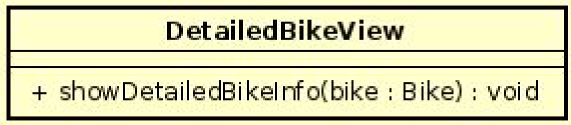
Class Design

1. Initial Design Classes

2. Define Relationships between classes

3. Class Design

3.1. Class “DetailedBikeView”:



Attribute

None

Operation

#	Name	Return type	Description (purpose)
1	showDetailedBikeInfo	Void	Display the detailed bike information view

Parameter

- bike – the bike selected

Exception

None

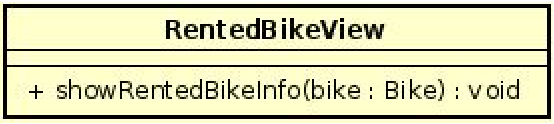
Method

None

State

None

3.2. Class “RentedBikeView”



Attribute

None

Operation

#	Name	Return type	Description (purpose)
1	showRentedBikeInfo	Void	Display the rented bike information view

Parameter

- bike – the bike selected

Exception

None

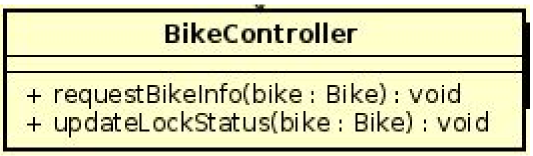
Method

None

State

None

3.3. Class “BikeController”


BikeController
+ requestBikeInfo(bike : Bike) : void
+ updateLockStatus(bike : Bike) : void

Attribute

None

Operation

#	Name	Return type	Description (purpose)
1	requestBikeInfo	Void	Request all related information for selected bike from database
2	updateLockStatus	Void	Change rented bike’s lock status

Parameter

- bike – the bike selected

Exception

None

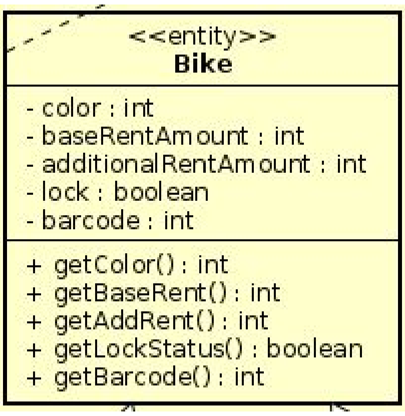
Method

None

State

None

3.4. Class “Bike”



Attribute

#	Name	Data type	Default value	Description
1	color	int	NULL	Color of the bike
2	baseRentAmount	int	NULL	Base amount of rent for the bike
3	additionalRentAmount	int	NULL	Additional amount of rent for the bike
4	lock	boolean	NULL	Lock status for rented bike
5	barcode	int	NULL	Barcode of the bike

Operation

#	Name	Return type	Description (purpose)
1	getColor	int	Request selected bike’s color
2	getBaseRent	int	Request selected bike’s base rent amount

3	getAddRent	int	Request selected bike’s additional rent amount
4	getLockStatus	boolean	Request selected bike’s lock status
5	getBarcode	int	Request selected bike’s barcode

Parameter

- bike – the bike selected

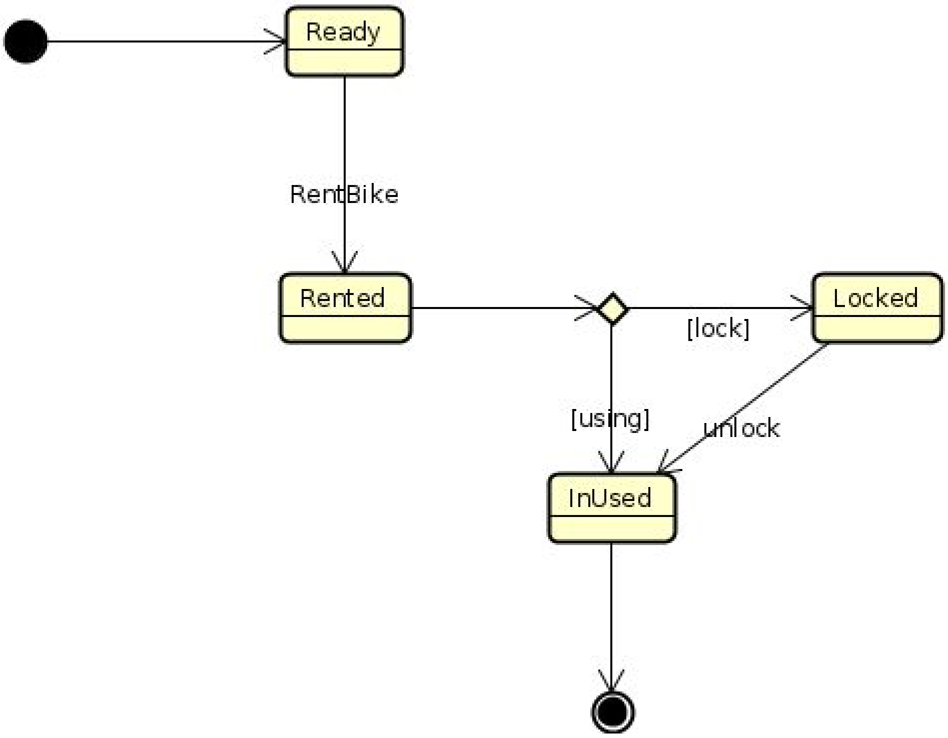
Exception

None

Method

None

State



4. Class Diagram