Islington College

islington college

Programming

CS4001

Coursework 1

Submitted By:

Student Name: Manish Giri

Student Id Number: 16034959

Group: L1N1

Date: July 28 2017

Submitted To:

Lecturer Name: Weenit Maharjan

Lecturer, IT Faculty.

Word Count: 1200

Semester: Spring

Table of Contents

Table of Tables:

Table of Figure:

Class Diagram

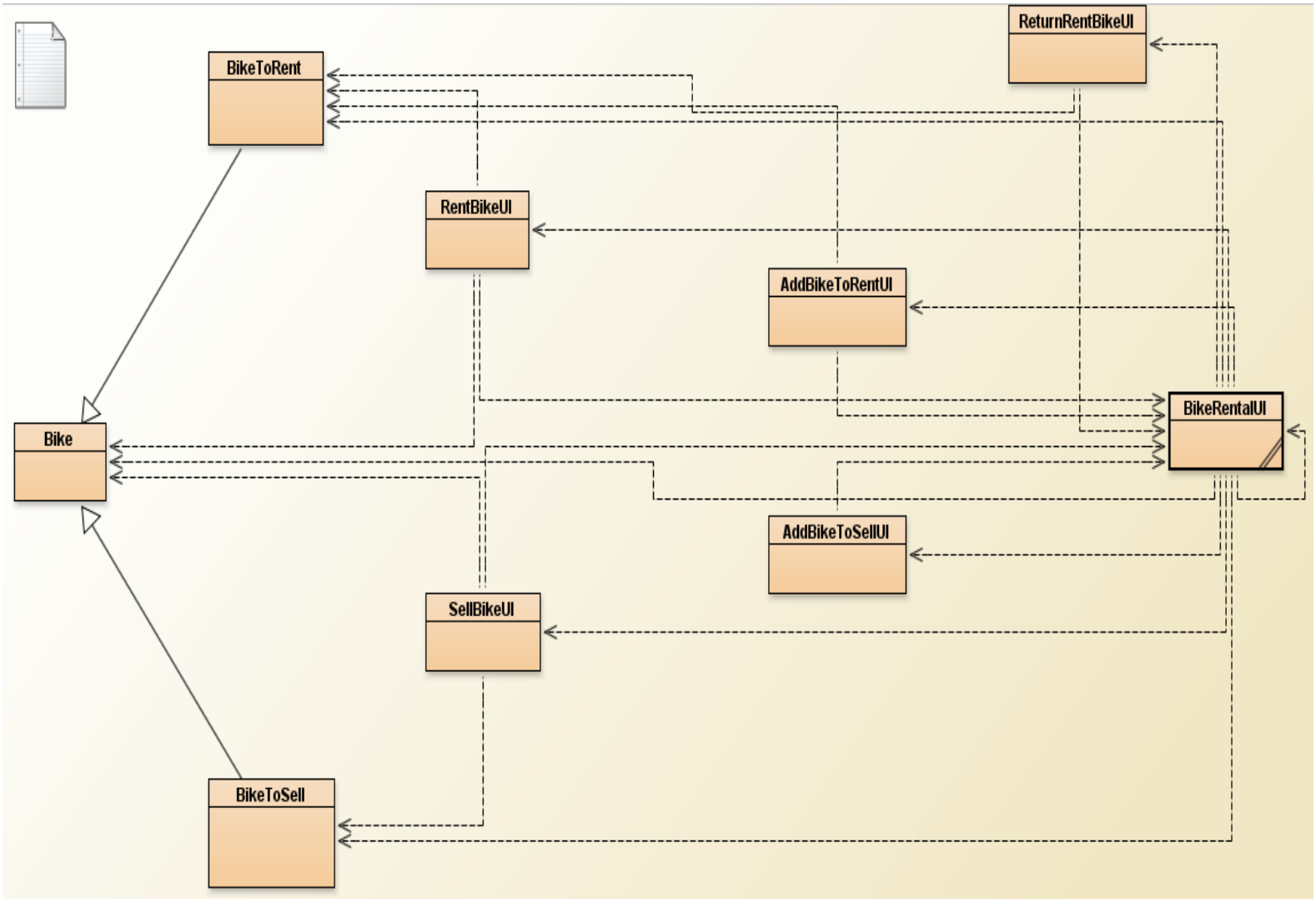

Figure 1 Class Diagram

Table 1 Bike Class Diagram

Bike
Attributes - bikeDescription: String - bikeManufacturer: String - customerName: String - contactNumber:String - customerEmail:String - bikeID:Integer
Methods + Bike(bikeDescription:String,bikeManufacturer:String,bikeID:Integer) + setCustomerName (customerName):void + setContactNumber (contactNumber):void + setCustomerEmail (customerEmail):void + getBikeDescription():String + getBikeManufacturer():String + getBikeID():Integer + getContactNumber():String + getCustomerEmail():String + void display():void

Table 2 BikeToRent Class Diagram

BikeToRent
Attributes - bikeHireDate: String - numberOfDays: int - dailyRate: int - totalRentalCost: int - bikeLoanStatus: boolean
Methods +BikeToRent(bikeDescription:String,bikeManufacturer:String, dailyRate:int, bikeId:Integer) +getBikeHireDate():String +getNumberOfDays():int +getDailyRate():int +getTotalRentalCost()int +getBikeLoanStatus():boolean + rentOutBike(customerName:String,contactNumber:String,customerEmail:String,ipbikeHireDate:String, ipnumberOfDays:int):void + makeBikeAvailable(): void

Table 3 BikeToSell Class Diagram

BikeToSell
Attributes - price: int - taxAmount: int - totalAmount: int - sellingDate: int - sellingStatus: boolean
Methods + BikeToSelli(bikeID: String, bikeDescription: String, price: int, taxAmount: int,BikeID:Integer) + getPrice(): int + getTaxAmount(): int + getTotalAmount(): int + getSellingDate(): String + SellingStatus(): boolean + bikeSellOut(String:customerName,String: contactNumber, String: customerEmail,String: ipsellingDate)

Table 4 BikeRentalUI Class Diagram

BikeRentalUI
Attributes -frame : JFrame -lblBikeCompany :Jlabel -btnAddBikeToRent : JButton -btnAddBikeToSell : JButton -btnRentBike : JButton -btnSellBike : JButton -btnReturn : JButton -btnDisplay : JButton
Methods +BikeRentalUI() +actionPerformed() : void +addBikeToRent(bikeDescription :String, bikeManufacturer :String, dailyRate :Integer, bikeID :Integer) +addBikeToSell(bikeDescription :String, bikeManufacturer :String, price :Integer, taxAmount :Integer, bikeID : Integer) +display() : void

Table 5 AddBikeToRentUI Class diagram

AddBikeToRentUI
Attributes -frame : JFrame -lblAddBikeToRent : JLabel -lblBikeId : JLabel -lblDescription: JLabel -lblDailyRate:JLabel -lblCompany:JLabel -txtBikeId : jtextfield -txtDescription :JTextField -txtDailyRate : JTextField -txtCompany : JTextField -btnClear : JButton -btnConfirm : JButton
Methods +AddBikeToRentUI() +BikeToRentalUI() : void +actionPerformed() : void

Table 6 AddBikeToSellUI Class Diagram

AddBikeToSellUI
Attributes -frame : JFrame -lblAddBikeToSell :JLabel -lblBikeId : JLabel -lblDescription : JLabel -lblPrice : JLabel -lblTaxRate : JLabel -lblCompany : JLabel -btnClear : JButton -btnConfirm : JButton -txtBikeId : JTextField -txtDescription : JTextField -txtPrice : JTextField -txtTaxRate : JTextField -txtCompany: JTextField
Methods +BikeToSellUI() +BikeToSellUI() :void +actionPerformed() : void

Table 7 RentBikeUI Class Diagram

RentBikeUI
Attributes -frame : JFrame -lblRentBike : JLabel - lblBikeId :Jlabel -lblDescription : Jlabel -lblCustomerName : Jlabel -lblContact : Jlabel -lblHireDate : Jlabel -lblDailyRate : Jlabel -lblTotalAmount: Jlabel -lblCompany: Jlabel -lblEmail: Jlabel -lblDays: Jlabel -btn check :button -btnCalculator : button -btnClear : button -btnConfirm : button -txtBikeId :text -txtDescription :text -txtCustomerName:text -txtContact:text -txtHireDate:text -txtDailyRate:text -txtTotalAmount :text -txtCompany :text -txtEmail:text -txtDays :text
Methods +RentBikeUI() +RentBikeUI() : void +actionPerformed() : void

Table 8 SellBikeUI Class Diagram

SellBikeUI
Attributes -frame : JFrame -lblSellBike : JLabel -lblBikeId: JLabel -lblDescription : JLabel - lblCustomerName : JLabel -lblContact : JLabel -lblSellDate : JLabel -lblPrice : JLabel -lblTotalAmount: JLabel -lblCompany: JLabel -lblEmail: JLabel -lblTaxRate: JLabel -btnCheck: JButton -btnClear : JButton -btnConfirm : JButton -txtBikeId : JTextField -txtDescription : JTextFieldtxtCustomerName : JTextField -txtContact: JTextField -txtSellDate : JTextField -txtPrice : JTextField -txtTotalAmount : JTextField -txtCompany: JTextField -txtEmail : JTextField -txtTaxRate : JTextField
Methods +SellBikeUI() +SellBikeUI() :void +actionPerformed() : void

Table 9 ReturnRentBikeUI Class Diagram

ReturnBikeUI
Attributes -frame : JFrame -lblReturnBike : JLabel -lblBikeId :JLabel -txtBikeId : JTextField -btnClear : JButton -btnConfirm : JButton

Pseudo code

A) Class: Bike

Method 1:

Bike(**Initialize through parameter** bikeDescription,bikeManufacturer and bikeID)

// initialise instance variables

this.bikeDescription **is set to** bikeDescription

this.bikeManufacturer **is set to** bikeManufacturer

this.bikeID **is set to** bikeID

customerName **is set to** " "

contactNumber **is set to** " "

customerEmail **is set to** " "

Method 2:

Set setCustomerName **through parameter using** customerName

customerName **is set to** customerName

Method 3:

Set setContactNumber **through parameter using** contactNumber

contactNumber **is set to** contactNumber

Method 4:

Set setCustomerEmail **through parameter using** customerEmail

customerEmail **is set to** customerEmail

Method 5:

Call getCustomerName

Return customerName

Method 6:

Call getBikeId

Return BikeID

Method 7:

Call getContactNumber

Return contactNumber

Method 8:

Call getCustomerEmail

Return customerEmail

Method 8:

Display()

nullString **is set to null**

empty **is set to new String**

Display "Description :" **with value of** bikeDescription

Display "Manufacturer :" **with value of** bikeManufacturer

if customerName **is not equal to** " "

Display "Customer's Name :" **with value of** customerName

if contactNumber **is not equal to** " "

Display "Customer's Contact Number :" **with value of** contactNumber

if customerEmail **is not equal to** " "

Display "Customer's Email :" **with value of** customerEmail

B) Class: BikeToRent

Method 1:

BikeToRent **Initialise through parameter**: bikeDescription, bikeID, dailyRate

Through inheritance with bike class bikeID,bikeDescription,bikeManufacturer

this.dailyRate **is set to** dailyRate

this.bikeHireDate **is set to** ""

this.numberOfDays **is set to** 0

this.totalRentalCost **is set to** 0

this.loanStatus **is set to** false

Method 2:

Call getBikeHireDate()

Return bikeHireDate

Method 3:

Call getNumberOfDays()

Return numberOfDays

Method 4:

Call getDailyRate()

Return dailyRate

Method 5:

Call getTotalRentalCosti()

Return totalRentalCost

Method 8:

Call getBikeLoanStatus()

Return bikeLoanStatus

Method 9:

Set setBikeLoanStatus (**Initialize through parameter:** bikeLoanStatus)

this. bikeLoanStatus **is equals to** bikeLoanStatus

Method 10:

rentOutBike(**Initialize through parameter:**customerName,contactNumber,customerEmail,ipbikeHireDate,ipnumberOfDays)

if bikeLoanStatus **is not true**

this.setCustomerName(customerName)

this.setContactNumber(contactNumber)

this.setCustomerEmail(customerEmail)

this.bikeHireDate **is equals to** ipbikeHireDate

this.numberOfDays **is equals to** ipnumberOfDays

this.bikeLoanStatus **is equals to** true

this.totalRentalCost **is equals to** dailyRate * numberOfDays

Return true;

else

Display "Sorry!The bike is already on Rent."

Return false;

Method 10:

public void makeBikeAvailable()

if bikeLoanStatus **is not true**

Display "Oh! The bike is already available."

else

Call setCustomerName **with value** " "

Call setContactNumber **with value** " "

Call setCustomerEmail **with value** " "

this.numberOfDays **is equals to** 0;

this.bikeHireDate **is equals to** " "

bikeLoanStatus **is equals to** false

C) Class: BikeToSell

Method 1:

BikeToSell (**Initialize through parameter:** bikeID, bikeDescription,bikeManfacturer, price, taxAmount)

super(bikeID,bikeDescription,bikeManufacturer);

this.price **is equals to** price;

this.taxAmount **is equals to** taxAmount;

this.sellingStatus **is equals to** false;

this.sellingDate **is equals to** "";

Method 2:

Call getPrice()

Return price

Method 3:

Call int getTaxAmount()

Return taxAmount

Method 4:

Call getTotalAmount()

Return totalAmount;

Method 5:

Call getSellingDate()

Return sellingDate

Method 6:

Call isSellingStatus()

Return sellingStatus

Method 7:

bikeSellOut(**Initialize through parameter** customerName, contactNumber, customerEmail, ipsellingDate)

if sellingStatus **is not equal to true**

this.setCustomerName(customerName) ;

this.setContactNumber(contactNumber) ;

this.setCustomerEmail(customerEmail) ;

this.totalAmount **is equals to** price+taxAmount ;

this.sellingStatus **is equals to** true ;

this.sellingDate **is equals to** ipsellingDate;

Return true;

else

Display "Sorry! The bike is already sold."

Display "The bike was sold on: "**with value** sellingDate

Return false;

D) Class: BikeRentalUI

Method 1:

actionPerformed(ActionEvent e)

```
if btnAddBikeToRent is  clicked

    list = new ArrayList();

    new AddBikeToRentUI();

if btnRentBike is  clicked

    new RentBikeUI();

if btnSellBike is  clicked

    new SellBikeUI();

if btnAddBikeToSell is  clicked

    new AddBikeToSellUI()

if btnReturn is  clicked

    new ReturnRentBikeUI()

if btnDisplay is  clicked

    display()
```

Method 2:

static addBikeToRent (**Initialize through parameter:** bikeDescription,bikeManufacturer,dailyRate,bikeID)

```
list.add(new BikeToRent(bikeDescription,bikeManufacturer,dailyRate,bikeID));
```

Method 3:

static addBikeToSell(**Initialize through parameter:**bikeDescription,bikeManufacturer,price, taxAmount,bikeID)

```
list.add(new BikeToSell(bikeDescription,bikeManufacturer,price,taxAmount,bikeID));
```

Method 4:

display()

```
for (int i = 0; i < list.size(); i++)

    list.get(i).display();
```

E) Class: AddBikeToRentUI

Method 1:

actionPerformed(ActionEvent e)

if btnConfirm **is clicked**

try

if txtDescription **is not equal to** " " **AND** txtCompany **is not equal to** " " **AND** txtDailyRate **is not equal to** " "

BikeRentalUI.addBikeToRent(txtDescription.getText(),txtCompany.getText(),Integer.parseInt(txtDailyRate.getText()),Integer.parseInt(txtBikeId.getText()));

Show Message Dialog "The bike has been successfully added for rent

else

Show Message Dialog "You cannot leave the fields empty inorder to continue"

catch(NumberFormatException nfe)

Show Message Dialog " Error Message\nPlease enter a valid number","Error",JOptionPane.ERROR_MESSAGE **"**

else if btnClear **is clicked**

txtDescription.setText("");

txtDailyRate.setText("");

txtCompany.setText("");

F) Class: AddBikeToSellUI

Method 1:

actionPerformed(ActionEvent e)

if btnConfirm **is clicked**

try

if txtDescription **is not equal to** " " **AND** txtCompany **is not equal to** " " **AND** txtPrice **is not equal to** " " **AND** txtTaxRate **is not equal to**" " **AND** txtBikeID **is not equal to** " "

BikeRentalUI.addBikeToSell(txtDescription.getText(),txtCompany.getText(),Integer.parseInt(txtPrice.getText()),Integer.parseInt(txtTaxRate.getText()),Integer.parseInt(txtBikeId.getText()));

Show Message Dialog "The bike has been succesfully added for sale"

else

Show Message Dialog " You cannot leave the fields empty to continue "

catch(NumberFormatException nfe)

else if btnClear **is clicked**

txtDescription.setText("");

txtDailyRate.setText("");

txtCompany.setText("");

G) Class:RentBikeUI

Method 1:

```
actionPerformed(ActionEvent e)

  if btnCheck is clicked

        try

        bikeFound is set to false;

        bikeRentStatus is set to false;

        for (int i = 0; i < BikeRentalUI.list.size(); i++)

        if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))

                if(BikeRentalUI.list.get(i).getBikeId()incomparision Integer.parseInt(txtBikeId.getText()))

                        bike is equal to (BikeToRent) BikeRentalUI.list.get(i);

                        if(!bike.getBikeLoanStatus())

                                txtCompanyis set to(bike.getBikeManufacturer());

                                txtDescriptionis set to (bike.getBikeDescription());

                                txtDailyRateis set toInteger.toString(bike.getDailyRate()));

                                txtTotalAmountis set toInteger.toString(bike.getTotalRentalCost()));

                                bikeRentStatus is set to true;

                        bikeFound is set to true;

                            break;

                if bikeFound AND bikeRentStatus is not equal to true

                   ShowMessageDialog"The bike you searched for is not available for rent"

                if bikeFound is not equal to True

                   ShowMessageDialog "The bike you searched for is not available"

                 catch(NumberFormatException nfe)

                   ShowMessageDialog "Error Message\nPlease enter a valid number","Error"
```

```
else if btnClear is clicked
        txtBikeId.setText("");
      txtDescription.setText("");
      txtCustomerName.setText("");
      txtContact.setText("");
      txtHireDate.setText("");
      txtDailyRate.setText("");
      txtTotalAmount.setText("");
      txtCompany.setText("");
      txtEmail.setText("");
      txtDays.setText("");
else if btnCalculateRent is clicked
        try
                if txtDailyRate isnot equal to" " AND txtDaysisnot equal to " "
                        int rate is set to Integer.parseInt(txtDailyRate.getText());
                        int noOfDays is set to Integer.parseInt(txtDays.getText());
                        txtTotalAmount is set to (Integer.toString(rate*noOfDays));
                else
                        ShowMessageDialog "You cannot leave the fields Daily Rate and No Of Days
empty to calculate"
        catch(NumberFormatException nfe)
                ShowMessageDialog "Error Message\nPlease enter a valid number","Error"
else if btnConfirm is clicked
        try
                if(!txtBikeId.getText().equals("") && !txtCustomerName.getText().equals("") &&
!txtContact.getText().equals("") && !txtEmail.getText().equals("") && !txtDays.getText().equals("") &&
!txtHireDate.getText().equals(""))
```

for (int i = 0; i < BikeRentalUI.list.size(); i++)

if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))

if(BikeRentalUI.list.get(i).getBikeId()**incomparision**Integer.parseInt(txtBikeId.getText()))

bike **is set to** (BikeToRent) BikeRentalUI.list.get(i);

bikeRentStatus **is set to** bike.rentOutBike(txtCustomerName.getText(),txtContact.getText(),txtEmail.getText(),txtHireDate.getText(),Integer.parseInt(txtDays.getText()));

ShowMessageDialog"The bike has been sucessfully rented."

if bikeRentStatus **is not equal to** true

ShowMessageDialog"The bike was hired on : "+bike.getBikeHireDate()**with**" for : "+bike.getNumberOfDays()**with**" days.");

break;

else

ShowMessageDialog"You cannot leave the fields empty inorder to continue"

catch(NumberFormatException nfe)

ShowMessageDialog"Error Message\nPlease enter a valid number","Error",JOptionPane.ERROR_MESSAGE);

H) Class:SellBikeUI

Method 1:

actionPerformed(ActionEvent e)

if btnCheck **is clicked**

try

bikeFound **is set to** false;

bikeRentStatus **is set to** false;

for (int i = 0; i < BikeRentalUI.list.size(); i++)

if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))

if(BikeRentalUI.list.get(i).getBikeId()**incomparision** Integer.parseInt(txtBikeId.getText()))

```
                    bike is equal to (BikeToRent) BikeRentalUI.list.get(i);
                    if(!bike.getBikeLoanStatus())
                        txtCompanyis set to(bike.getBikeManufacturer());
                        txtDescriptionis set to (bike.getBikeDescription());
                        txtDailyRateis set toInteger.toString(bike.getDailyRate()));
                        txtTotalAmountis set toInteger.toString(bike.getTotalRentalCost()));
                        bikeRentStatus is set to true;
                    bikeFound is set to true;
                        break;
            if bikeFound AND bikeRentStatus is not equal to true
              ShowMessageDialog"The bike you searched for is not available for rent"
            if bikeFound is not equal to True
              ShowMessageDialog "The bike you searched for is not available"
            catch(NumberFormatException nfe)
              ShowMessageDialog "Error Message\nPlease enter a valid number","Error"


else if btnClear is clicked
        txtBikeId.setText("");
      txtDescription.setText("");
      txtCustomerName.setText("");
      txtContact.setText("");
      txtHireDate.setText("");
      txtDailyRate.setText("");
      txtTotalAmount.setText("");
      txtCompany.setText("");
```

```
        txtEmail.setText("");

        txtDays.setText("");

else if btnCalculateRent is clicked

        try

                if txtDailyRate isnot equal to" " AND txtDaysisnot equal to " "

                        int rate is set to Integer.parseInt(txtDailyRate.getText());

                        int noOfDays is set to Integer.parseInt(txtDays.getText());

                        txtTotalAmount is set to (Integer.toString(rate*noOfDays));

                else

                        ShowMessageDialog "You cannot leave the fields Daily Rate and No Of Days
empty to calculate"

        catch(NumberFormatException nfe)

                ShowMessageDialog "Error Message\nPlease enter a valid number","Error"

else if btnCheck is clicked

        try

                if(!txtBikeId.getText().equals("") && !txtCustomerName.getText().equals("") &&
!txtContact.getText().equals("") && !txtEmail.getText().equals("") && !txtDays.getText().equals("") &&
!txtHireDate.getText().equals""))

                for (int i = 0; i < BikeRentalUI.list.size(); i++)

            if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))

                if(BikeRentalUI.list.get(i).getBikeId()incomparisionInteger.parseInt(txtBikeId.getText()))

                        bike is set to (BikeToRent) BikeRentalUI.list.get(i);

                        bikeRentStatus is set to
bike.rentOutBike(txtCustomerName.getText(),txtContact.getText(),txtEmail.getText(),txtHireDate.getTex
t(),Integer.parseInt(txtDays.getText()));

                        ShowMessageDialog"The bike has been sucessfully rented."

                        if bikeRentStatus is not equal to true
```

```
                    ShowMessageDialog"The bike was hired on : "+bike.getBikeHireDate()with" for : "+bike.getNumberOfDays()with" days.");

                    break;

          else

            ShowMessageDialog"You cannot leave the fields empty inorder to continue"

          catch(NumberFormatException nfe)

                ShowMessageDialog"Error Message\nPlease enter a valid number","Error",JOptionPane.ERROR_MESSAGE);
```

I) Class:ReturnRentBikeUI

Method 1:

```
if btnClear is clicked

        txtBikeId.is set to("");

      else if btnConfirm is clicked

        if txtBikeId.getText()isnotequal to" "

          try

        boolean bikeFound is set to false;

        boolean bikeRentStatus is set to false;

        for (int i = 0; i < BikeRentalUI.list.size(); i++)

         if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))

            if(BikeRentalUI.list.get(i).getBikeId()incomparisionInteger.parseInt(txtBikeId.getText()))

              bike is set to (BikeToRent) BikeRentalUI.list.get(i);

              if(bike.getBikeLoanStatus())

               bike.makeBikeAvailable();

               bikeRentStatus is set to  true;

               ShowMessageDialog "The bike was on rent now its sucessfully freed for you."

              bikeFound is set to  true;
```

```
            break;
        if(bikeFound AND !bikeRentStatus is not equal to true)
          ShowMessageDialog"The bike you searched  is not currently on rent"
       else
        if!bikeFounds not equal to true
            ShowMessageDialog "The bike you searched for is not available"
       catch(NumberFormatException nfe)
          ShowMessageDialog "Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE
      else
          ShowMessageDialog"The field cannot be empty inorder to continue"
```

Method Description

A) Class: Bike

Methods	Description
public Bike(String bikeDescription,String bikeManufacturer,Integer bikeID)	It is used to pass description and manufacturer parameters.
public String getBikeDescription()	It is used to return description.
public String getBikeManufacturer()	It is used to return manufacturer.
public String getCustomerName()	It is used to return customerName.
public String getEmail()	It's used to return email.
public String getContactNumber()	It is used to return contactNumber.
public void setCustomerName(String CustomerName)	It is used to set customerName.
public void setCustomerEmail (String customerEmail)	It's used to set Email. It's used to set customer Email
public void setContactNumber(String ContactNumber)	It's used to set cotact Number
public Integer getBikeId()	It's used to get Bike Id.
public void display()	It's used to display attributes.

Table 10 Method Description Class:Bike

B) Class: BikeToRent

Methods	Descriptions
BikeToRent(String description, String manufacturer, intdailyrate, Integer bikeId)	Its used to pass parameters.
public String getBikeHireDate()	It's used to return bikeHireDate.
public String getNumberOfDays()	It is used to return numberOfDays.
public intgetDailyRate()	It is used to return dailyRate.
public intgetTotalRentalCost()	It is used to return RentalCost.
public boolean getBikeLoanStatus()	It is used to return bikeLoanStatus.
public void setBikeLoanStatus(boolean bikeLoanStatus)	It is used to set bike loanStatus
public void makeBikeAvailable()	It is used to check bike status and then make it available.
public boolean rentOutBike(String customerName,String contactNumber,String customerEmail,String ipbikeHireDate,int ipnumberOfDays)	It is used to rent the bike if it is available to be rented.

Table 11 Method Description Class:BikeToRent

C) Class: BikeToSell

Methods	Descriptions
public BikeToSell(String bikeDescription,String bikeManufacturer,int price,int taxAmount, Integer BikeId)	It is used to pass those parameters.
public int getPrice()	It is used to return price.
public int getTaxAmount()	It is used to return taxAmount.
public int getTotalAmount()	It is used to return totalAmount.
public String getSellingDate()	It is used to return sellingDate.
public boolean getSellingStatus()	It is used to return sellingStatus.
public boolean bikeSellOut(String customerName,String contactNumber, String customerEmail,String ipsellingDate)	It is used to sell the bike if it is available.

Table 12 Method Description Class:BikeToSell

D) Class: BikeRentalUI

Methods	Descriptions
public BikeRentalUI()	It is the constructor in which instance variable initialises
public void makeFrame()	It is used to create a frame.
public void display()	It is used to display description and manufacturer of the bike
public void actionPerformed(ActionEvent e)	It is used to create action events when the respective buttons are clicked.
public static void main(String args[])	It is the main method of the class
public static void addBikeToRent(String bikeDescription, String bikeManufacturer, int dailyRate, Integer bikeID)	
public static void addBikeToSell(String bikeDescription,String bikeManufacturer,int price,int taxAmount,Integer bikeID)	

Table 13 Method Description Class:BikeRentalUI

D) Class: AddBikeToRentUI

Table 14 Method Description AddBikeToRentUI

Methods	Descriptions
public AddBikeToRentUI()	It is the constructor in which instance variable initialises
public void BikeToRentWindow()	It is used to create a frame.
public void actionPerformed(ActionEvent e)	It is used to create action events when the respective buttons are clicked. It is the main method of the class

E) Class: AddBikeToSellUI

Methods	Descriptions
public AddBikeToSellUI()	It is the constructor in which instance variable initialises
public void BikeToSellWindow()	It is used to create a frame.
public void actionPerformed(ActionEvent e)	It is used to create action events when the respective buttons are clicked. It is the main method of the class

Table 15 Method Description AddBikeToSellUI

F) Class: RentBikeUI

Methods	Descriptions
public RentBikeUI()	It is the constructor in which instance variable initialises
public void RentBikeWindow()	It is used to create a frame.
public void actionPerformed(ActionEvent e)	It is used to create action events when the respective buttons are clicked. It is the main method of the class

Table 16 Method Description RentBikeUI

G) Class: SellBikeUI

Methods	Descriptions
public SellBikeUI()	It is the constructor in which instance variable initialises
public void SellBikeWindow()	It is used to create a frame.
public void actionPerformed(ActionEvent e)	It is used to create action events when the respective buttons are clicked. It is the main method of the class

Table 17 Method Description SellBikeUI

H) Class:ReturnRentBikeUI

Methods	Descriptions
public ReturnRentBikeUI ()	It is the constructor in which instance variable initialises
public void ReturnRentBikeWindow()	It is used to create a frame.
public void actionPerformed(ActionEvent e)	It is used to create action events when the respective buttons are clicked. It is the main method of the class

Table 18 Method Description ReturnRentBIkeUI

Testing

Test 1:

Test that the program can be compiled and run using the command prompt,

Test No.	1
Action	Testing that program can be compiled and run using the command prompt
Expected Result	All the files to be compiled and run correctly
Actual Result	All the data were compiled and run successfully
Conclusion	The application is independant

Table 19 Test that the program can be compiled and run using the command prompt

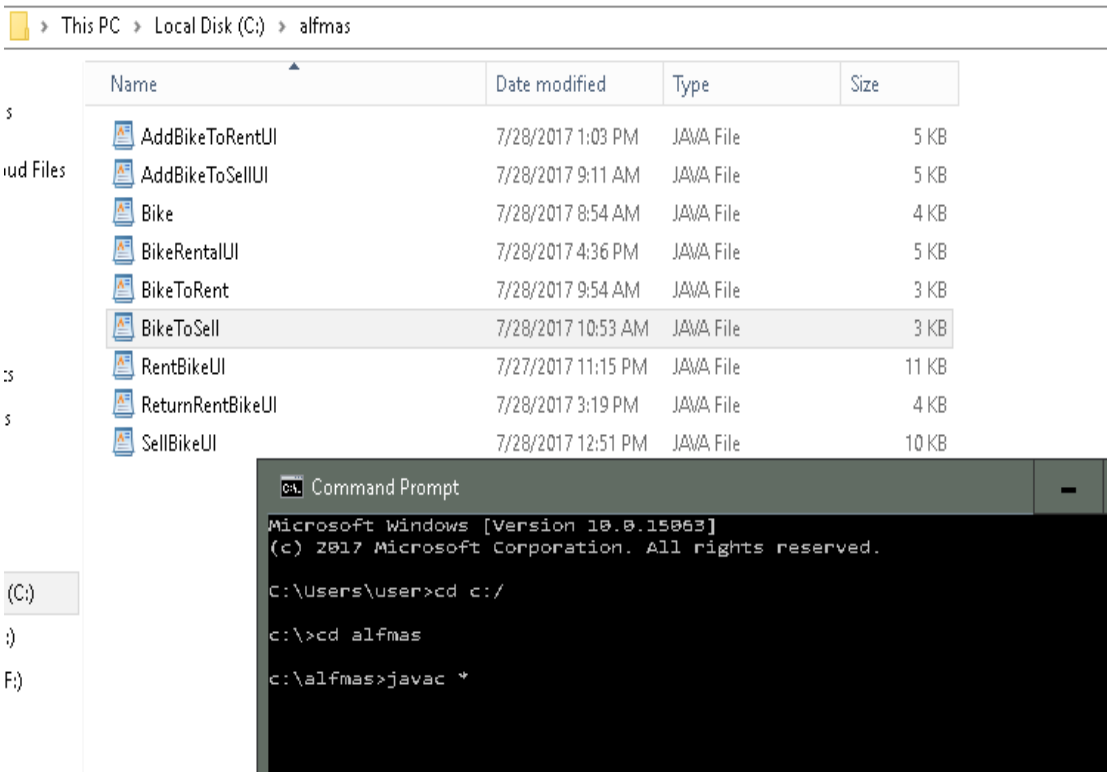

Figure 2 Test through CommandPrompt 1.1

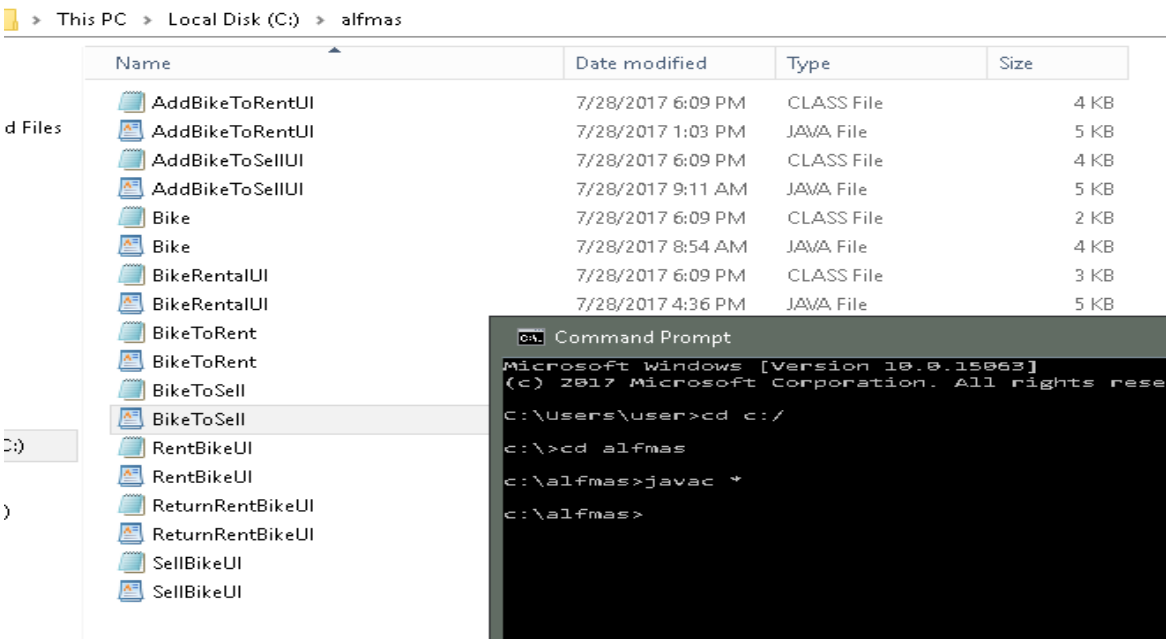

Figure 3 Test through CommandPrompt 1.2

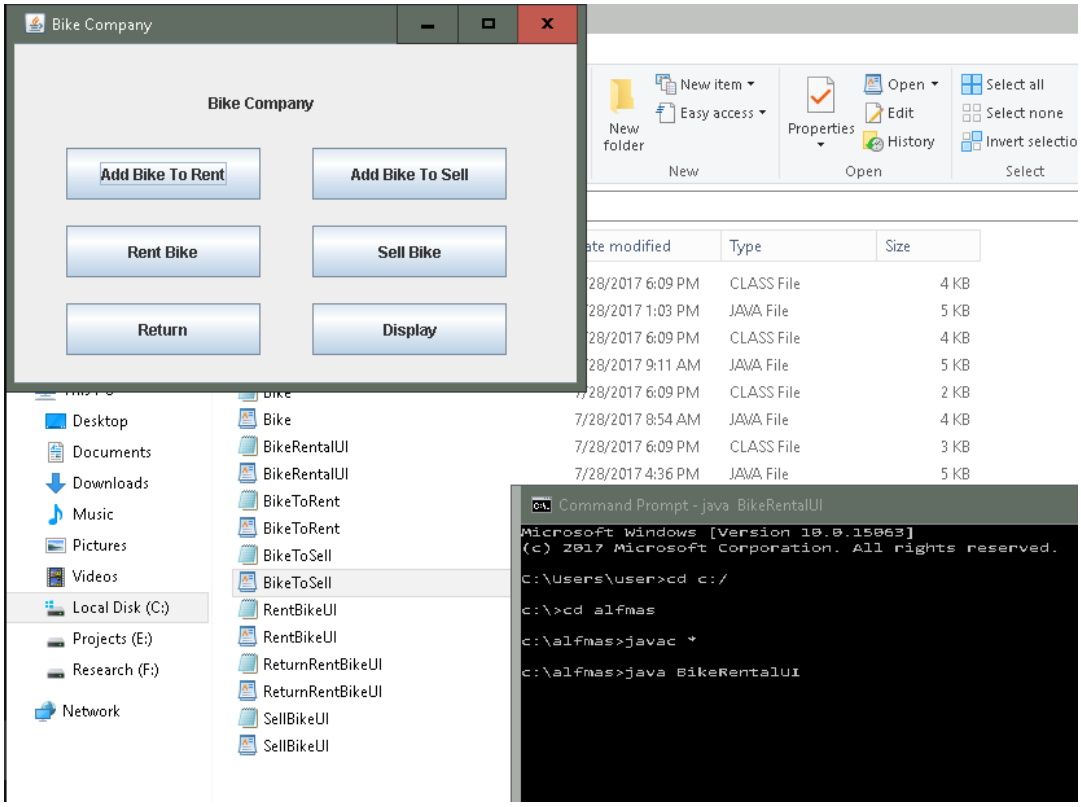

Figure 4 Test through CommandPrompt 1.3

Test 2:

Test 2.1:

Adding A bike to Register

Test No.	2.1
Action	Adding a bike to register
Expected Result	Bike should be registered normaly after the entries
Actual Result	Details of bike are added
Conclusion	Details of bike to sell has been displayed.

Table 20 Adding a bike to Register

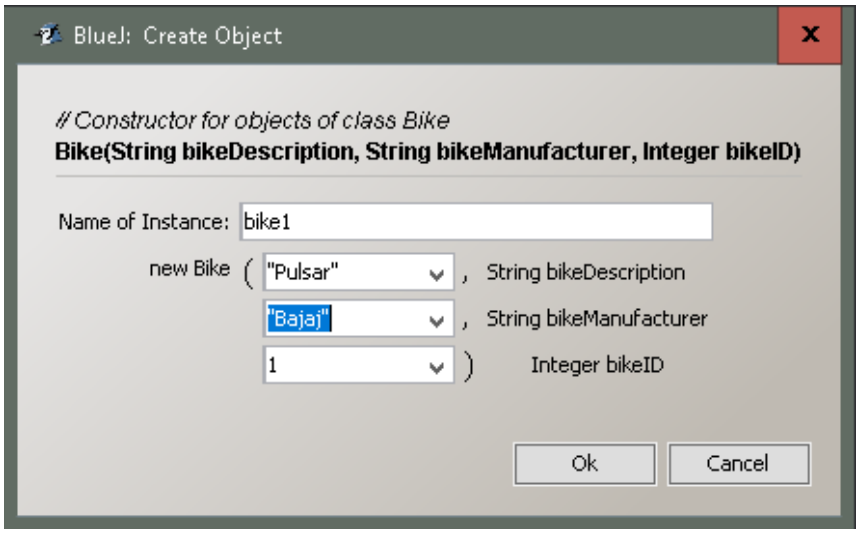

Figure 5 Adding A bike to Register 2.1

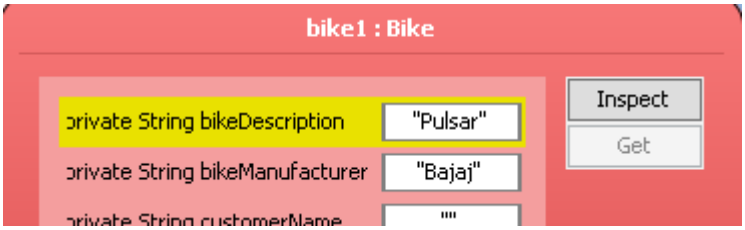

Figure 6 Results of Adding A bike to Register 2.1

Test 2.2:

Adding a bike to rent

Test No.	2
Action	Adding a bike to rent
Expected Result	Details of biketo rent to be entered successfully
Actual Result	All the details of biketo rent were entered successfully
Conclusion	Details of Add bike to rent has been displayed.

Table 21 Adding a bike to rent

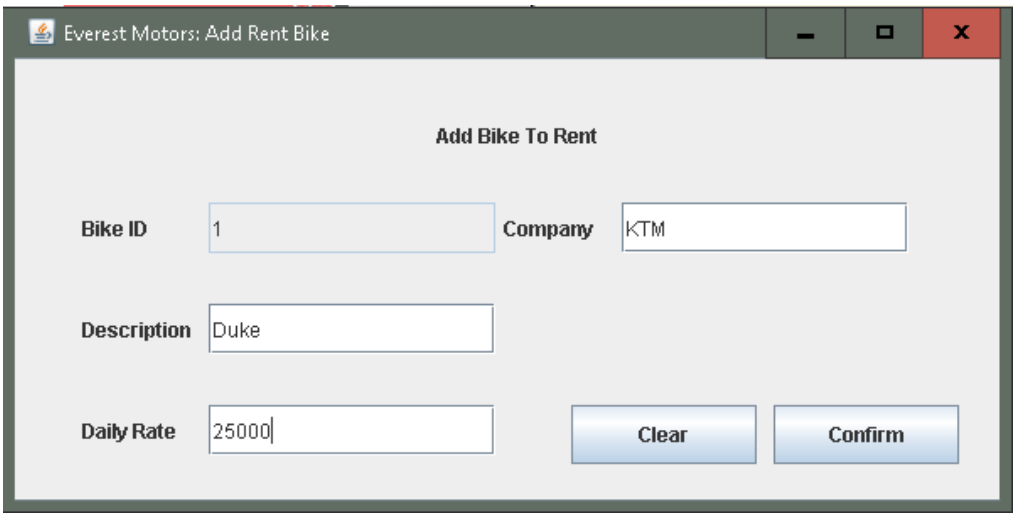

Figure 7 Adding a bike to rent 2.2

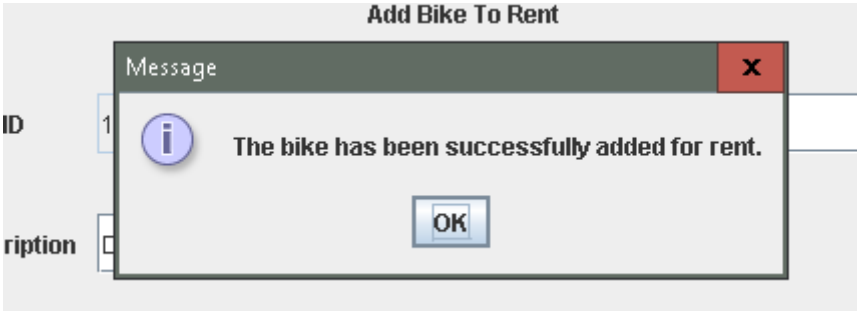

Figure 8 Results Adding a bike to rent

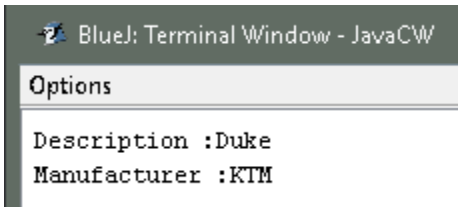

Figure 9 Displaying data of Adding a bike to rent

Test2.3:

Sell A bike

Table 22 Sell A bike

Test No.	2.3
Action	First a bike has been added then sold
Expected Result	The bike should be sold succesfully
Actual Result	The bike was sold successfuly
Conclusion	Selling bike part of the appication works

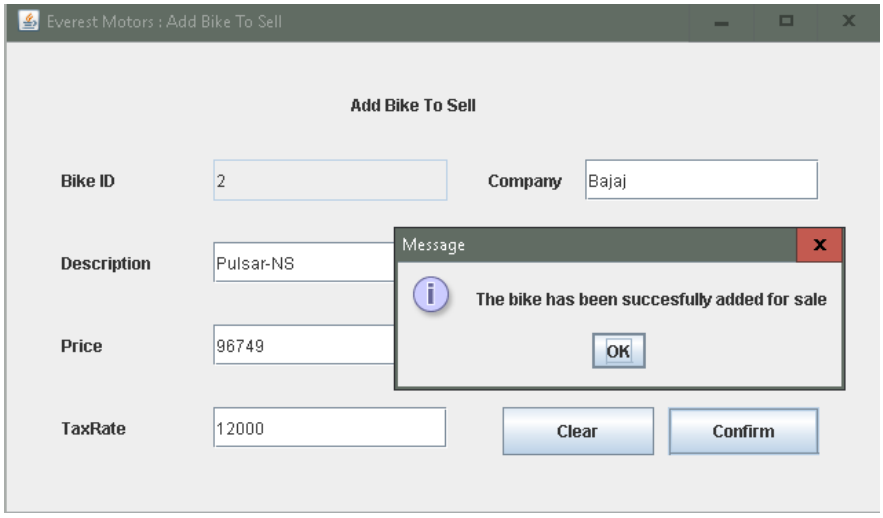

Figure 10 Adding a bike to sell

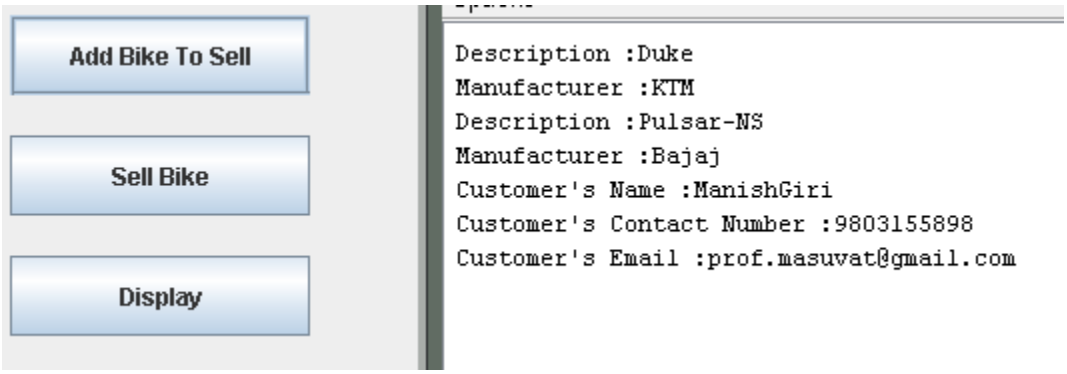

Figure 11 Results after the bike was sold

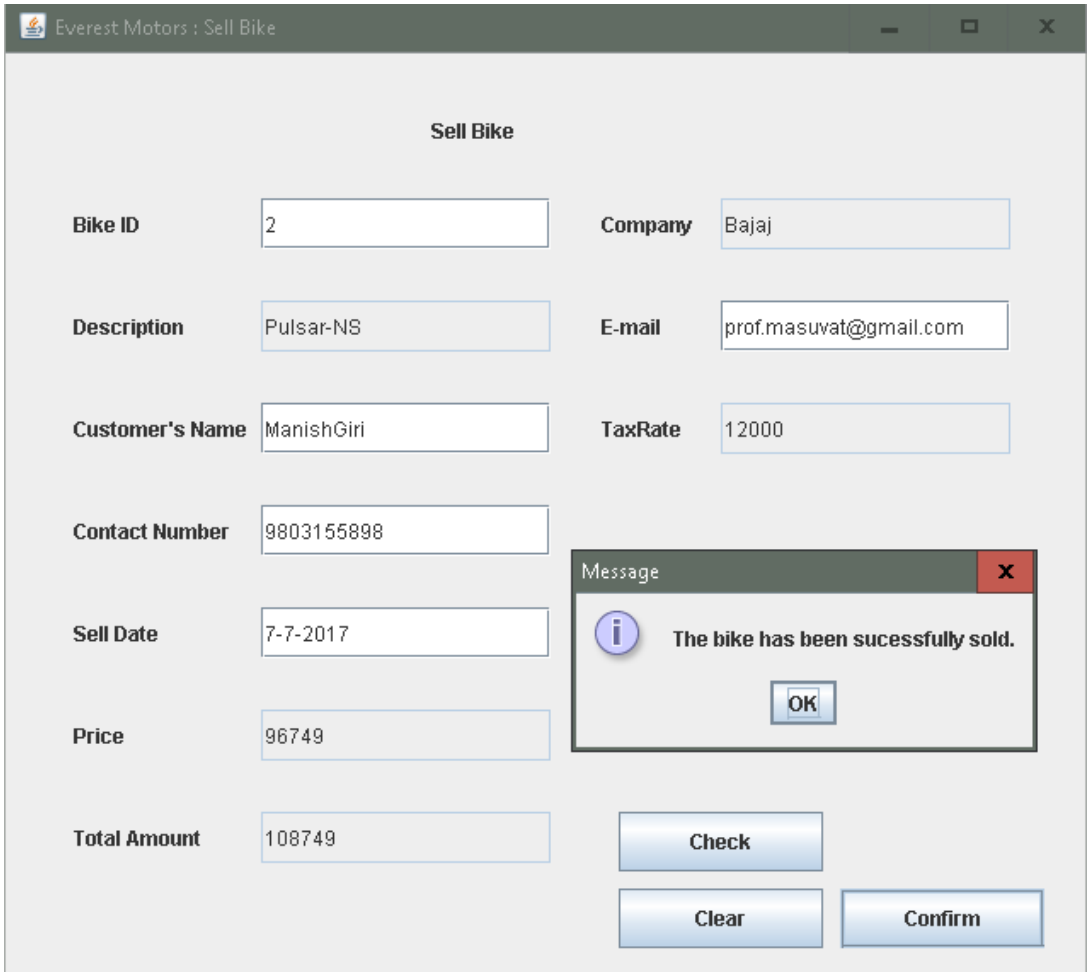

Figure 12 Selling a bike

Test 2.4:

Rent a bike

Table 23: Rent A bike

Test No.	2.4
Action	Renting a bike
Expected result	Bike should be rented successfully
Actual result	The bike was rented succesfully
Conclusion	Renting bike part of application works.

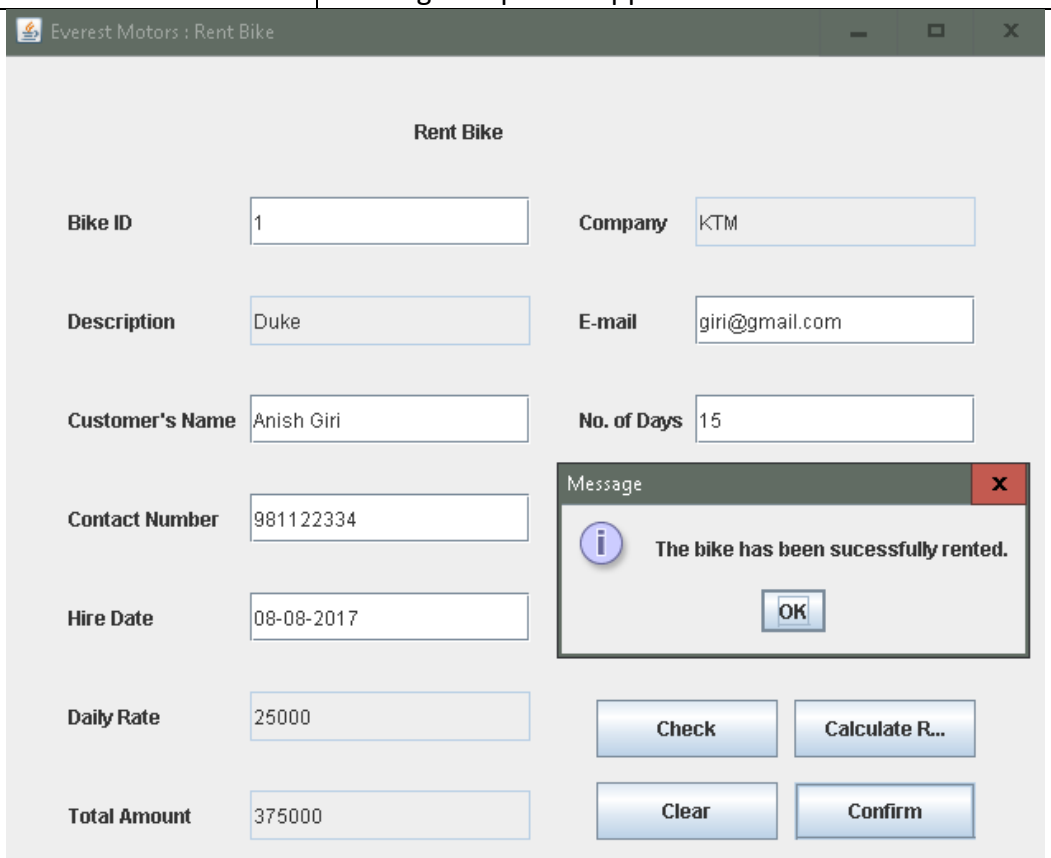

Figure 13 Renting a bike

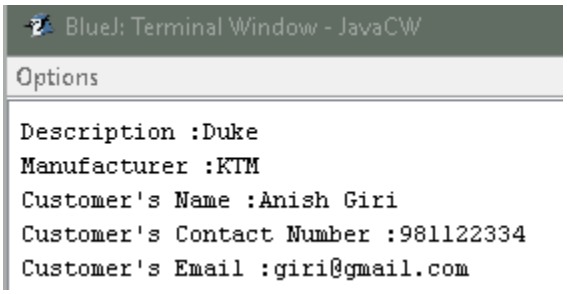

Figure 14 Display Method after Renting a bike

Test 2.5:

returning a bike

Table 24 returning a bike.

Test No.	2.5
Action	Returning a bike
Expected result	The bike should be returned succesfully.
Actual result	The bike returned successfully
Conclusion	Bike has returned and now we can rent the bikeagain.

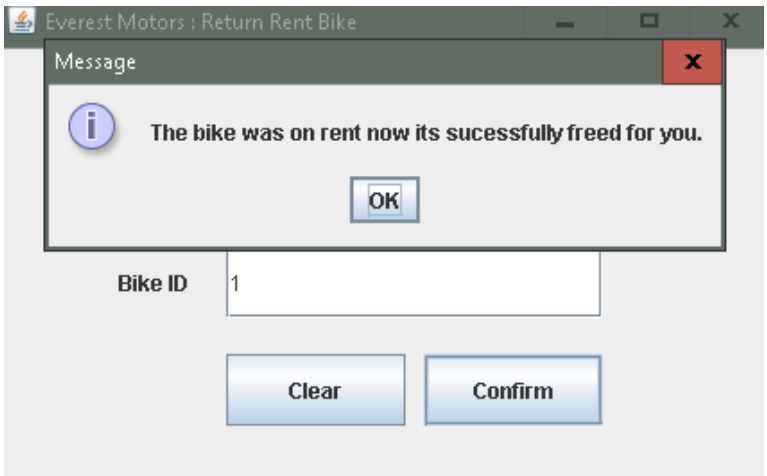

Figure 15 Returning a bike

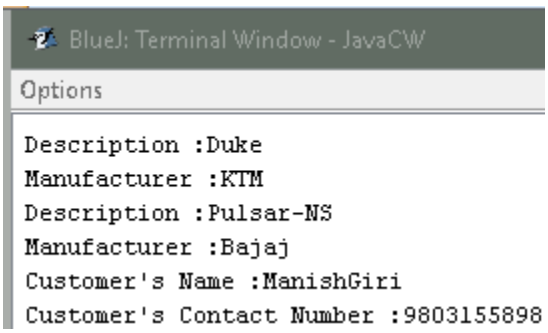

Figure 16 Display method after returning a bike

Test 3:

Test that appropriate dialog boxes appear when unsuitable values are entered for the car number,

Test No.	3
Action	Test that appropriate dialog boxes appear when unsuitable values are entered for the car number,
Expected result	Dialog boxes should popup
Actual result	The dialog boxes popped up
Conclusion	Bike has returned and now we can rent the bike.

Table 25 Test for dialog boxes

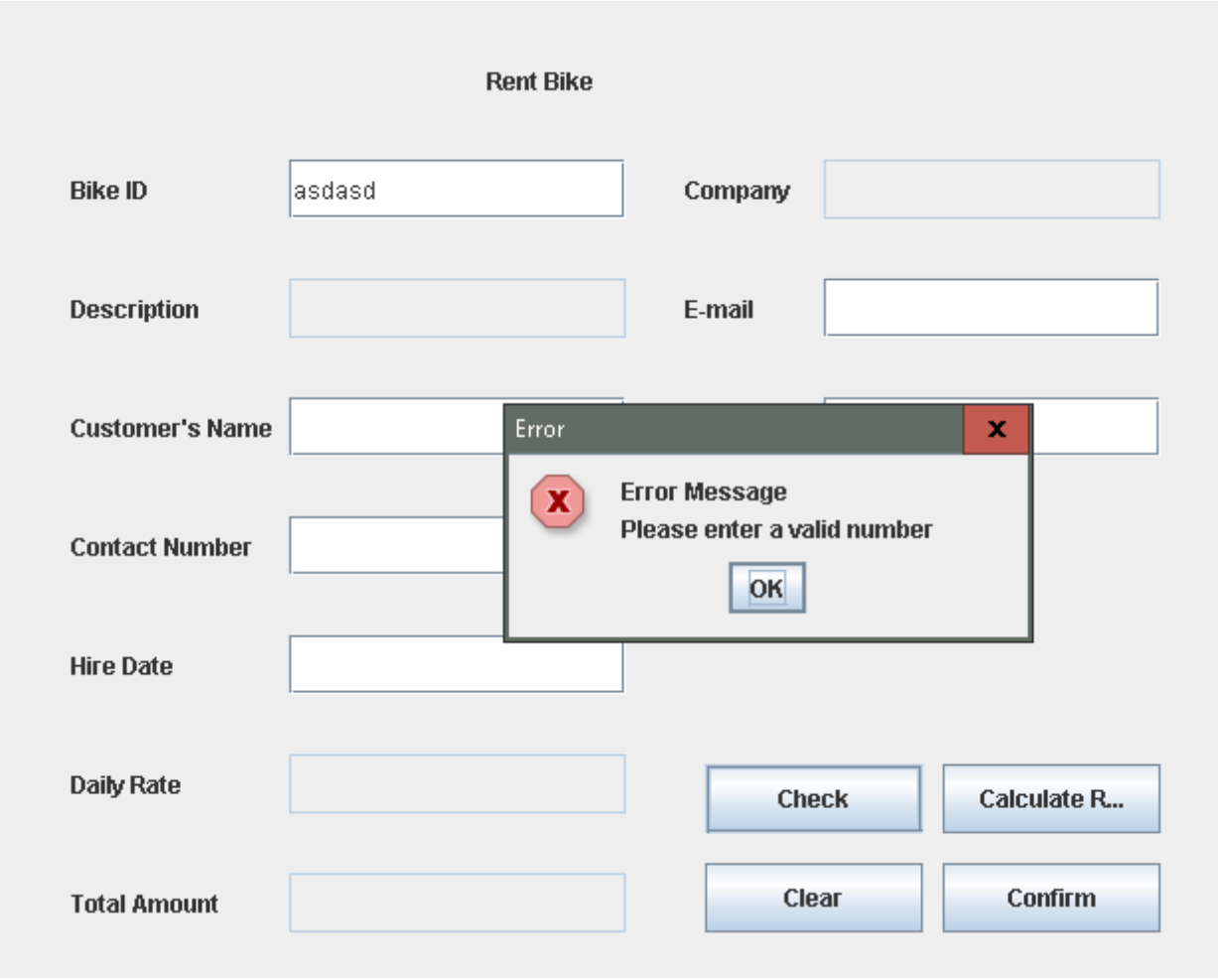

Figure 17 Test For dialouge boxes

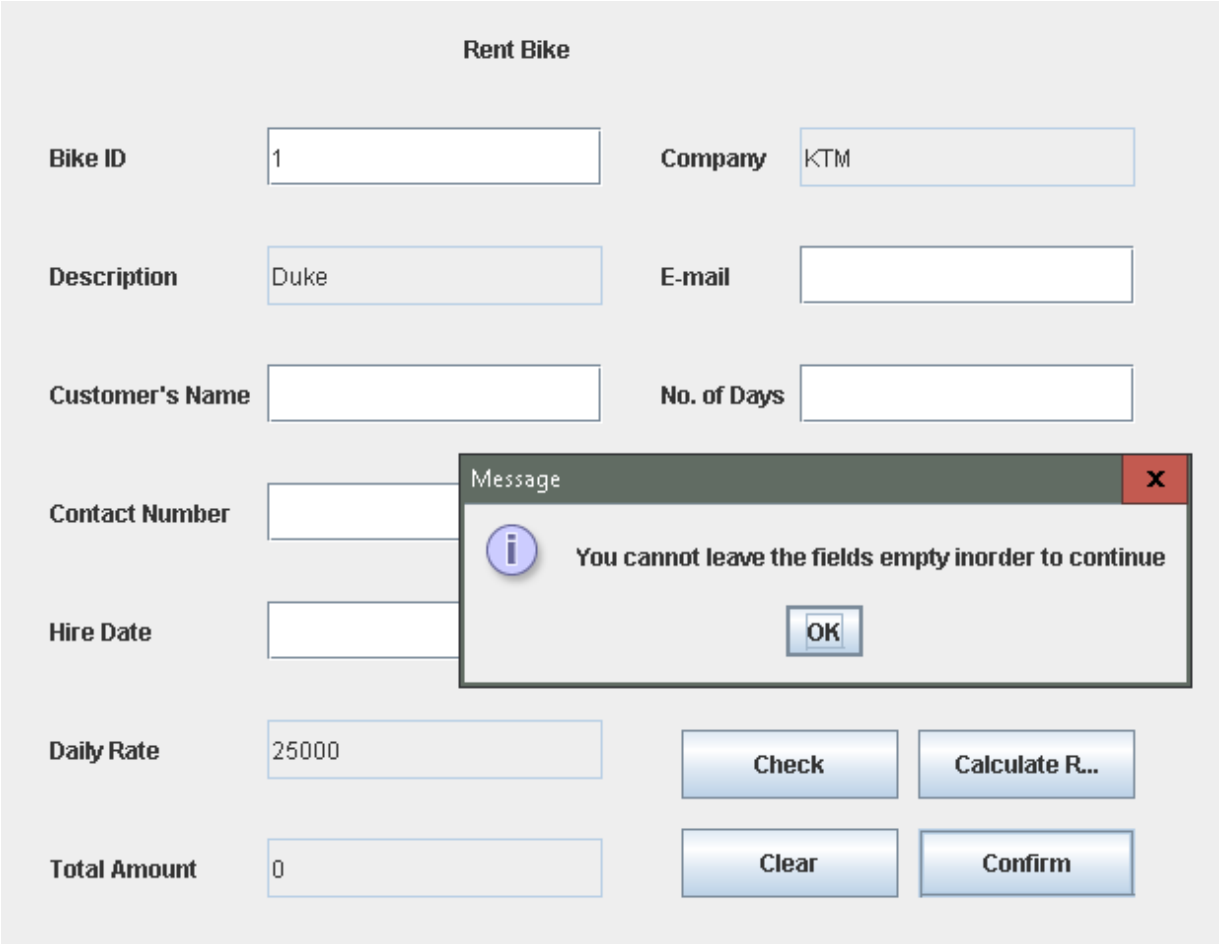

Figure 18 Empty field dialog box

Error Detection

Error 1

```
{
    new BikeRentalUI();
}

/**
 * actionPerformed - This
```

```
java.lang.StackOverflowError:
null
```

Figure 19 Stack OverFlow error

Error 2

```
public static void addBikeToRent(String bikeDescript
    list.add(new BikeToRent(bikeDescription,bikeManu
    this.bikeDescription=Description;
```

```
non-static variable this cannot be referenced from a static context
```

Figure 20 non static variable Error

Error 3

```
        if(e.getSource()=btnDisplay)
        {
            display();
        }
    }
```

```
unexpected type
 required: variable;   found:   value
```

Figure 21 Type Error

Conclusion

We want to thank our module leader for doling out this kind of coursework for building up our programming abilities, to be imaginative and be familiar with coding. We have utilized a "BlueJ" java programming for coding the program. In this course work we have made a class "Bicycle" and two sub classes "BikeToRent" and "BikeToSell" which is associated with another class "CustomerDetails".

This coursework helped me to build up my aptitudes and innovativeness. After the finishing of the coursework we are such a great amount of well-known to the java programming dialect. A considerable lot of the issue are face amid the coding, strategy review and some more. Every one of the issues are manage the assistance of our module pioneer, books and on the web.

Appendix

BIKE CLASS

```
/**
 * Bike class mainly deals with the bike description, manufacturer,
 * customer's name ,number and their email addresses.
 *
 * @author (Manish Giri)
 * @version (Sun,23 JULY)
 */
public class Bike
{
   // instance variables - replace the example below with your own
   private String bikeDescription;
   private String bikeManufacturer;
   private String customerName;
   private String contactNumber;
```

```
private String customerEmail;
private Integer bikeID;

/**
 * Constructor for objects of class Bike
 */
public Bike(String bikeDescription,String bikeManufacturer,Integer bikeID)
{
   // initialise instance variables
   this.bikeDescription = bikeDescription ;
   this.bikeManufacturer = bikeManufacturer ;
   this.bikeID = bikeID;
   customerName ="";
   contactNumber ="";
   customerEmail ="";
}

/**
 * setCustomerName - This method sets the customer name.
 *
 * @param  customerName   Name of the customer
 * @return
 */
public void setCustomerName(String CustomerName)
{
```

```
  this.customerName = CustomerName;
}

 /**
 * setContactNumber - This method sets the contact number of customers.
 *
 * @param  contactNumber   Customer's Contact Number
 */
public  void setContactNumber(String ContactNumber)
{
  this.contactNumber = ContactNumber;
}

 /**
 * setCustomerEmail - This method sets the email of customers.
 *
 * @param  customerEmail   Customer's Email
 */
public void setCustomerEmail(String customerEmail)
{
  this.customerEmail = customerEmail;
}

/**
 * getBikeDescription - This method provides the description of bikes.
```

```
 *
 * @return  bikeDescription   String
 */
public String getBikeDescription()
{
  return bikeDescription ;
}

/**
 * getBikeManufacturer - This method provides the name of bike manufacturers.
 *
 * @return  bikeManufacturer   String
 */
public String getBikeManufacturer()
{
  return bikeManufacturer ;
}

/**
 * getCustomerName - This method provides the name of customers.
 *
 * @return  customerName   String
 */
public String getCustomerName()
{
```

```
  return customerName ;
}

 /**
 * getBikeId - This method provides theBikeID.
 *
 * @return  getBikeId   int
 */
public Integer getBikeId()
{
  return bikeID;
}


/**
 * getContactNumber - This method provides the contact numbers of customers.
 *
 * @return  contactNumber   String
 */
public String getContactNumber()
{
  return contactNumber ;
}

/**
```

```
 * getCustomerEmail - This method provides the email's of customers.
 *
 * @return  customerEmail   String
 */
public String getCustomerEmail()
{
  return customerEmail ;
}


/**
 * display - This method displays the details of the customers and the bikes.
 */
public void display()
{
  String nullString = null ;
  String empty = new String();
  System.out.println("Description :"+ bikeDescription);
  System.out.println("Manufacturer :"+ bikeManufacturer);



  if(!customerName.equals(""))
  {
     System.out.println("Customer's Name :"+ customerName);
    }
```

```
    if(!contactNumber.equals(""))
    {
        System.out.println("Customer's Contact Number :"+ contactNumber);
      }

    if (!customerEmail.equals(""))
    {
        System.out.println("Customer's Email :"+ customerEmail);
      }

  }

}
```

BIKETORENT CLASS

```
/**
 * BikeToRent class provides the mechanism for the process to rent a bike
 *
 * @author (Manish Giri)
 * @version (07282017)
 */
public class BikeToRent extends Bike
{
   // instance variables
   private  String bikeHireDate;
   private  int numberOfDays;
```

```
private  int dailyRate;

private  int totalRentalCost;

private  boolean bikeLoanStatus;


/**

 * Constructor for objects of class BikeToRent

 */

public BikeToRent(String bikeDescription,String bikeManufacturer,int dailyRate, Integer bikeId)

{

   // initialise instance variables

   super(bikeDescription,bikeManufacturer,bikeId) ;

   this.dailyRate = dailyRate ;

   this.totalRentalCost = 0 ;

   this.numberOfDays = 0 ;

   this.bikeLoanStatus = false ;

   this.bikeHireDate = "";

}


public String getBikeHireDate()

{

   return bikeHireDate ;

}


public int getNumberOfDays()

{
```

```
        return numberOfDays ;
    }

    public int getDailyRate()
    {
        return dailyRate ;
    }

    public int getTotalRentalCost()
    {
        return totalRentalCost ;
    }

    public boolean getBikeLoanStatus()
    {
        return bikeLoanStatus ;
    }

    public void setBikeLoanStatus(boolean bikeLoanStatus)
    {
        this.bikeLoanStatus = bikeLoanStatus ;
    }

    public boolean rentOutBike(String customerName,String contactNumber,String customerEmail,String
ipbikeHireDate,int ipnumberOfDays)
    {
```

```
    if(!bikeLoanStatus)
    {
        this.setCustomerName(customerName) ;
        this.setContactNumber(contactNumber) ;
        this.setCustomerEmail(customerEmail) ;

        this.bikeHireDate = ipbikeHireDate ;
        this.numberOfDays = ipnumberOfDays ;

        this.bikeLoanStatus = true ;
        this.totalRentalCost = dailyRate * numberOfDays ;
        return true;

    }
    else
    {
        System.out.println("Sorry!The bike is already on Rent.");
        return false;
    }
  }

  public void makeBikeAvailable()
  {
    if (!bikeLoanStatus){
        System.out.println("Oh! The bike is already available.");
```

```
        }
        else{
            setCustomerName("") ;
            setContactNumber("") ;
            setCustomerEmail("") ;

            this.numberOfDays = 0 ;
            this.bikeHireDate = "" ;
            bikeLoanStatus = false ;
        }
    }
}
```

BIKETOSELL CLASS

```
/**
 * Bike to sell class provides the mechanism for selling a bike
 *
 * @author (Manish Giri)
 * @version (7272017)
 */
public class BikeToSell extends Bike
{
    // instance variables - replace the example below with your own
    private int price;
    private int taxAmount;
    private int totalAmount;
```

```
    private String sellingDate;

    private boolean sellingStatus;


    /**

     * Constructor for objects of class BikeToSell

     */

    public BikeToSell(String bikeDescription,String bikeManufacturer,int price,int taxAmount, Integer
BikeId)

    {

            // initialise instance variables

        super(bikeDescription,bikeManufacturer,BikeId);

        this.price = price;

        this.taxAmount = taxAmount;

        sellingStatus= false;

        sellingDate ="";

    }


    /**

     * getPrice - This method return the value of price

     *

     * @return   int  price

     */

    public int getPrice()

    {

        return price;

    }
```

```
/**
 * getTaxAmount - This method return the value of taxAmount
 *
 * @return   int  taxAmount
 */
public int getTaxAmount()
{
   return taxAmount;
}

/**
 * getTotaAmount - This method return the value of totalAmount
 *
 * @return   int  totalAmount
 */
public int getTotalAmount()
{
   return totalAmount;
}

/**
 * getSellingDate - This method return the value of sellingDate
 *
 * @return   String  sellingDate
```

```
 */
public String getSellingDate()
{
    return sellingDate;
}


/**
 * getSellingStatus - This method return the value of sellingDate
 *
 * @return   boolean  sellingStatus
 */
public boolean getSellingStatus()
{
    return sellingStatus;
}


public boolean bikeSellOut(String customerName,String contactNumber, String customerEmail,String
ipsellingDate)
{
    if(!sellingStatus)
    {
        this.setCustomerName(customerName) ;
        this.setContactNumber(contactNumber) ;
        this.setCustomerEmail(customerEmail) ;


        this.totalAmount = price+taxAmount ;
```

```
        this.sellingStatus = true ;
        this.sellingDate = ipsellingDate;
        return true;
    }
    else
    {
        System.out.println("Sorry! The bike is already sold.");
        System.out.println("The bike was sold on: "+sellingDate);
        return false;
    }
  }
}
```

BIKERENTALUI CLASS

```
import javax.swing.JLabel;
import javax.swing.JFrame;
import javax.swing.JButton;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.util.ArrayList ;
/**
 * BIKEUI class contains the graphical user interface of the main
 * bike company.
 *
 * @author (Manish Giri)
```

```
 * @version (Sun,23 JULY)
 */
public class BikeRentalUI implements ActionListener
{
   // instance variables
   private JFrame frame;
   private JLabel lblBikeCompany;
   private JButton btnAddBikeToRent;
   private JButton btnAddBikeToSell;
   private JButton btnRentBike;
   private JButton btnSellBike;
   private JButton btnReturn;
   private JButton btnDisplay;
   public static ArrayList<Bike> list = new ArrayList<Bike>();
   /**
    *
    * Constructor for objects of class BikeUI
    */
   public BikeRentalUI()
   {
      // initialise instance variables
      makeFrame();
   }

   /**
```

```
 * makeFrame - This method creates the required labels and buttons
 *           for this class.
 */
public void makeFrame()
{
    frame = new JFrame("Bike Company");

    lblBikeCompany = new JLabel("Bike Company");
    frame.add(lblBikeCompany);
    lblBikeCompany.setBounds(150,30,200,30);

    btnAddBikeToRent= new JButton("Add Bike To Rent");
    frame.add(btnAddBikeToRent);
    btnAddBikeToRent.setBounds(40,80,150,40);
    btnAddBikeToRent.addActionListener(this);

    btnAddBikeToSell= new JButton("Add Bike To Sell");
    frame.add(btnAddBikeToSell);
    btnAddBikeToSell.setBounds(230,80,150,40);
    btnAddBikeToSell.addActionListener(this);

    btnRentBike= new JButton("Rent Bike");
    frame.add(btnRentBike);
    btnRentBike.setBounds(40,140,150,40);
    btnRentBike.addActionListener(this);
```

```
    btnSellBike= new JButton("Sell Bike");
    frame.add(btnSellBike);
    btnSellBike.setBounds(230,140,150,40);
    btnSellBike.addActionListener(this);

    btnReturn= new JButton("Return");
    frame.add(btnReturn);
    btnReturn.setBounds(40,200,150,40);
    btnReturn.addActionListener(this);

    btnDisplay= new JButton("Display");
    frame.add(btnDisplay);
    btnDisplay.setBounds(230,200,150,40);
    btnDisplay.addActionListener(this);

    frame.setLayout(null);
    frame.setSize(450,300);
    frame.setVisible(true);
  }

  /**
   * main - This is the main method of the class.
   */
  public static void main(String args[])
```

```
{
    new BikeRentalUI();
}

/**
 * actionPerformed - This method links all the classes with the buttons
 *                   created on the Graphical User Interface.
 */
public void actionPerformed(ActionEvent e)
{
    if (e.getSource()==btnAddBikeToRent)
    {
        new BikeToRentUI();
    }

    if(e.getSource()==btnRentBike)
    {
        new RentBikeUI();
    }

    if (e.getSource()==btnSellBike)
    {
        new SellBikeUI();
    }
```

```
        if(e.getSource()==btnAddBikeToSell)
        {
            new BikeToSellUI();
        }

        if(e.getSource()==btnReturn)
        {
            new ReturnRentBikeUI();
        }

        if(e.getSource()==btnDisplay)
        {
            display();
        }
    }

    public static void addBikeToRent(String bikeDescription, String bikeManufacturer, int dailyRate,
Integer bikeID) {
        list.add(new BikeToRent(bikeDescription,bikeManufacturer,dailyRate,bikeID));

    }

    public static void addBikeToSell(String bikeDescription,String bikeManufacturer,int price,int
taxAmount,Integer bikeID) {
        list.add(new BikeToSell(bikeDescription,bikeManufacturer,price,taxAmount,bikeID));
```

```
    }

    private void display()
    {

        for (int i = 0; i < list.size(); i++)
        {
            list.get(i).display();
        }

    }

}
```

BIKETORENTUI CLASS

```
import javax.swing.JLabel;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.*;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.util.* ;
```

```
/**
 * BikeToRentUI class contains the graphical user interface of rental
 * page of the bike company
 *
 *
 * @author (Manish Giri)
 * @version (Sun,23 JULY)
 */
public class BikeToRentUI implements ActionListener
{
   // instance variables
   private JFrame frame;

   private JLabel lblAddBikeToRent;
   private JLabel lblBikeId;
   private JLabel lblDescription;
   private JLabel lblDailyRate;
   private JLabel lblCompany;

   private JTextField txtBikeId;
   private JTextField txtDescription;
   private JTextField txtDailyRate;
   private JTextField txtCompany;

   private JButton btnClear;
```

```
private JButton btnConfirm;

private BikeToRent bikeToRent;
private BikeRentalUI bikeUI;
/**
 * Constructor for objects of class AddBikeToRentUI
 */
public BikeToRentUI()
{
    // initialise instance variables
    BikeToRentWindow();
}

/**
 * BikeToRentWindow - This method creates the required labels and buttons
 *                    for this class.
 */
public void BikeToRentWindow()
{
    frame = new JFrame("Everest Motors: Add Rent Bike");

    lblAddBikeToRent = new JLabel("Add Bike To Rent");
    frame.add(lblAddBikeToRent);
    lblAddBikeToRent.setBounds(250,30,200,30);
```

```
lblBikeId = new JLabel("Bike ID");
frame.add(lblBikeId);
lblBikeId.setBounds(40,80,150,40);

lblDescription = new JLabel("Description");
frame.add(lblDescription);
lblDescription.setBounds(40,140,150,40);

lblDailyRate = new JLabel("Daily Rate");
frame.add(lblDailyRate);
lblDailyRate.setBounds(40,200,150,40);

lblCompany = new JLabel("Company");
frame.add(lblCompany);
lblCompany.setBounds(290,80,150,40);

txtBikeId = new JTextField("");
frame.add(txtBikeId);
txtBikeId.setBounds(115,85,170,30);
txtBikeId.setEditable(false);
int a = BikeRentalUI.list.size()+1;
txtBikeId.setText(Integer.toString(a));


txtDescription = new JTextField("");
```

```
    frame.add(txtDescription);
    txtDescription.setBounds(115,145,170,30);

    txtDailyRate = new JTextField("");
    frame.add(txtDailyRate);
    txtDailyRate.setBounds(115,205,170,30);

    txtCompany = new JTextField("");
    frame.add(txtCompany);
    txtCompany.setBounds(360,85,170,30);

    btnClear = new JButton("Clear");
    frame.add(btnClear);
    btnClear.setBounds(330,205,110,35);
    btnClear.addActionListener(this);

    btnConfirm = new JButton("Confirm");
    frame.add(btnConfirm);
    btnConfirm.setBounds(450,205,110,35);
    btnConfirm.addActionListener(this);

    frame.setLayout(null);
    frame.setSize(600,300);
    frame.setVisible(true);
  }
```

```
    /**
    * actionPerformed - This method links all the classes with the buttons
    *                 created on the Graphical User Interface.
    */
   public void actionPerformed(ActionEvent e)
   {
      if (e.getSource()==btnConfirm)
      {
        try
        {
          if(!txtDescription.getText().equals("") && !txtCompany.getText().equals("") &&
!txtDailyRate.getText().equals(""))
          {

BikeRentalUI.addBikeToRent(txtDescription.getText(),txtCompany.getText(),Integer.parseInt(txtDailyRat
e.getText()),Integer.parseInt(txtBikeId.getText()));
            JOptionPane.showMessageDialog(frame,"The bike has been successfully added for rent.");
          }
          else
          {
            JOptionPane.showMessageDialog(frame,"You cannot leave the fields empty inorder to
continue");
          }
        }
        catch(NumberFormatException nfe)
        {
```

```
            JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE);

        }

      }

      else if(e.getSource()==btnClear)

      {

         txtDescription.setText("");

         txtDailyRate.setText("");

         txtCompany.setText("");

      }

   }

}
```

BIKETOSELLUI CLASS

```
import javax.swing.JLabel;

import javax.swing.JFrame;

import javax.swing.JButton;

import javax.swing.JTextField;

import javax.swing.*;


import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/**

 * BikeToSellUI class contains the graphical user interface of rental

 * page of the bike company

 *
```

```
 *
 * @author (Manish Giri)
 * @version (Sun,24 JULY)
 */
public class BikeToSellUI implements ActionListener
{
   // instance variables
   private JFrame frame;
   private JLabel lblAddBikeToSell;
   private JLabel lblBikeId;
   private JLabel lblDescription;
   private JLabel lblPrice;
   private JLabel lblTaxRate;
   private JLabel lblCompany;

   private JButton btnClear;
   private JButton btnConfirm;

   private JTextField txtBikeId;
   private JTextField txtDescription;
   private JTextField txtPrice;
   private JTextField txtTaxRate;
   private JTextField txtCompany;
```

```
/**
 * Constructor for objects of class AddBikeToSellUI
 */
public BikeToSellUI()
{
    // initialise instance variables
   AddBikeToSellWindow();
}


/**
 * RentBikeWindow - This method creates the required labels and buttons
 *                 sfor this class.
 */
public void AddBikeToSellWindow()
{
    // put your code here
    frame = new JFrame("Everest Motors : Add Bike To Sell");


    lblAddBikeToSell = new JLabel("Add Bike To Sell");
    frame.add(lblAddBikeToSell);
    lblAddBikeToSell.setBounds(250,30,200,30);


    lblBikeId = new JLabel("Bike ID");
    frame.add(lblBikeId);
    lblBikeId.setBounds(40,80,150,40);
```

```
lblDescription = new JLabel("Description");
frame.add(lblDescription);
lblDescription.setBounds(40,140,150,40);

lblPrice = new JLabel("Price");
frame.add(lblPrice);
lblPrice.setBounds(40,200,150,40);

lblTaxRate = new JLabel("TaxRate");
frame.add(lblTaxRate);
lblTaxRate.setBounds(40,260,150,40);

lblCompany = new JLabel("Company");
frame.add(lblCompany);
lblCompany.setBounds(350,80,150,40);

txtBikeId = new JTextField("");
frame.add(txtBikeId);
txtBikeId.setBounds(150,85,170,30);
txtBikeId.setEditable(false);
int a = BikeRentalUI.list.size()+1;
txtBikeId.setText(Integer.toString(a));

txtDescription = new JTextField("");
```

```
frame.add(txtDescription);
txtDescription.setBounds(150,145,170,30);

txtPrice= new JTextField("");
frame.add(txtPrice);
txtPrice.setBounds(150,205,170,30);

txtTaxRate= new JTextField("");
frame.add(txtTaxRate);
txtTaxRate.setBounds(150,265,170,30);

txtCompany = new JTextField("");
frame.add(txtCompany);
txtCompany.setBounds(420,85,170,30);

btnClear = new JButton("Clear");
frame.add(btnClear);
btnClear.setBounds(360,265,110,35);
btnClear.addActionListener(this);

btnConfirm = new JButton("Confirm");
frame.add(btnConfirm);
btnConfirm.setBounds(480,265,110,35);
btnConfirm.addActionListener(this);
```

```
    frame.setLayout(null);
    frame.setSize(650,380);
    frame.setVisible(true);
  }

  /**
  * actionPerformed - This method links all the classes with the buttons
  *                 created on the Graphical User Interface.
  */
  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource()==btnConfirm)
    {
      try
      {
        if(!txtDescription.getText().equals("") && !txtCompany.getText().equals("") &&
!txtPrice.getText().equals("") && !txtTaxRate.getText().equals("") && !txtBikeId.getText().equals(""))
        {

BikeRentalUI.addBikeToSell(txtDescription.getText(),txtCompany.getText(),Integer.parseInt(txtPrice.getT
ext()),Integer.parseInt(txtTaxRate.getText()),Integer.parseInt(txtBikeId.getText()));
          JOptionPane.showMessageDialog(frame,"The bike has been succesfully added for sale");
        }
        else
        {
          JOptionPane.showMessageDialog(frame,"You cannot leave the fields empty to continue");
        }
```

```
        }
        catch(NumberFormatException nfe)
        {
            JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE);
        }


      }
      else if(e.getSource() == btnClear)
      {
         txtDescription.setText("");
         txtPrice.setText("");
         txtTaxRate.setText("");
         txtCompany.setText("");
      }
   }
}
```

SELLBIKEUI CLASS

```
import javax.swing.JLabel;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.*;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
/**
 * SellBikeUI class contains the graphical user interface of the bike  selling page of the company
 *
 * @author (Manish)
 * @version (7272017)
 */
public class SellBikeUI implements ActionListener
{
   // instance variables - replcace the example below with your own
   private JFrame frame;

   private JLabel lblRentBike;
   private JLabel lblBikeId;
   private JLabel lblDescription;
   private JLabel lblCustomerName;
   private JLabel lblContact;
   private JLabel lblSellDate;
   private JLabel lblPrice;
   private JLabel lblTotalAmount;
   private JLabel lblCompany;
   private JLabel lblEmail;
   private JLabel lblTaxRate;

   private JButton btnCheck;
   private JButton btnClear;
```

```
private JButton btnConfirm;

private JTextField txtBikeId;
private JTextField txtDescription;
private JTextField txtCustomerName;
private JTextField txtContact;
private JTextField txtSellDate;
private JTextField txtPrice;
private JTextField txtTotalAmount;
private JTextField txtCompany;
private JTextField txtEmail;
private JTextField txtTaxRate;

private BikeToSell bike;

/**
 * Constructor for objects of class SellBike
 */
public SellBikeUI()
{
    // initialise instance variables
    SellBikeWindow();
}
```

```
/**
 * SellBikeWindow - This method creates the required labels and buttons
 *                  for this class.
 */
public void SellBikeWindow()
{
    frame = new JFrame("Everest Motors : Sell Bike");


    lblRentBike = new JLabel("Sell Bike");
    frame.add(lblRentBike);
    lblRentBike.setBounds(250,30,200,30);


    lblBikeId = new JLabel("Bike ID");
    frame.add(lblBikeId);
    lblBikeId.setBounds(40,80,150,40);


    lblDescription = new JLabel("Description");
    frame.add(lblDescription);
    lblDescription.setBounds(40,140,150,40);


    lblCustomerName = new JLabel("Customer's Name");
    frame.add(lblCustomerName);
    lblCustomerName.setBounds(40,200,150,40);


    lblContact = new JLabel("Contact Number");
```

```
frame.add(lblContact);
lblContact.setBounds(40,260,150,40);

lblSellDate = new JLabel("Sell Date");
frame.add(lblSellDate);
lblSellDate.setBounds(40,320,150,40);

lblPrice = new JLabel("Price");
frame.add(lblPrice);
lblPrice.setBounds(40,380,150,40);

lblTotalAmount = new JLabel("Total Amount");
frame.add(lblTotalAmount);
lblTotalAmount.setBounds(40,440,150,40);

lblCompany = new JLabel("Company");
frame.add(lblCompany);
lblCompany.setBounds(350,80,150,40);

lblEmail = new JLabel("E-mail");
frame.add(lblEmail);
lblEmail.setBounds(350,140,150,40);

lblTaxRate = new JLabel("TaxRate");
frame.add(lblTaxRate);
```

```
lblTaxRate.setBounds(350,200,150,40);

txtBikeId = new JTextField("");
frame.add(txtBikeId);
txtBikeId .setBounds(150,85,170,30);

txtDescription = new JTextField("");
frame.add(txtDescription);
txtDescription.setBounds(150,145,170,30);
txtDescription.setEditable(false);

txtCustomerName= new JTextField("");
frame.add(txtCustomerName);
txtCustomerName.setBounds(150,205,170,30);

txtContact= new JTextField("");
frame.add(txtContact);
txtContact.setBounds(150,265,170,30);

txtSellDate= new JTextField("");
frame.add(txtSellDate);
txtSellDate.setBounds(150,325,170,30);

txtPrice = new JTextField("");
frame.add(txtPrice);
```

```
txtPrice.setBounds(150,385,170,30);
txtPrice.setEditable(false);

txtTotalAmount = new JTextField("");
frame.add(txtTotalAmount);
txtTotalAmount.setBounds(150,445,170,30);
txtTotalAmount.setEditable(false);

txtCompany = new JTextField("");
frame.add(txtCompany);
txtCompany.setBounds(420,85,170,30);
txtCompany.setEditable(false);

txtEmail= new JTextField("");
frame.add(txtEmail);
txtEmail.setBounds(420,145,170,30);

txtTaxRate = new JTextField("");
frame.add(txtTaxRate);
txtTaxRate.setBounds(420,205,170,30);
txtTaxRate.setEditable(false);

btnCheck = new JButton("Check");
frame.add(btnCheck);
btnCheck.setBounds(360,445,120,35);
```

```
    btnCheck.addActionListener(this);

    btnClear = new JButton("Clear");
    frame.add(btnClear);
    btnClear.setBounds(360,490,120,35);
    btnClear.addActionListener(this);

    btnConfirm = new JButton("Confirm");
    frame.add(btnConfirm);
    btnConfirm.setBounds(490,490,120,35);
    btnConfirm.addActionListener(this);

    frame.setLayout(null);
    frame.setSize(650,600);
    frame.setVisible(true);
  }

  public void actionPerformed(ActionEvent e)
  {
    if(e.getSource()==btnCheck)
    {
      try
      {
        boolean bikeFound = false;
        boolean bikeAvailableStatus = false;
```

```
for (int i = 0; i < BikeRentalUI.list.size(); i++)
{
   if(BikeToSell.class.isInstance(BikeRentalUI.list.get(i)))
   {
      if(BikeRentalUI.list.get(i).getBikeId() == Integer.parseInt(txtBikeId.getText()))
      {
         bike = (BikeToSell) BikeRentalUI.list.get(i);
         if(!bike.getSellingStatus())
         {
          txtCompany.setText(bike.getBikeManufacturer());
          txtDescription.setText(bike.getBikeDescription());
          txtPrice.setText(Integer.toString(bike.getPrice()));
          txtTaxRate.setText(Integer.toString(bike.getTaxAmount()));
          txtTotalAmount.setText(Integer.toString(bike.getPrice()+bike.getTaxAmount()));
          bikeAvailableStatus = true;
         }
          bikeFound = true;
          break;
      }
   }

}

 if(bikeFound && !bikeAvailableStatus)
 {
```

```
            JOptionPane.showMessageDialog(frame,"The bike you searched for is not available for sale");


            }


            if(!bikeFound)
              {
               JOptionPane.showMessageDialog(frame,"The bike you searched for is not available");
              }
        }
        catch(NumberFormatException nfe)
        {
           JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE);
        }
      }
      else if(e.getSource()==btnClear)
      {
        txtBikeId.setText("");
        txtDescription.setText("");
        txtCustomerName.setText("");
        txtContact.setText("");
        txtSellDate.setText("");
        txtPrice.setText("");
        txtTotalAmount.setText("");
        txtCompany.setText("");
        txtEmail.setText("");
```

```
        txtTaxRate.setText("");
      }
      else if(e.getSource()==btnConfirm)
      {
        try
        {
          if(!txtBikeId.getText().equals("") && !txtCustomerName.getText().equals("") &&
!txtContact.getText().equals("") && !txtEmail.getText().equals("") && !txtSellDate.getText().equals(""))
          {
            for (int i = 0; i < BikeRentalUI.list.size(); i++)
            {
              if(BikeToSell.class.isInstance(BikeRentalUI.list.get(i)))
              {
                if(BikeRentalUI.list.get(i).getBikeId() == Integer.parseInt(txtBikeId.getText()))
                {
                  bike = (BikeToSell) BikeRentalUI.list.get(i);
                  boolean
bikeSellStatus=bike.bikeSellOut(txtCustomerName.getText(),txtContact.getText(),txtEmail.getText(),txtS
ellDate.getText());
                  if(!bikeSellStatus)
                  {
                    JOptionPane.showMessageDialog(frame,"The bike was sold on :
"+bike.getSellingDate());
                  }else {

                    JOptionPane.showMessageDialog(frame,"The bike has been sucessfully sold.");
                    }
```

```
                    break;
                }
            }
        }
    }
    else
    {
        JOptionPane.showMessageDialog(frame,"You cannot leave the fields empty inorder to continue");
    }
}
catch(NumberFormatException nfe)
{
    JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid number","Error",JOptionPane.ERROR_MESSAGE);
}
}
}
}
```

RENTBIKEUICLASS

```
import javax.swing.JLabel;

import javax.swing.JFrame;

import javax.swing.JButton;

import javax.swing.JTextField;

import javax.swing.*;


import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
/**
 * RentBikeUI class contains the graphical user interface of rental
 * page of the bike company
 *
 *
 * @author (Manish Giri)
 * @version (7232017)
 */
public class RentBikeUI implements ActionListener
{
   // instance variables
   private JFrame frame;

   private JLabel lblRentBike;
   private JLabel lblBikeId;
   private JLabel lblDescription;
   private JLabel lblCustomerName;
   private JLabel lblContact;
   private JLabel lblHireDate;
   private JLabel lblDailyRate;
   private JLabel lblTotalAmount;
   private JLabel lblCompany;
   private JLabel lblEmail;
   private JLabel lblDays;
```

```
    private JButton btnCheck;
    private JButton btnCalculateRent;
    private JButton btnClear;
    private JButton btnConfirm;

    private JTextField txtBikeId;
    private JTextField txtDescription;
    private JTextField txtCustomerName;
    private JTextField txtContact;
    private JTextField txtHireDate;
    private JTextField txtDailyRate;
    private JTextField txtTotalAmount;
    private JTextField txtCompany;
    private JTextField txtEmail;
    private JTextField txtDays;

    private BikeToRent bike;
    /**
     * Constructor for objects of class RentBike
     */
    public RentBikeUI()
    {
        // initialise instance variables
        RentBikeWindow();
```

```
}

/**
 * RentBikeWindow - This method creates the required labels and buttons
 *                  sfor this class.
 */
public void RentBikeWindow()
{
    frame = new JFrame("Everest Motors : Rent Bike");

    lblRentBike = new JLabel("Rent Bike");
    frame.add(lblRentBike);
    lblRentBike.setBounds(250,30,200,30);

    lblBikeId = new JLabel("Bike ID");
    frame.add(lblBikeId);
    lblBikeId.setBounds(40,80,150,40);

    lblDescription = new JLabel("Description");
    frame.add(lblDescription);
    lblDescription.setBounds(40,140,150,40);

    lblCustomerName = new JLabel("Customer's Name");
    frame.add(lblCustomerName);
    lblCustomerName.setBounds(40,200,150,40);
```

```
lblContact = new JLabel("Contact Number");
frame.add(lblContact);
lblContact.setBounds(40,260,150,40);

lblHireDate = new JLabel("Hire Date");
frame.add(lblHireDate);
lblHireDate.setBounds(40,320,150,40);

lblDailyRate = new JLabel("Daily Rate");
frame.add(lblDailyRate);
lblDailyRate.setBounds(40,380,150,40);

lblTotalAmount = new JLabel("Total Amount");
frame.add(lblTotalAmount);
lblTotalAmount.setBounds(40,440,150,40);

lblCompany = new JLabel("Company");
frame.add(lblCompany);
lblCompany.setBounds(350,80,150,40);

lblEmail = new JLabel("E-mail");
frame.add(lblEmail);
lblEmail.setBounds(350,140,150,40);
```

```
lblDays = new JLabel("No. of Days");
frame.add(lblDays);
lblDays.setBounds(350,200,150,40);

txtBikeId = new JTextField("");
frame.add(txtBikeId);
txtBikeId .setBounds(150,85,170,30);

txtDescription = new JTextField("");
frame.add(txtDescription);
txtDescription.setBounds(150,145,170,30);
txtDescription.setEditable(false);

txtCustomerName= new JTextField("");
frame.add(txtCustomerName);
txtCustomerName.setBounds(150,205,170,30);

txtContact= new JTextField("");
frame.add(txtContact);
txtContact.setBounds(150,265,170,30);

txtHireDate= new JTextField("");
frame.add(txtHireDate);
txtHireDate.setBounds(150,325,170,30);
```

```
txtDailyRate = new JTextField("");
frame.add(txtDailyRate);
txtDailyRate.setBounds(150,385,170,30);
txtDailyRate.setEditable(false);

txtTotalAmount = new JTextField("");
frame.add(txtTotalAmount);
txtTotalAmount.setBounds(150,445,170,30);
txtTotalAmount.setEditable(false);

txtCompany = new JTextField("");
frame.add(txtCompany);
txtCompany.setBounds(420,85,170,30);
txtCompany.setEditable(false);

txtEmail= new JTextField("");
frame.add(txtEmail);
txtEmail.setBounds(420,145,170,30);

txtDays = new JTextField("");
frame.add(txtDays);
txtDays.setBounds(420,205,170,30);

btnCheck = new JButton("Check");
frame.add(btnCheck);
```

```
    btnCheck.setBounds(360,390,110,35);
    btnCheck.addActionListener(this);

    btnCalculateRent = new JButton("Calculate Rent");
    frame.add(btnCalculateRent);
    btnCalculateRent.setBounds(480,390,110,35);
    btnCalculateRent.addActionListener(this);

    btnClear = new JButton("Clear");
    frame.add(btnClear);
    btnClear.setBounds(360,440,110,35);
    btnClear.addActionListener(this);

    btnConfirm = new JButton("Confirm");
    frame.add(btnConfirm);
    btnConfirm.setBounds(480,440,110,35);
    btnConfirm.addActionListener(this);

    frame.setLayout(null);
    frame.setSize(650,550);
    frame.setVisible(true);
  }

  /**
   * actionPerformed - This method links all the classes with the buttons
```

```
 *                 created on the Graphical User Interface.
 */
public void actionPerformed(ActionEvent e)
{
   if (e.getSource()==btnCheck)
   {
      try
      {
       boolean bikeFound = false;
       boolean bikeRentStatus = false;
       for (int i = 0; i < BikeRentalUI.list.size(); i++)
       {
        if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))
        {
           if(BikeRentalUI.list.get(i).getBikeId() == Integer.parseInt(txtBikeId.getText()))
           {
              bike = (BikeToRent) BikeRentalUI.list.get(i);
              if(!bike.getBikeLoanStatus())
              {
              txtCompany.setText(bike.getBikeManufacturer());
              txtDescription.setText(bike.getBikeDescription());
              txtDailyRate.setText(Integer.toString(bike.getDailyRate()));
              txtTotalAmount.setText(Integer.toString(bike.getTotalRentalCost()));
              bikeRentStatus = true;
             }
```

```
                bikeFound = true;
                break;
            }
        }
        }

        if(bikeFound && !bikeRentStatus)
        {
            JOptionPane.showMessageDialog(frame,"The bike you searched for is not available for rent");
        }

        if(!bikeFound)
          {
            JOptionPane.showMessageDialog(frame,"The bike you searched for is not available");
          }
        }
        catch(NumberFormatException nfe)
        {
          JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE);

        }

    }
    else if(e.getSource()==btnClear)
```

```
    {
        txtBikeId.setText("");
        txtDescription.setText("");
        txtCustomerName.setText("");
        txtContact.setText("");
        txtHireDate.setText("");
        txtDailyRate.setText("");
        txtTotalAmount.setText("");
        txtCompany.setText("");
        txtEmail.setText("");
        txtDays.setText("");
    }
    else if(e.getSource()==btnCalculateRent)
    {
        try
        {
            if (!txtDailyRate.getText().equals("") && !txtDays.getText().equals(""))
            {
                int rate = Integer.parseInt(txtDailyRate.getText());
                int noOfDays = Integer.parseInt(txtDays.getText());
                txtTotalAmount.setText(Integer.toString(rate*noOfDays));
            }
            else
            {
                JOptionPane.showMessageDialog(frame,"You cannot leave the fields Daily Rate and No Of
Days empty to calculate");
```

```
            }
          }
          catch(NumberFormatException nfe)
          {
            JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE);
          }
      }
      else if(e.getSource()==btnConfirm)
      {
        try
        {
          if(!txtBikeId.getText().equals("") && !txtCustomerName.getText().equals("") &&
!txtContact.getText().equals("") && !txtEmail.getText().equals("") && !txtDays.getText().equals("") &&
!txtHireDate.getText().equals""))
          {
            for (int i = 0; i < BikeRentalUI.list.size(); i++)
            {
              if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))
              {
                  if(BikeRentalUI.list.get(i).getBikeId() == Integer.parseInt(txtBikeId.getText()))
                    {
                        bike = (BikeToRent) BikeRentalUI.list.get(i);
                        boolean
bikeRentStatus=bike.rentOutBike(txtCustomerName.getText(),txtContact.getText(),txtEmail.getText(),txt
HireDate.getText(),Integer.parseInt(txtDays.getText()));
                        JOptionPane.showMessageDialog(frame,"The bike has been sucessfully rented.");
```

```
                    if(!bikeRentStatus)
                    {
                        JOptionPane.showMessageDialog(frame,"The bike was hired on : "+bike.getBikeHireDate()+" for : "+bike.getNumberOfDays()+" days.");
                    }
                    break;
                }
            }
        }
    }
    else
    {
        JOptionPane.showMessageDialog(frame,"You cannot leave the fields empty inorder to continue");
    }
  }
  catch(NumberFormatException nfe)
  {
    JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid number","Error",JOptionPane.ERROR_MESSAGE);
  }
 }
}
}
```

RETURNRENTBIKE CLASS

```
import javax.swing.JLabel;

import javax.swing.JFrame;
```

```
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.*;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 * ReturnRentBikeUI class contains the graphical user interface of the bike  selling page of the company
 *
 * @author (Manish)
 * @version (7272017)
 */
public class ReturnRentBikeUI implements ActionListener
{
   // instance variables
   private JFrame frame;

   private JLabel lblReturnRentBike;
   private JLabel lblBikeId;

   private JTextField txtBikeId;

   private JButton btnClear;
   private JButton btnConfirm;
   private BikeToRent bike;
```

```
/**
 * Constructor for objects of class ReturnRentBike
 */
public ReturnRentBikeUI()
{
    ReturnRentBikeWindow();
}
public void ReturnRentBikeWindow()
{
    // initialise instance variables
    frame = new JFrame("Everest Motors : Return Rent Bike");

    lblReturnRentBike = new JLabel("Return Rent Bike");
    frame.add(lblReturnRentBike);
    lblReturnRentBike.setBounds(180,30,200,30);

    lblBikeId = new JLabel("Bike ID");
    frame.add(lblBikeId);
    lblBikeId.setBounds(70,110,150,40);

    txtBikeId = new JTextField("");
    frame.add(txtBikeId);
    txtBikeId .setBounds(130,110,210,40);
```

```
    btnClear = new JButton("Clear");
    frame.add(btnClear);
    btnClear.setBounds(130,170,100,40);
    btnClear.addActionListener(this);

    btnConfirm = new JButton("Confirm");
    frame.add(btnConfirm);
    btnConfirm.setBounds(240,170,100,40);
    btnConfirm.addActionListener(this);

    frame.setLayout(null);
    frame.setSize(450,290);
    frame.setVisible(true);
  }

  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource()==btnClear)
    {
      txtBikeId.setText("");
    }
    else if(e.getSource()==btnConfirm)
    {
      if(!txtBikeId.getText().equals(""))
      {
```

```
    try
   {
   boolean bikeFound = false;
   boolean bikeRentStatus = false;
   for (int i = 0; i < BikeRentalUI.list.size(); i++)
   {
    if(BikeToRent.class.isInstance(BikeRentalUI.list.get(i)))
    {
      if(BikeRentalUI.list.get(i).getBikeId() == Integer.parseInt(txtBikeId.getText()))
      {
        bike = (BikeToRent) BikeRentalUI.list.get(i);
        if(bike.getBikeLoanStatus())
        {
         bike.makeBikeAvailable();
         bikeRentStatus = true;
         JOptionPane.showMessageDialog(frame,"The bike was on rent now its sucessfully freed for
you.");
        }
        bikeFound = true;
        break;
      }
    }
   }

    if(bikeFound && !bikeRentStatus)
   {
```

```
            JOptionPane.showMessageDialog(frame,"The bike you searched  is not currently on rent");
          }
          else
          {
          }

           if(!bikeFound)
             {
               JOptionPane.showMessageDialog(frame,"The bike you searched for is not available");
             }

         }
          catch(NumberFormatException nfe)
         {
             JOptionPane.showMessageDialog(frame,"Error Message\nPlease enter a valid
number","Error",JOptionPane.ERROR_MESSAGE);

         }
         }
         else
         {
            JOptionPane.showMessageDialog(frame,"The field cannot be empty inorder to continue");
         }
      }
   }
}
```