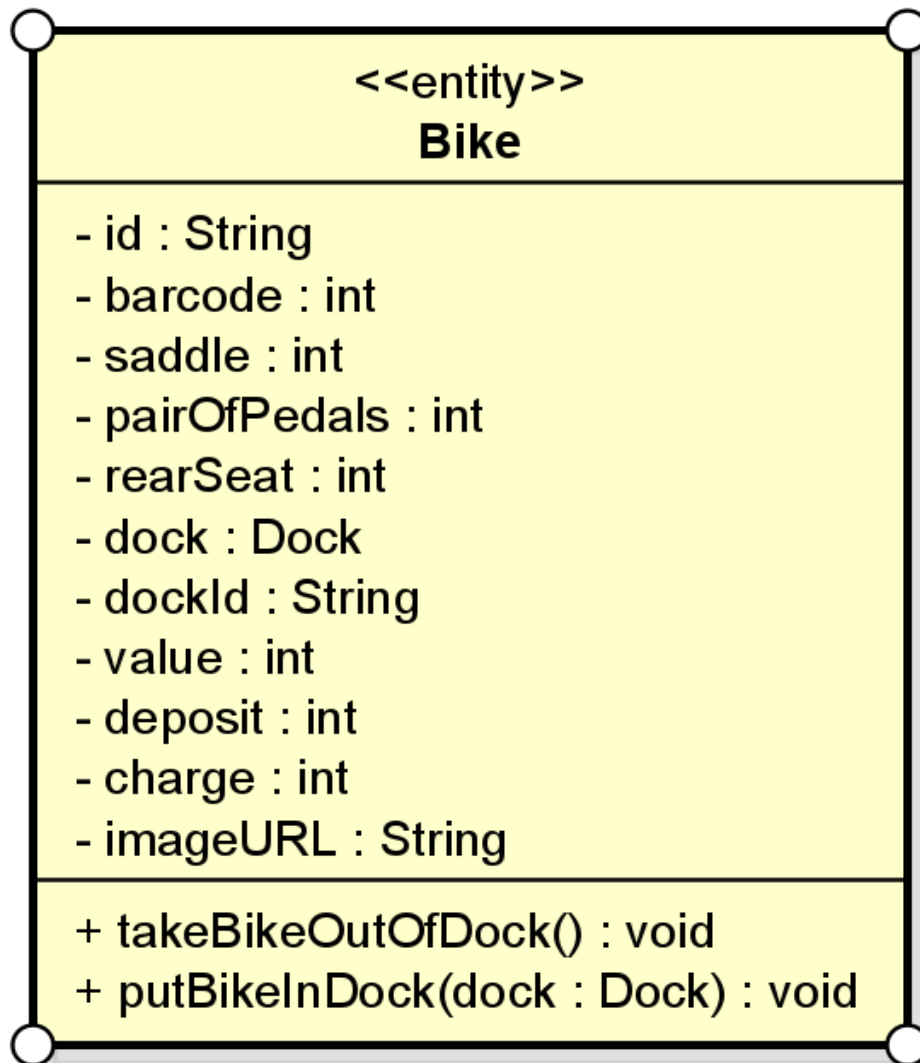   - ⊞ Ⓠ PaymentTransactionManager
  - 📁 session
    - ⊞ Ⓠ Session
    - ⊞ Ⓠ SessionManager
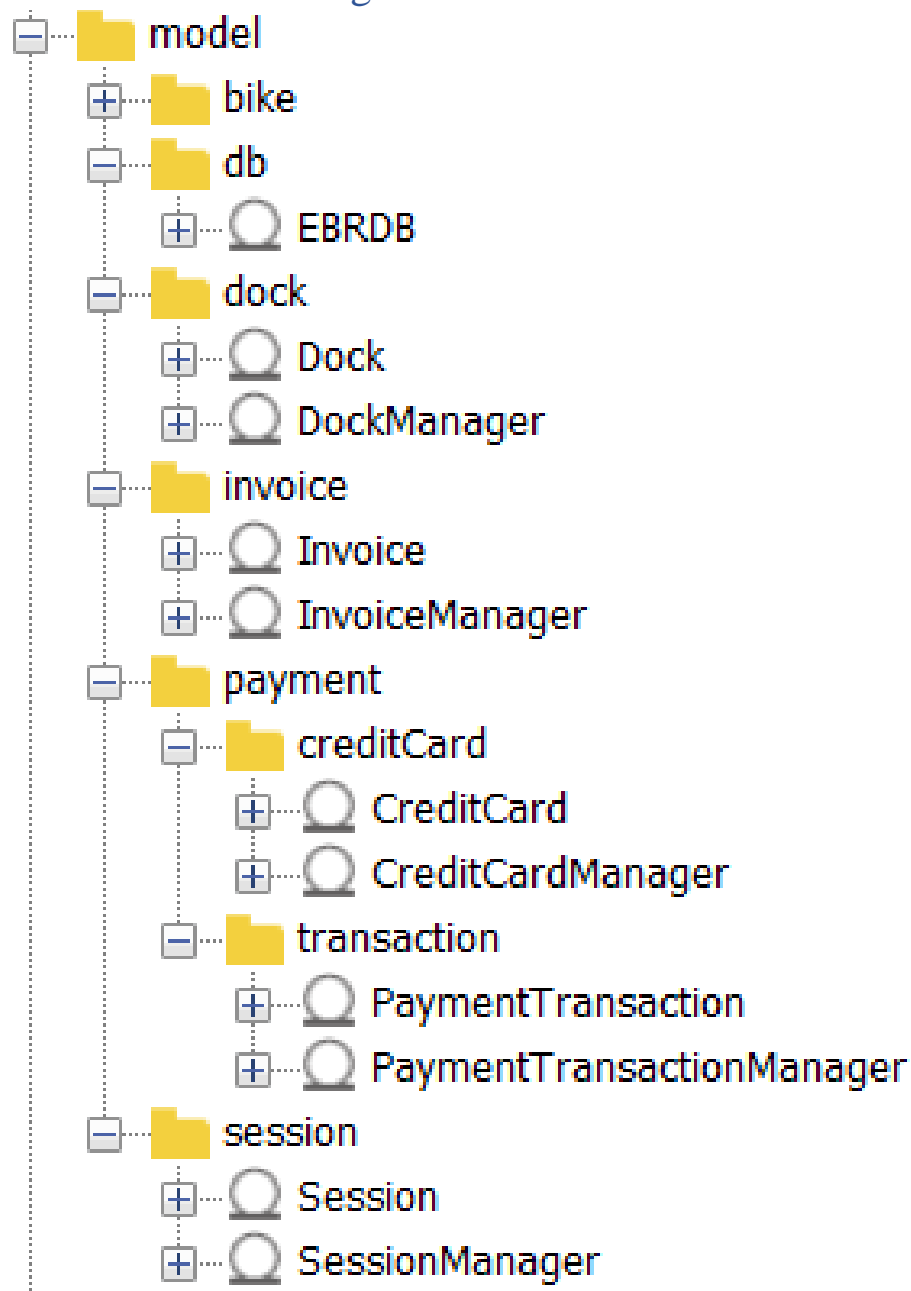
2. Define Relationships Between Classes
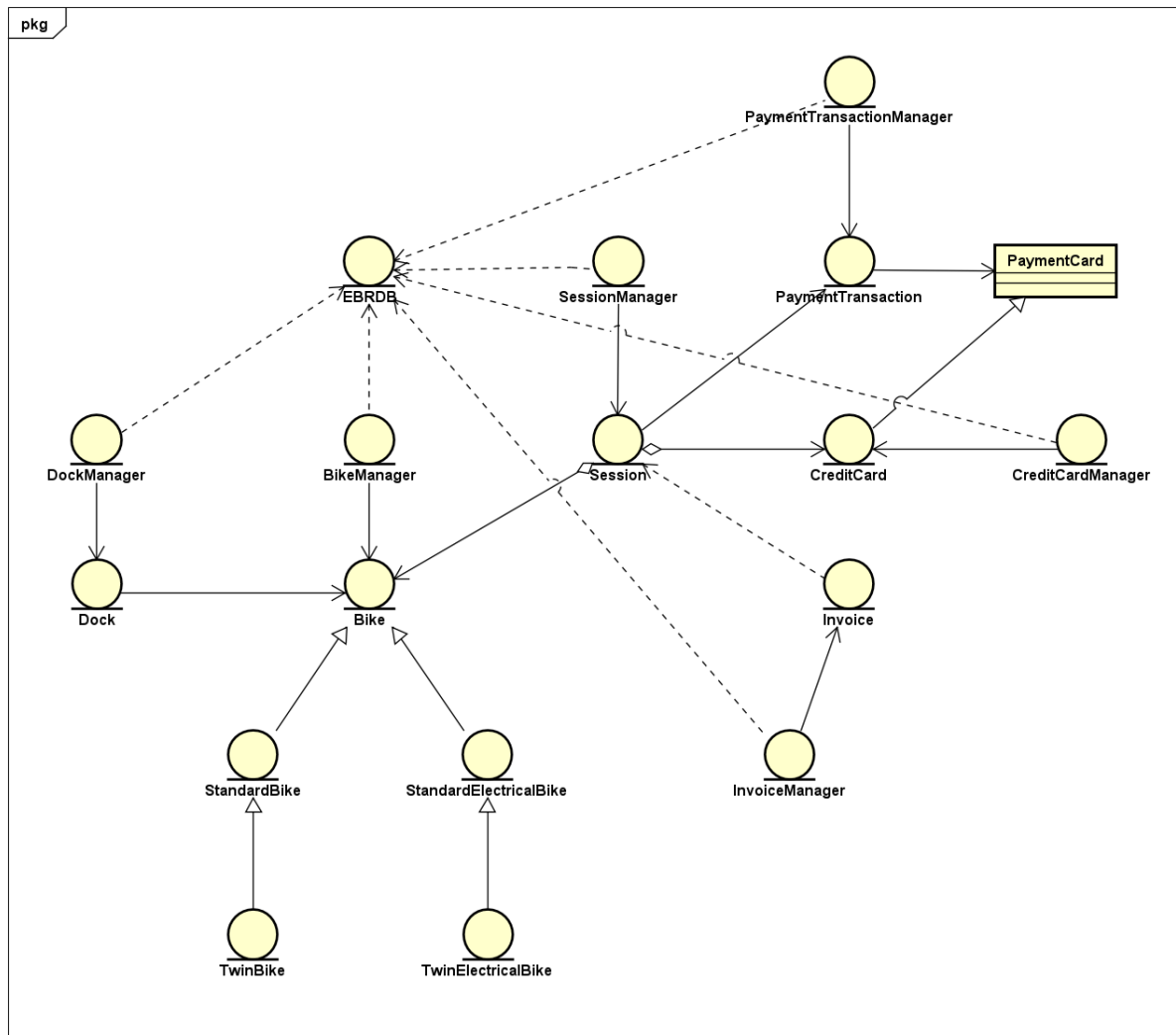
3. Class Design

3.1. Class "Bike"

Attribute

#	Name	Data type	Default value	Description
1	id	String	NULL	Bike id
2	barcode	int	NULL	barcode
3	saddle	int	NULL	saddle
4	pairOfPedals	int	NULL	Pair of pedals
5	rearSeat	int	NULL	Number of rear seats
6	dock	Dock	NULL	dock
7	dockId	String	NULL	Dock id
8	value	int	NULL	value
9	doposit	int	NULL	deposit
10	charge	int	NULL	charge
11	imageURL	String	NULL	Image URL

Operation

#	Name	Return type	Desciption
1	takeBikeOutOfDock	void	Take bike out of dock
2	putBikeInDock	void	Put bike to a dock

Parameter:

      dock

Exception:

      None

Method

      None

State

      None

3.2. Class "BikeManager"
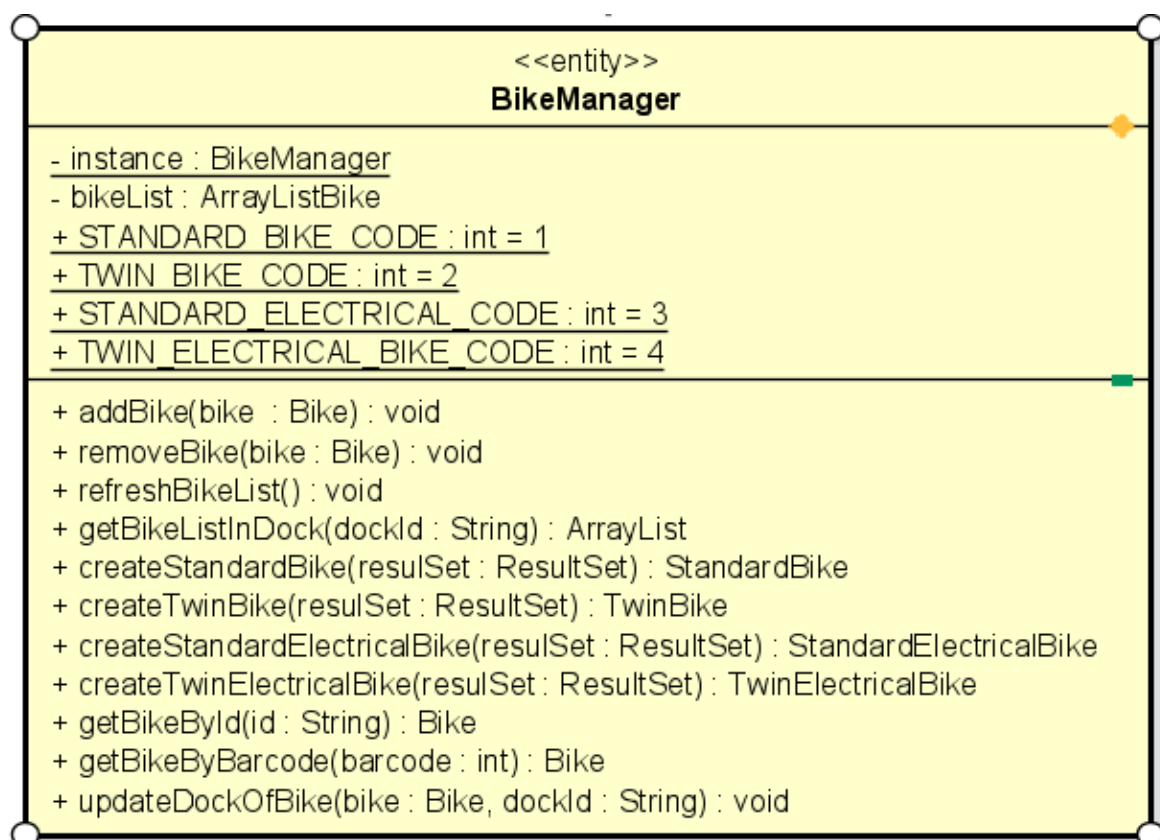
Attribute

#	Name	Data type	Default value	Description
1	instance	BikeManager	NULL	Bike instance
2	bikeList	ArrayList	NULL	Bike list
3	STANDARD_BIKE_CODE	int	1	Code of standard bike
4	TWIN_BIKE_CODE	int	2	Code of twin bike
5	STANDARD_ELECTRICAL_BIKE_CODE	int	3	Code of standard e bike
6	TWIN_ELEC TRICAL_BIKE_CODE	int	4	Code of twin e bike

Operation

#	Name	Return type	Desciption
1	addBike	void	Add bike
2	removeBike	void	Remove bike
3	refreshBikeList	void	Refresh bike list
4	getBikeListInDock	ArrayList	Get bike list in dock
5	createStandardBike	StandardBike	Create standard bike
6	createTwinBike	TwinBike	Create twin bike
7	createStandardEBike	StandardEBike	Create standard e bike
8	createTwinEBike	TwinEBike	Create twin e bike
9	getBikeById	Bike	Get bike by id
10	getBikeByBarcode	Bike	Get bike by barcode
11	updateDockOfBike	void	Update dock

Parameter:

    bike

    DockId

    ResultSet

    Id – bikeId

    barcode

Exception:

    None

Method

    None

State

> None

3.3. Class "StandardBike"

Attribute

> None

Operation

> None

Parameter:

> None

Exception:

> None

Method

> None

State

> None

3.4. Class "StandardElectricalBike"
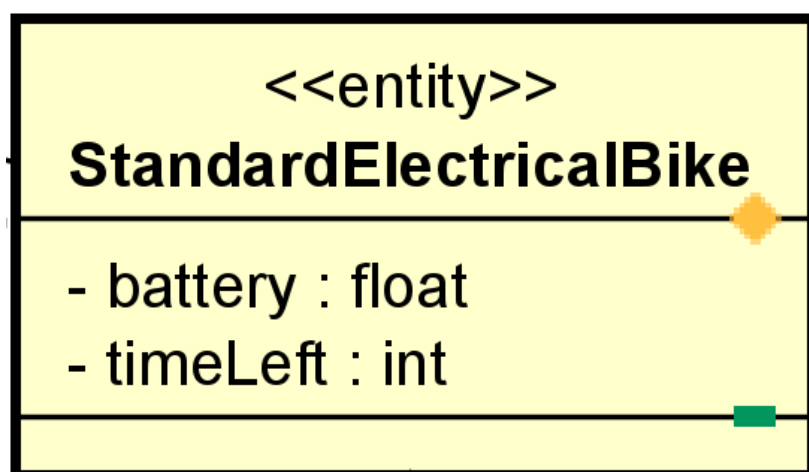
Attribute

#	Name	Data type	Default value	Description
1	battery	float	100.0	battery remained
2	timeLeft	int	NULL	usage time base on battery and bike

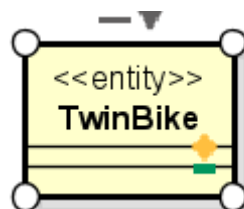**Operation**

  None

Parameter:

  None

Exception:

  None

Method

  None

State

  None

3.5. Class "TwinBike"

Attribute

  None
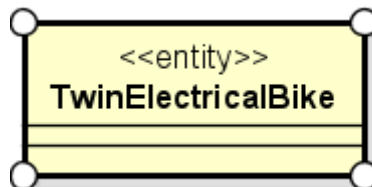
Operation

  None

Parameter:

  None

Exception:

  None

Method

None

State

None

3.6. Class "TwinElectricalBike"

Attribute

None

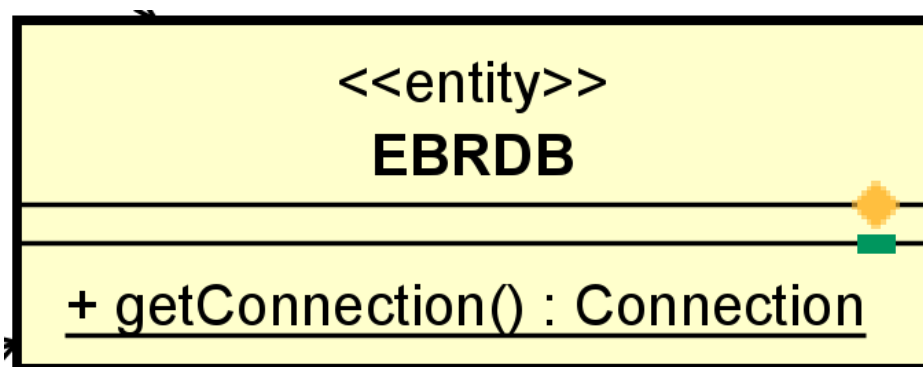**Operation**

None

Parameter:

None

Exception:

None

Method

None

State

None

3.7. Class "EBRDB"

Attribute

None

Operation

#	Name	Return type	Desciption
1	getConnection	Connect	get a connection to PostgreSQL database

Parameter:

None

Exception:

None

Method

None

State

None

3.8. Class "Dock"
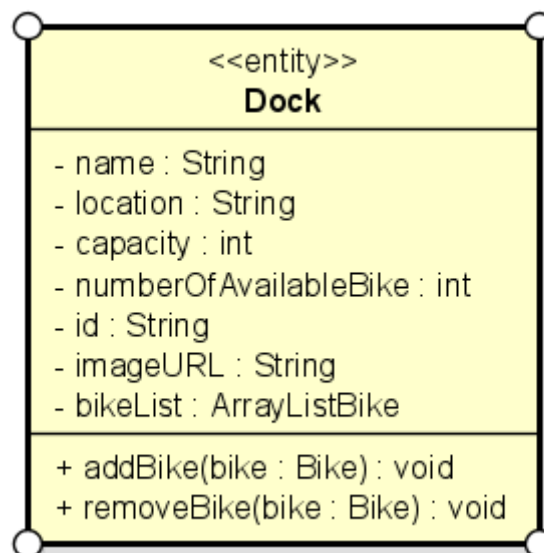
Attribute

#	Name	Data type	Default value	Description
1	name	String	NULL	Name of dock
2	location	String	NULL	Dock location

3	capacity	int	NULL	Dock size
4	numberOfAvailableBike	int	NULL	Number of available bikes
5	id	String	NULL	Dock id
6	imageURL	String	NULL	Dock image
7	bikeList	ArrayList<Bike>	NULL	Bike list in dock

Operation

#	Name	Return type	Desciption
1	addBike	void	Add bike to dock
2	removeBike	void	Remove bike out of dock

Parameter:

      bike

Exception:

      None

Method

      None

State

      None

3.9. Class "DockManager"
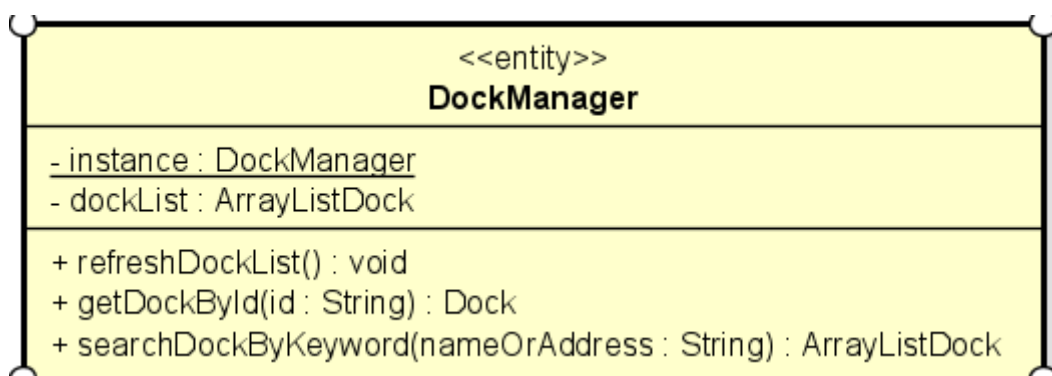
Attribute

#	Name	Data type	Default value	Description
1	instance	DockManager	NULL	Instance of a dock
2	dockList	ArrayList<Dock>	NULL	Dock list

Operation

#	Name	Return type	Desciption
1	refreshDockList	void	Refresh dock list
2	getDockById	Dock	Get dock by id
3	searchDockByKeyword	ArrayList<Dock>	Get docks by keyword

Parameter:

> Id - dock id

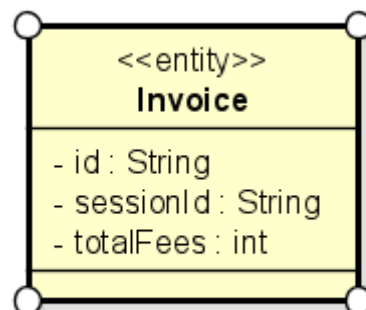> NameOfAddress – dock address

Exception:

> None

Method

> None

State

> None

3.10. Class "Invoice"

Attribute

#	Name	Data type	Default value	Description
1	id	String	NULL	Invoice id
2	sessionId	String	NULL	Session id
3	totalFees	int	NULL	Total charge

Operation

> None

Parameter:

None

Exception:

None

Method

None

State

None

3.11. Class "InvoiceManager"
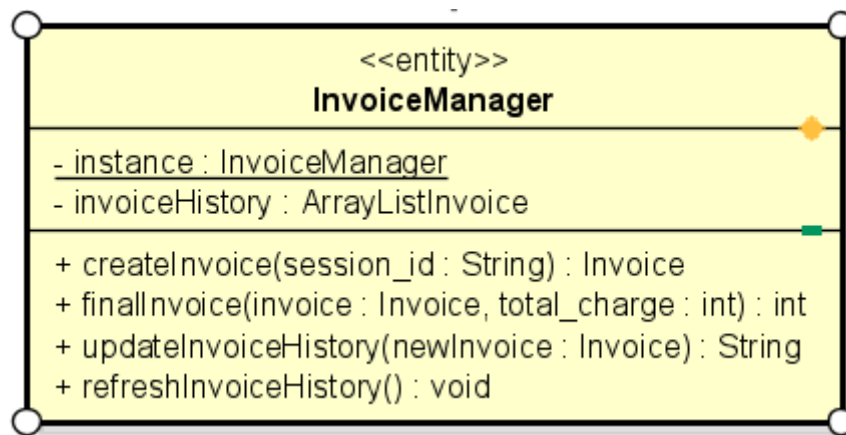
Attribute

#	Name	Data type	Default value	Description
1	instance	DockManager	NULL	Instance of a dock
2	invoiceHistory	ArrayList<Invoice>	NULL	Invoice History

Operation

#	Name	Return type	Desciption
1	createInvoice	Invoice	Create new invoice
2	finalInvoice	int	Complete invoice
3	updateInvoiceHistory	String	Update invoice history
4	refreshInvoiceHistory	void	Refresh invoice history

Parameter:

Session_id
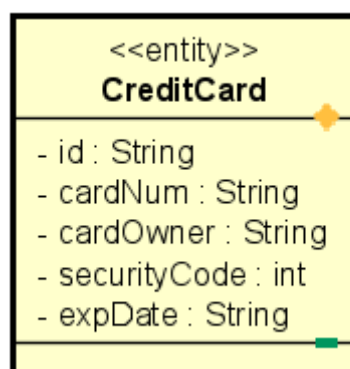
Invoice

Total_charge

newInvoice

Exception:

    None

Method

    None

State

    None

3.12. Class "CreditCard"

Attribute

#	Name	Data type	Default value	Description
1	id	String	NULL	Card id
2	cardNum	String	NULL	Card number
3	cardOwner	String	NULL	Card owner's name
4	securityCode	int	NULL	Card security code
5	expDate	String	NULL	Card exp date

Operation

    None

Parameter:

    None

Exception:

    None

Method

    None

State

None

3.13. Class "CreditCardManager"
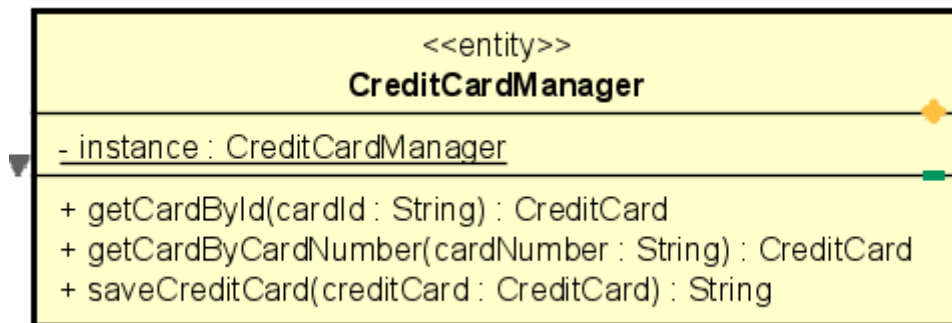
Attribute

#	Name	Data type	Default value	Description
1	instance	CreditCardManager	NULL	Instance of a card

Operation

#	Name	Return type	Desciption
1	getCardById	CreditCard	Get card by id
2	getCardByCardNumber	CreditCard	Get card by card number
3	saveCreditCard	String	Save credit card

Parameter:

CardId – card id
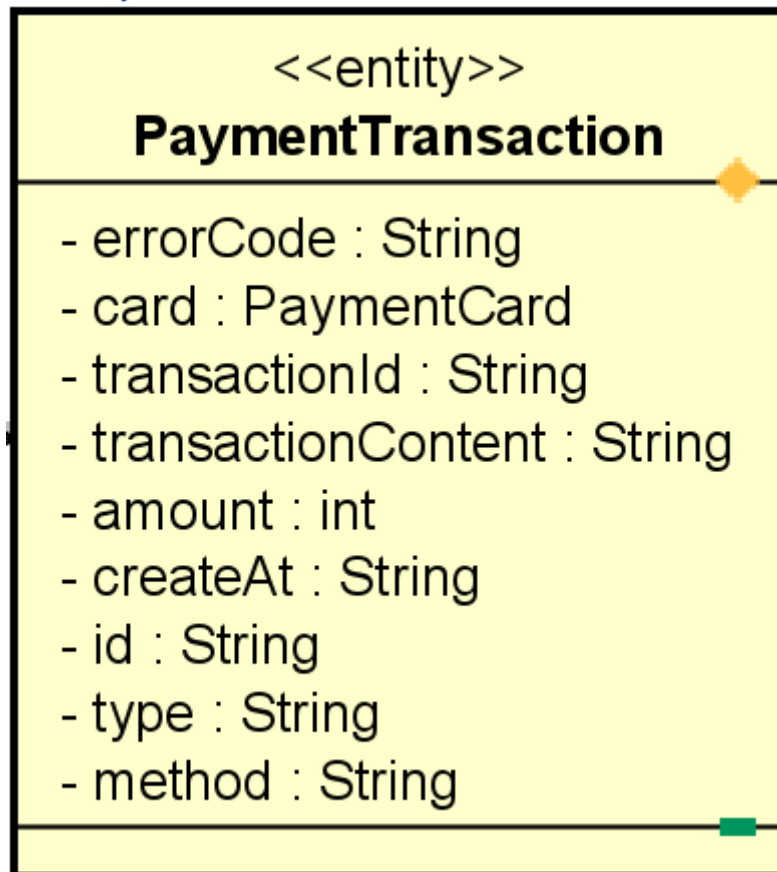
CardNumber – card number

creditCard

Exception:

None

Method

None

State

None

3.14. Class "PaymentTransaction"

Attribute

#	Name	Data type	Default value	Description
1	errorCode	String	NULL	Error code
2	card	PaymentCard	NULL	Credit card
3	transactionId	String	NULL	Transaction id
4	transactionContent	String	NULL	Contents of transaction
5	amount	int	NULL	Transaction amount
6	createAt	String	NULL	Transaction create time
7	id	String	NULL	PaymentTransaction id
8	type	String	NULL	Transaction type
9	method	String	NULL	Transaction method

Operation

None

Parameter:

None

Exception:

None

Method

None

State

None

3.15. Class "PaymentTransactionManager"

```
                        <<entity>>
               PaymentTransactionManager

  - instance : PaymentTransactionManager

  + getTransactionById(transactionId : String) : PaymentTransaction
  + savePaymentTransaction(paymentTransaction : PaymentTransaction) : String
```
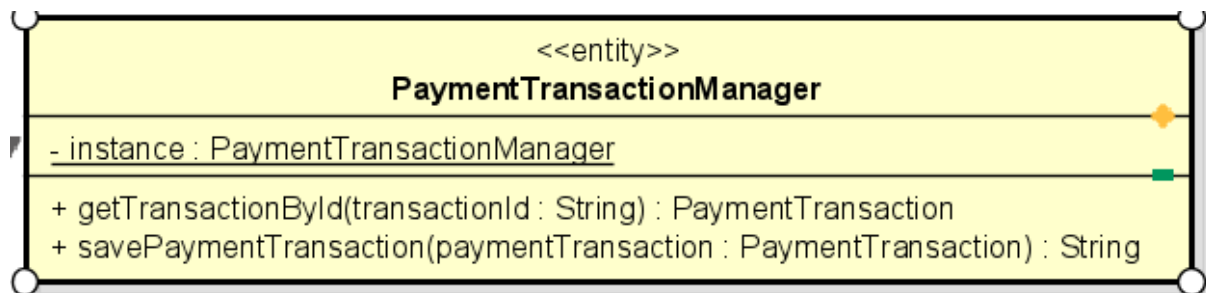
Attribute

#	Name	Data type	Default value	Description
1	instance	PaymentTransaction	NULL	Instance of transaction

Operation

#	Name	Return type	Desciption
1	getTransactionById	PaymentTransaction	Get transaction by Id
2	getCardByCardNumber	String	Save payment transaction

Parameter:

TransactionId – transaction id
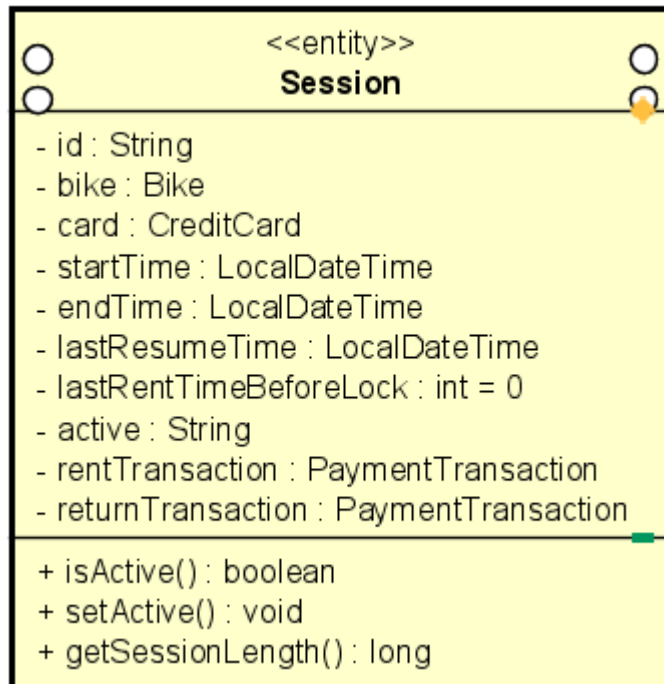
paymentTransaction

Exception:

None

Method

None

State

None

3.16. Class "Session"

Attribute

#	Name	Data type	Default value	Description
1	id	String	NULL	Session id
2	bike	Bike	NULL	Bike in session
3	card	CreditCard	NULL	Card in session
4	startTime	LocalDateTime	NULL	Start time
5	endTime	LocalDateTime	NULL	End time
6	lastResumeTime	LocalDateTime	NULL	Last resume time
7	lastRentTimeBeforeLock	int	0	Last rent time before lock
8	active	String	NULL	Session active or not
9	rentTransaction	PaymentTransaction	NULL	Rent transaction
10	returnTransaction	PaymentTransaction	NULL	Return transaction

Operation

#	Name	Return type	Desciption
1	isActive	boolean	Session is active or not
2	setActive	void	Set session active

3	getSessionLength	long	Get session length

Parameter:

None

Exception:

None

Method

None

State

None

3.17. Class "SessionManager"
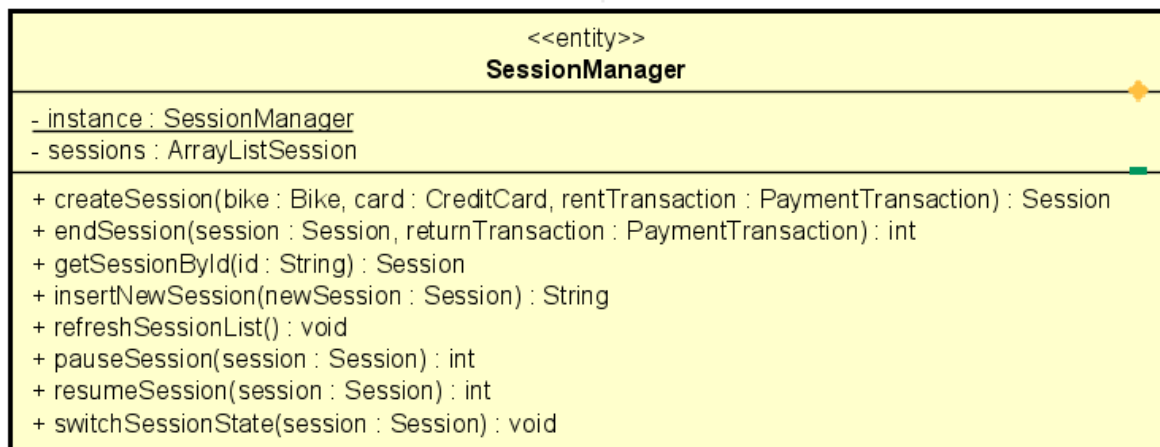
Attribute

#	Name	Data type	Default value	Description
1	instance	SessionManager	NULL	Session instance
2	sessions	ArrayList<Session>	NULL	Session list

Operation

#	Name	Return type	Desciption
1	createSession	Session	Create a session
2	endSession	int	Complete session
3	getSessionById	Session	Get session by id
4	insertNewSession	String	Insert new session to db
5	refreshSessionList	void	Refresh session list

| 6 | pauseSession | int | Pause session |
| 7 | resumeSession | int | Resume session |

Parameter:

    Bike

    Card

    PaymentTransaction

    Session

    newSession

Exception:

    None

Method

    None

State

    None

4. Class Diagram