Object Oriented Software Development Module Code: CO4403

Lease Management System

Complete Source Code

Introduction

This file contain complete source code of Lease management System

Booking.java

```
package BackendCode;


import java.io.EOFException;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.util.ArrayList;


/**

 *

 * @author @AbdullahShahid01

 */

public class Booking implements Serializable {


    private int ID;
```

```
    private Customer customer;
    private Item item;
    private long RentTime, ReturnTime; // stores System time when the Book() method is
called

    public Booking() {
    }

    public Booking(int ID, Customer customer, Item item, long RentTime, long ReturnTime) {
        this.ID = ID;
        this.customer = customer;
        this.item = item;
        this.RentTime = RentTime;
        this.ReturnTime = ReturnTime;
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
```

```
    this.ID = ID;
}

public Customer getCustomer() {
    return customer;
}

public void setCustomer(Customer customer) {
    this.customer = customer;
}

public Item getItem() {
    return item;
}

public void setItem(Item item) {
    this.item = item;
}

public long getRentTime() {
    return RentTime;
```

```
    }

    public void setRentTime(long RentTime) {
        this.RentTime = RentTime;
    }

    public long getReturnTime() {
        return ReturnTime;
    }

    public void setReturnTime(long ReturnTime) {
        this.ReturnTime = ReturnTime;
    }

    @Override
    public String toString() {
        return "Booking{" + "ID=" + ID + ", \ncustomer=" + customer.toString() + ", \nitem=" +
item.toString() + ", \nRentTime=" + RentTime + ", ReturnTime=" + ReturnTime + '}' + "\n";
    }

    public void Add() {
```

```
ArrayList<Booking> booking = Booking.View();
if (booking.isEmpty()) {
    this.ID = 1;
} else {
    this.ID = booking.get(booking.size() - 1).ID + 1; // Auto ID ...
}
this.ReturnTime = 0;
booking.add(this);
File file = new File("Booking.ser");
if (!file.exists()) {
    try {
        file.createNewFile();
    } catch (IOException ex) {
        System.out.println(ex);
    }
}
ObjectOutputStream outputStream = null;
try {
    outputStream = new ObjectOutputStream(new FileOutputStream(file));
    for (int i = 0; i < booking.size(); i++) {
        outputStream.writeObject(booking.get(i));
```

```
        }
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    } catch (IOException ex) {
        System.out.println(ex);
    } finally {
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}

public void Update() {
    ArrayList<Booking> booking = Booking.View();

    // for loop for replacing the new Booking object with old one with same ID
    for (int i = 0; i < booking.size(); i++) {
```

```
    if (booking.get(i).ID == ID) {
        booking.set(i, this);
    }
}

// code for writing new Booking record
ObjectOutputStream outputStream = null;
try {
    outputStream = new ObjectOutputStream(new FileOutputStream("Booking.ser"));
    for (int i = 0; i < booking.size(); i++) {
        outputStream.writeObject(booking.get(i));
    }
} catch (FileNotFoundException ex) {
    System.out.println(ex);
} catch (IOException ex) {
    System.out.println(ex);
} finally {
    if (outputStream != null) {
        try {
            outputStream.close();
        } catch (IOException ex) {
```

```
            System.out.println(ex);
        }
    }
  }
}

public void Remove() {

  ArrayList<Booking> booking = Booking.View();
  // for loop for deleting the required Booking
  for (int i = 0; i < booking.size() - 1; i++) {
    if ((booking.get(i).ID == ID)) {

      for (int j = i; j < booking.size() - 1; j++) {
        booking.set(j, (booking.get(j + 1)));
      }

    }
  }
  // code for writing new Booking record
  ObjectOutputStream outputStream = null;
```

```
    try {
        outputStream = new ObjectOutputStream(new FileOutputStream("Booking.ser"));
        for (int i = 0; i < booking.size() - 1; i++) {
            outputStream.writeObject(booking.get(i));
        }
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    } catch (IOException ex) {
        System.out.println(ex);
    } finally {
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}

public int calculateBill() {
```

```
        // rent calculation
        long rentTime = this.getRentTime();
        long returnTime = this.getReturnTime();
        long totalTime = returnTime - rentTime;
        totalTime = totalTime / (1000 * 60 * 60);

        int rentPerHour = Integer.parseInt(this.getItem().getRentPerHour()) ;
        if (totalTime != 0) {
            return (int) (rentPerHour * totalTime);
        } else {
            return rentPerHour;
        }
    }

    public static ArrayList<Booking> SearchByCustomerID(int CustomerID) {
        ArrayList<Booking> bookingList = new ArrayList<>(0);
        ObjectInputStream inputStream = null;
        try {
// open file for reading
            inputStream = new ObjectInputStream(new FileInputStream("Booking.ser"));
            boolean EOF = false;
```

```
// Keep reading file until file ends
            while (!EOF) {
                try {
                    Booking myObj = (Booking) inputStream.readObject();
                    if (myObj.customer.getID() == CustomerID) {
                        bookingList.add(myObj);
                    }
                } catch (ClassNotFoundException e) {
                    System.out.println(e);
                } catch (EOFException end) {
                    EOF = true;
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        } finally {
            try {
                if (inputStream != null) {
                    inputStream.close();
```

```
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }
    return bookingList;
  }

  public static ArrayList<Booking> SearchByItemRegNo(String ItemRegNo) {
    System.out.println("Search by id "+ ItemRegNo);
    ArrayList<Booking> bookingList = new ArrayList<>(0);
    ObjectInputStream inputStream = null;
    try {
// open file for reading
        inputStream = new ObjectInputStream(new FileInputStream("Booking.ser"));
        boolean EOF = false;
// Keep reading file until file ends
        while (!EOF) {
            try {
                Booking myObj = (Booking) inputStream.readObject();
                System.out.println("Searching id "+ItemRegNo+ " In "+myObj);
```

```
            if (myObj.item.getID()== Integer.parseInt(ItemRegNo)) {
                bookingList.add(myObj);
            }
        } catch (ClassNotFoundException e) {
            System.out.println(e);
        } catch (EOFException end) {
            EOF = true;
        }
    }
} catch (FileNotFoundException e) {
    System.out.println(e);
} catch (IOException e) {
    System.out.println(e);
} finally {
    try {
        if (inputStream != null) {
            inputStream.close();
        }
    } catch (IOException e) {
        System.out.println(e);
    }
```

```
        }
        return bookingList;
    }

    public static ArrayList<Booking> SearchByItemID(int itemID) {
        ArrayList<Booking> bookingList = new ArrayList<>(0);
        ObjectInputStream inputStream = null;
        try {
// open file for reading
            inputStream = new ObjectInputStream(new FileInputStream("Booking.ser"));
            boolean EOF = false;
// Keep reading file until file ends
            while (!EOF) {
                try {
                    Booking myObj = (Booking) inputStream.readObject();
                    if (myObj.item.getID() == itemID) {
                        bookingList.add(myObj);
                    }
                } catch (ClassNotFoundException e) {
                    System.out.println(e);
                } catch (EOFException end) {
```

```
                EOF = true;
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println(e);
    } catch (IOException e) {
        System.out.println(e);
    } finally {
        try {
            if (inputStream != null) {
                inputStream.close();
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }
    return bookingList;
}

public static ArrayList<Booking> View() {
    ArrayList<Booking> bookingList = new ArrayList<>(0);
```

```
        ObjectInputStream inputStream = null;
        try {
// open file for reading
            inputStream = new ObjectInputStream(new FileInputStream("Booking.ser"));
            boolean EOF = false;
// Keep reading file until file ends
            while (!EOF) {
               try {
                  Booking myObj = (Booking) inputStream.readObject();
                  bookingList.add(myObj);
               } catch (ClassNotFoundException e) {
                  System.out.println(e);
               } catch (EOFException end) {
                  EOF = true;
               }
            }
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        } finally {
```

```
        try {
            if (inputStream != null) {
                inputStream.close();
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }
    return bookingList;
}

public static ArrayList<Item> getBookedItems() {
    ArrayList<Item> bookedItems = new ArrayList<>();
    ArrayList<Booking> bookings = Booking.View();
    for (int i = 0; i < bookings.size(); i++) {
        if (bookings.get(i).ReturnTime == 0) {
            bookedItems.add(bookings.get(i).item);
        }
    }
    return bookedItems;
}
```

```
    public static ArrayList<Item> getUnbookedItems() {
        ArrayList<Item> allItems = Item.View();
        ArrayList<Item> bookedItems = Booking.getBookedItems();
        for (int i = 0; i < bookedItems.size(); i++) {
            allItems.remove(bookedItems.get(i));
        }
        return allItems;
    }
}
```

Customer.java

```
package BackendCode;


import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;


/**
 *
 * @author @AbdullahShahid01
 */
public class Customer extends Person implements Serializable {

    private int Bill; // increases after every HOUR when a customers has Booked items(s)

    public Customer() {
        super();
    }

    public Customer(int Bill, int ID, String Name, String Contact_No,String Address) {
        super(ID, Name, Contact_No,Address);
        this.Bill = Bill;
    }
```

```
public int getBill() {
    return Bill;
}

public void setBill(int Bill) {
    this.Bill = Bill;
}

@Override
public String toString() {
    return super.toString() + "Customer{" + "Bill=" + Bill + '}' + "\n";
}

@Override
public void Add() {
    ArrayList<Customer> customers = Customer.View();
    if (customers.isEmpty()) {
        this.ID = 1;
    } else {
        this.ID = customers.get((customers.size() - 1)).ID + 1; // Auto ID...
```

```
}
customers.add(this);
File file = new File("Customer.ser");
if (!file.exists()) {
    try {
        file.createNewFile();
    } catch (IOException ex) {
        System.out.println(ex);
    }
}
ObjectOutputStream outputStream = null;
try {
    outputStream = new ObjectOutputStream(new FileOutputStream(file));
    for (int i = 0; i < customers.size(); i++) {
        outputStream.writeObject(customers.get(i));
    }
} catch (FileNotFoundException ex) {
    System.out.println(ex);
} catch (IOException ex) {
    System.out.println(ex);
} finally {
```

```
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}

@Override
public void Update() {
    ArrayList<Customer> customers = Customer.View();

    // for loop for replacing the new Customer object with old one with same ID
    for (int i = 0; i < customers.size(); i++) {
        if (customers.get(i).ID == ID) {
            customers.set(i, this);
        }
    }
```

```
    // code for writing new Customer record
    ObjectOutputStream outputStream = null;
    try {
        outputStream = new ObjectOutputStream(new FileOutputStream("Customer.ser"));
        for (int i = 0; i < customers.size(); i++) {
            outputStream.writeObject(customers.get(i));
        }
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    } catch (IOException ex) {
        System.out.println(ex);
    } finally {
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}
```

```
//////////////////////

@Override

public void Remove() {


    ArrayList<Customer> customers = Customer.View();


    // for loop for deleting the required Customer

    for (int i = 0; i < customers.size(); i++) {

        if (customers.get(i).ID == ID) {

            customers.remove(i);

        }

    }


    // code for writing new Customer record

    ObjectOutputStream outputStream = null;

    try {

        outputStream = new ObjectOutputStream(new FileOutputStream("Customer.ser"));

        for (int i = 0; i < customers.size(); i++) {

            outputStream.writeObject(customers.get(i));

        }
```

```
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    } catch (IOException ex) {
        System.out.println(ex);
    } finally {
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}

public static ArrayList<Customer> SearchByName(String name) {
    ArrayList<Customer> customers = Customer.View();
    ArrayList<Customer> s = new ArrayList<>();

    for (int i = 0; i < customers.size(); i++) {
        if (customers.get(i).Name.equalsIgnoreCase(name)) {
```

```
            s.add(customers.get(i));
        }
    }
    return s;
}

public static Customer SearchByAddress(String CustomerAddress) {
    ArrayList<Customer> customers = Customer.View();
    for (int i = 0; i < customers.size(); i++) {
        if (customers.get(i).Address.equalsIgnoreCase(CustomerAddress)) {
            return customers.get(i);
        }
    }
    return null;
}

public static Customer SearchByID(int id) {
    ArrayList<Customer> customers = Customer.View();
    for (int i = 0; i < customers.size(); i++) {
        if (customers.get(i).ID == id) {
            return customers.get(i);
```

```
        }
      }
      return null;
   }

   public static ArrayList<Customer> View() {
      ArrayList<Customer> CustomerList = new ArrayList<>(0);
      ObjectInputStream inputStream = null;
      try {
// open file for reading
         inputStream = new ObjectInputStream(new FileInputStream("Customer.ser"));
         boolean EOF = false;
// Keep reading file until file ends
         while (!EOF) {
            try {
               Customer myObj = (Customer) inputStream.readObject();
               CustomerList.add(myObj);
            } catch (ClassNotFoundException e) {
               System.out.println(e);
            } catch (EOFException end) {
               EOF = true;
```

```
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println(e);
    } catch (IOException e) {
        System.out.println(e);
    } finally {
        try {
            if (inputStream != null) {
                inputStream.close();
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }
    return CustomerList;
  }


}
```

Item.java

```
package BackendCode;


import java.io.EOFException;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.util.ArrayList;


/**

 *

 * @author @AbdullahShahid01

 */

public class Item implements Serializable {


    int ID;
```

```
    String Name, Model, Type,Specifications;
    String length,breadth, height,weight,RentPerHour;


    @Override
    public String toString() {
        return "item_new{" +
                "ID=" + ID +
                ", Name=" + Name +
                ", Model=" + Model +
                ", \nType=" + Type +
                ", Specification=" + Specifications +
                ", length=" + length +
                ", breadth=" + breadth +
                ", height=" + height +
                ", weight=" + weight +
                ", RentPerHour=" + RentPerHour +
                '}' + "\n";
    }
    public Item(int ID, String Name, String Model, String Type, String Specifications, String
length, String breadth, String height, String weight, String RentPerHour) {
        this.ID = ID;
```

```
        this.Name = Name;
        this.Model = Model;
        this.Type = Type;
        this.Specifications = Specifications;
        this.length = length;
        this.breadth = breadth;
        this.height = height;
        this.weight = weight;
        this.RentPerHour = RentPerHour;
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public String getName() {
        return Name;
```

```
}

public void setName(String Name) {
    this.Name = Name;
}

public String getModel() {
    return Model;
}

public void setModel(String Model) {
    this.Model = Model;
}

public String getType() {
    return Type;
}

public void setType(String Type) {
    this.Type = Type;
}
```

```
public String getSpecifications() {
    return Specifications;
}

public void setSpecifications(String Specifications) {
    this.Specifications = Specifications;
}

public String getLength() {
    return length;
}

public void setLength(String length) {
    this.length = length;
}

public String getBreadth() {
    return breadth;
}
```

```
public void setBreadth(String breadth) {
    this.breadth = breadth;
}

public String getHeight() {
    return height;
}

public void setHeight(String height) {
    this.height = height;
}

public String getWeight() {
    return weight;
}

public void setWeight(String weight) {
    this.weight = weight;
}

public String getRentPerHour() {
```

```
    return RentPerHour;
}

public void setRentPerHour(String RentPerHour) {
    this.RentPerHour = RentPerHour;
}

public Item() {}




public void Add() {
    ArrayList<Item> leaseItem = Item.View();
    if (leaseItem.isEmpty()) {
        this.ID = 1;
    } else {
        this.ID = leaseItem.get(leaseItem.size() - 1).ID + 1; // Auto ID...
    }
    leaseItem.add(this);
    File file = new File("item.ser");
    if (!file.exists()) {
```

```
        try {
            file.createNewFile();
        } catch (IOException ex) {
            System.out.println(ex);
        }
    }
    ObjectOutputStream outputStream = null;
    try {
        outputStream = new ObjectOutputStream(new FileOutputStream(file));
        for (int i = 0; i < leaseItem.size(); i++) {
            System.out.println("+ adding "+leaseItem.get(i));
            outputStream.writeObject(leaseItem.get(i));
        }
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    } catch (IOException ex) {
        System.out.println(ex);
    } finally {
        if (outputStream != null) {
            try {
                outputStream.close();
```

```
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}


public void Update() {
    ArrayList<Item> leaseItem = Item.View();

    // for loop for replacing the new Item object with old one with same ID
    for (int i = 0; i < leaseItem.size(); i++) {
        if (leaseItem.get(i).ID == ID) {
            leaseItem.set(i, this);
        }
    }

    // code for writing new Item record
    ObjectOutputStream outputStream = null;
    try {
```

```
            outputStream = new ObjectOutputStream(new FileOutputStream("item.ser"));
            for (int i = 0; i < leaseItem.size(); i++) {
                outputStream.writeObject(leaseItem.get(i));
            }
        } catch (FileNotFoundException ex) {
            System.out.println(ex);
        } catch (IOException ex) {
            System.out.println(ex);
        } finally {
            if (outputStream != null) {
                try {
                    outputStream.close();
                } catch (IOException ex) {
                    System.out.println(ex);
                }
            }
        }
    }

    public void Remove() {
```

```
ArrayList<Item> leaseItem = Item.View();
// for loop for deleting the required Item
for (int i = 0; i < leaseItem.size(); i++) {
    if ((leaseItem.get(i).ID == ID)) {
        leaseItem.remove(i);
    }
}
// code for writing new Item record
ObjectOutputStream outputStream = null;
try {
    outputStream = new ObjectOutputStream(new FileOutputStream("item.ser"));
    for (int i = 0; i < leaseItem.size(); i++) {
        outputStream.writeObject(leaseItem.get(i));
    }
} catch (FileNotFoundException ex) {
    System.out.println(ex);
} catch (IOException ex) {
    System.out.println(ex);
} finally {
    if (outputStream != null) {
        try {
```

```
                outputStream.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}

public static ArrayList<Item> SearchByName(String name) {
    ArrayList<Item> leaseItem = Item.View();
    ArrayList<Item> s = new ArrayList<>();
    for (int i = 0; i < leaseItem.size(); i++) {
        if (leaseItem.get(i).Name.equalsIgnoreCase(name)) {
            s.add(leaseItem.get(i));
        }
    }
    return s;
}

public static Item SearchByID(int id) {
    ArrayList<Item> leaseItem = Item.View();
```

```
    for (int i = 0; i < leaseItem.size(); i++) {
        System.out.println("Lease item id "+leaseItem.get(i).ID);
        if (leaseItem.get(i).ID == id) {
            return leaseItem.get(i);
        }
    }
    return null;
}

public static Item SearchByRegNo(String model) {
    ArrayList<Item> leaseItem = Item.View();
    for (int i = 0; i < leaseItem.size(); i++) {
        if (leaseItem.get(i).Model.equalsIgnoreCase(model)) {
            return leaseItem.get(i);
        }
    }
    return null;
}

public static ArrayList<Item> View() {
    ArrayList<Item> itemList = new ArrayList<>(0);
```

```
        ObjectInputStream inputStream = null;
        try {
// open file for reading
            inputStream = new ObjectInputStream(new FileInputStream("item.ser"));
            boolean EOF = false;
// Keep reading file until file ends
            while (!EOF) {
               try {
                  Item myObj = (Item) inputStream.readObject();
                  itemList.add(myObj);
               } catch (ClassNotFoundException e) {
                  System.out.println(e);
               } catch (EOFException end) {
                  EOF = true;
               }
            }
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        } finally {
```

```
            try {
                if (inputStream != null) {
                    inputStream.close();
                }
            } catch (IOException e) {
                System.out.println(e);
            }
        }
        return itemList;
    }


    public static boolean isNameValid(String Name) {
        boolean flag = false;
        for (int i = 0; i < Name.length(); i++) {
//            Name can contain white spaces
            if (Character.isLetter(Name.charAt(i)) |Character.isDigit(Name.charAt(i))|
Name.charAt(i) == ' ') {
                flag = true;
            } else {
                flag = false;
                break;
```

```
        }
    }
    return flag;
}

public static boolean isRegNoValid(String RegNo) {
    if(Integer.parseInt(RegNo)>0){
        return true;
    }else{
        return false;
    }
}

public boolean isRented() {
    ArrayList<Item> BookedItems = Booking.getBookedItems();
    for (int i = 0; i < BookedItems.size(); i++) {
        if (BookedItems.get(i).ID == this.ID) {
            return true;
        }
    }
    return false;
```

```
    }

}
```

Person.java

```
package BackendCode;

import java.io.Serializable;

/**
 *
 * @author @AbdullahShahid01
 */
public abstract class Person implements Serializable {

    protected int ID;
    protected String Name, Contact_No,Address;

    public Person() {
    }
```

```
public Person(int ID, String Address, String Name, String Contact_No) {
    this.ID = ID;
    this.Address = Address;
    this.Name = Name;
    this.Contact_No = Contact_No;
}
public int getID() {
    return ID;
}
public void setID(int ID) {
    this.ID = ID;
}
public String getAddress() {
    return Address;
}
public void setAddress(String Address) {
    this.Address = Address;
}
public String getName() {
    return Name;
}
```

```
    public void setName(String Name) {
        this.Name = Name;
    }
    public String getContact_No() {
        return Contact_No;
    }
    public void setContact_No(String Contact_No) {
        this.Contact_No = Contact_No;
    }
    public abstract void Add();
    public abstract void Update();
    public abstract void Remove();
    @Override
    public String toString() {
        return "Person_new{" + "ID=" + ID + ", Address=" + Address + ", Name=" + Name + ",
Contact_No=" + Contact_No + '}';
    }

    /**
     * A valid name can contain only letters and white spaces
     * @param Name
```

```
     * @return true if the name is valid
     */
    public static boolean isNameValid(String Name) {
        boolean flag = false;
        for (int i = 0; i < Name.length(); i++) {
//            Name can contain white spaces
            if (Character.isLetter(Name.charAt(i)) | Name.charAt(i) == ' ') {
                flag = true;
            } else {
                flag = false;
                break;
            }
        }
        return flag;
    }
    /**
     * A valid ID can only be digit greater than 0
     * @param ID
     * @return true if the ID is valid
     */
    public static boolean isIDvalid(String ID) {
```

```
        boolean flag = true;
        for (int i = 0; i < ID.length(); i++) {
            if (!Character.isDigit(ID.charAt(i))) {
                flag = false;
                break;
            }
        }
        if (flag) {
            if (Integer.parseInt(ID) <= 0) {
                flag = false;
            }
        }
        return flag;
    }
}
```

Booking_BookItem.java

```
package GUI;


import BackendCode.Booking;
import BackendCode.Item;
```

```
import BackendCode.Customer;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.*;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;


/**
 *
 * @author @AbdullahShahid01
 */
public class Booking_BookItem extends JFrame {

    JButton Book_Button, Cancel_Button;
    JLabel ItemID_Label, ItemIDValidity_Label, CustomerID_Label, CustomerIDValidity_Label;
    JTextField ItemID_TextField, CustomerID_TextField;

    private Item item;
    private Customer customer;
```

```
public Booking_BookItem() {
    super("Book Item");
    setLayout(new FlowLayout());
    setSize(new Dimension(300, 200));
    setResizable(false);
    setLocationRelativeTo(this);
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        }
    });

    Book_Button = new JButton("Book");
    Cancel_Button = new JButton("Cancel");

    ItemID_Label = new JLabel("Enter Item ID to be Booked");
    ItemIDValidity_Label = new JLabel();
```

```
ItemID_TextField = new JTextField();


CustomerID_Label = new JLabel("Enter Customer ID");
CustomerIDValidity_Label = new JLabel();
CustomerID_TextField = new JTextField();


ItemID_TextField.setPreferredSize(new Dimension(240, 22));
ItemIDValidity_Label.setPreferredSize(new Dimension(240, 9));


CustomerID_TextField.setPreferredSize(new Dimension(240, 22));
CustomerIDValidity_Label.setPreferredSize(new Dimension(240, 9));


Book_Button.setPreferredSize(new Dimension(100, 22));
Cancel_Button.setPreferredSize(new Dimension(100, 22));


ItemIDValidity_Label.setForeground(Color.red);
CustomerIDValidity_Label.setForeground(Color.red);


add(ItemID_Label);
add(ItemID_TextField);
add(ItemIDValidity_Label);
```

```
add(CustomerID_Label);
add(CustomerID_TextField);
add(CustomerIDValidity_Label);

add(Book_Button);
add(Cancel_Button);

Book_Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        String ItemID = ItemID_TextField.getText().trim();
        if (!ItemID.isEmpty()) {
            try {
                if (Integer.parseInt(ItemID) > 0) {
                    ItemIDValidity_Label.setText("");
                    item = Item.SearchByID(Integer.parseInt(ItemID));
                    if (item != null) {
                        if (!item.isRented()) {
                            ItemIDValidity_Label.setText("");
```

```
                    } else {
                        item = null;
//                          JOptionPane.showMessageDialog(null, "This item is already booked
!");
                        ItemIDValidity_Label.setText( "This item is already booked !");
                    }
                } else {
                    ItemID = null;
                    ItemIDValidity_Label.setText("Item ID does not exists !");
                }
            } else {
                ItemID = null;
                ItemIDValidity_Label.setText("ID cannot be '0' or negative !");
            }
        } catch (NumberFormatException ex) {
            ItemID = null;
            ItemIDValidity_Label.setText("Invalid ID !");
        }
    } else {
        ItemID = null;
        ItemIDValidity_Label.setText("Enter Item ID !");
```

```
        }


        String customerID = CustomerID_TextField.getText().trim();
        if (!customerID.isEmpty()) {
            try {
                if (Integer.parseInt(customerID) > 0) {
                    CustomerIDValidity_Label.setText("");
                    customer = Customer.SearchByID(Integer.parseInt(customerID));
                    if (customer != null) {
                        CustomerIDValidity_Label.setText("");
                    } else {
                        customerID = null;
//                          JOptionPane.showMessageDialog(null, "Customer ID does not exists !");
                        CustomerIDValidity_Label.setText("Customer ID does not exists !");
                    }
                } else {
                    customerID = null;
                    CustomerIDValidity_Label.setText("ID cannot be '0' or negative !");
                }
            } catch (NumberFormatException ex) {
                customerID = null;
```

```
                CustomerIDValidity_Label.setText("Invalid ID !");
            }
        } else {
            customerID = null;
            CustomerIDValidity_Label.setText("Enter Customer ID !");
        }

        if (ItemID != null & customerID != null) {
            setEnabled(false);
            int showConfirmDialog = JOptionPane.showConfirmDialog(null,
                    "You are about to Book the Item: \n" + item.toString() + "\n against the
Customer: \n"
                    + customer.toString() + "\n Are you sure you want to continue??",
                    "Book Confirmation", JOptionPane.OK_CANCEL_OPTION);
            if (showConfirmDialog == 0) {
                Booking booking = new Booking(0, customer, item,
System.currentTimeMillis(), 0);
                booking.Add();
                Parent_JFrame.getMainFrame().getContentPane().removeAll();
                Booking_Details cd = new Booking_Details();
                Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                Parent_JFrame.getMainFrame().getContentPane().revalidate();
```

```
                JOptionPane.showMessageDialog(null, "Item Successfully Booked !");
                Parent_JFrame.getMainFrame().setEnabled(true);
                dispose();
            }


        }
      }
    }
    );
    Cancel_Button.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        Parent_JFrame.getMainFrame().setEnabled(true);
        dispose();
      }
    });
  }
}
```

Booking_Details.java

```
package GUI;


import BackendCode.Booking;
import BackendCode.Item;
import BackendCode.Customer;
import java.awt.Dimension;
import javax.swing.table.DefaultTableModel;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
```

```
import javax.swing.JTextField;
import javax.swing.table.DefaultTableCellRenderer;


public class Booking_Details {


   private static DefaultTableModel tablemodel; // it is made static so that it can be accessed in add GUI class to update the Jtable when a new record is added


   private static JButton SearchCustomerID_Button, SearchItemRegNo_Button,
         BackButton, LogoutButton, BookItem_Button, UnbookItem_Button;
   private static JTextField CustomerID_TextField, ItemRegNo_TextField;
   private static JScrollPane jScrollPane1;
   private static JTable jTable1;
   private JPanel MainPanel;


   public Booking_Details() {
      MainPanel = new JPanel();
      Parent_JFrame.getMainFrame().setTitle("Booking Details - Lease-A-Item Management System");
      MainPanel.setLayout(new AbsoluteLayout());
      MainPanel.setMinimumSize(new Dimension(1366, 730));
```

```
        SearchCustomerID_Button = new JButton("Search by Customer ID");
        SearchItemRegNo_Button = new JButton("Search by Item id");
        BackButton = new JButton("Back");
        LogoutButton = new JButton("Logout");
        BookItem_Button = new JButton("Book");
        UnbookItem_Button = new JButton("Unbook");

        CustomerID_TextField = new JTextField();
        ItemRegNo_TextField = new JTextField();

        jScrollPane1 = new JScrollPane();
        jTable1 = new JTable();
//ID, Maker, Name, Colour, Type, SeatingCapacity, Model, Condition, RegNo,
RentPerHour, IsRented RentDate, itemOwner customer

        String[] columns = {"Sr#", "ID", "Customer ID+Name", "Item Name", "Rent Time",
"Return Time"};
        tablemodel = new DefaultTableModel(columns, 0) {

            @Override
            public boolean isCellEditable(int row, int column) {
                //all cells false
```

```
            return false;
        }
    };

    jTable1 = new JTable(getTablemodel());
    jTable1.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
    jScrollPane1 = new JScrollPane();
    jScrollPane1.setViewportView(jTable1);
    jTable1.setFillsViewportHeight(true);// makes the size of table equal to that of scroll pane to fill the table in the scrollpane
    ArrayList<Booking> Booking_objects = Booking.View();
    for (int i = 0; i < Booking_objects.size(); i++) {
//ID, Maker, Name, Colour, Type, SeatingCapacity, Model, Condition, RegNo,
//RentPerHour, IsRented RentDate, ItemOwner customer
        int ID = Booking_objects.get(i).getID();
        String customer_ID_Name = Booking_objects.get(i).getCustomer().getID()
                + ": " + Booking_objects.get(i).getCustomer().getName();
        String itemName = Booking_objects.get(i).getItem().getName();
        String itemID = Booking_objects.get(i).getItem().getID()+"";
        SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm a dd-MM-yyyy");
        Date rentime = new Date(Booking_objects.get(i).getRentTime());
```

```
        String rentTime = dateFormat.format(rentime);


        long returnTime_ = Booking_objects.get(i).getReturnTime();
        String returnTime;
        if (returnTime_ != 0) {
            Date returntime = new Date(returnTime_);
            returnTime = dateFormat.format(returntime);
        } else {
            returnTime = "Not returned yet !";
        }


        String[] one_s_Record = {((i + 1) + ""), "" + ID, customer_ID_Name, itemID+":
"+itemName, rentTime, returnTime};
        tablemodel.addRow(one_s_Record);
    }


    // center aligning the text in all the columns
    DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
    centerRenderer.setHorizontalAlignment(JLabel.CENTER);
    jTable1.getColumnModel().getColumn(0).setCellRenderer(centerRenderer);
    jTable1.getColumnModel().getColumn(1).setCellRenderer(centerRenderer);
```

```
jTable1.getColumnModel().getColumn(2).setCellRenderer(centerRenderer);
jTable1.getColumnModel().getColumn(3).setCellRenderer(centerRenderer);
jTable1.getColumnModel().getColumn(4).setCellRenderer(centerRenderer);
jTable1.getColumnModel().getColumn(5).setCellRenderer(centerRenderer);

// adjusting size of each column
jTable1.getColumnModel().getColumn(0).setMinWidth(80);
jTable1.getColumnModel().getColumn(1).setMinWidth(80);
jTable1.getColumnModel().getColumn(2).setMinWidth(400);
jTable1.getColumnModel().getColumn(3).setMinWidth(300);
jTable1.getColumnModel().getColumn(4).setMinWidth(230);
jTable1.getColumnModel().getColumn(5).setMinWidth(235);

jTable1.getTableHeader().setReorderingAllowed(false);

MainPanel.add(jScrollPane1, new AbsoluteConstraints(10, 60, 1330, 550));
MainPanel.add(BackButton, new AbsoluteConstraints(1106, 625, 100, 22));
MainPanel.add(LogoutButton, new AbsoluteConstraints(1236, 625, 100, 22));
MainPanel.add(BookItem_Button, new AbsoluteConstraints(10, 625, 130, 22));
MainPanel.add(UnbookItem_Button, new AbsoluteConstraints(160, 625, 130, 22));
```

```
    MainPanel.add(SearchItemRegNo_Button, new AbsoluteConstraints(10, 15, 160, 22));
    MainPanel.add(ItemRegNo_TextField, new AbsoluteConstraints(185, 15, 240, 22));
    MainPanel.add(SearchCustomerID_Button, new AbsoluteConstraints(440, 15, 180, 22));
    MainPanel.add(CustomerID_TextField, new AbsoluteConstraints(635, 15, 240, 22));

    SearchCustomerID_Button.addActionListener(new Booking_Details_ActionListener());
    SearchItemRegNo_Button.addActionListener(new Booking_Details_ActionListener());
    BackButton.addActionListener(new Booking_Details_ActionListener());
    LogoutButton.addActionListener(new Booking_Details_ActionListener());
    BookItem_Button.addActionListener(new Booking_Details_ActionListener());
    UnbookItem_Button.addActionListener(new Booking_Details_ActionListener());
}

public static DefaultTableModel getTablemodel() {
    return tablemodel;
}

public JPanel getMainPanel() {
    return MainPanel;
}
```

```
    private class Booking_Details_ActionListener implements ActionListener {


        @Override
        public void actionPerformed(ActionEvent e) {


            switch (e.getActionCommand()) {


                case "Back": {
                    Parent_JFrame.getMainFrame().setTitle("Lease-A-Item Management System
[REBORN]");
                    MainMenu mm = new MainMenu();
                    Parent_JFrame.getMainFrame().getContentPane().removeAll();
                    Parent_JFrame.getMainFrame().add(mm.getMainPanel());
                    Parent_JFrame.getMainFrame().getContentPane().revalidate();
                }
                break;
                case "Logout": {
                    Parent_JFrame.getMainFrame().dispose();
                    Runner r = new Runner();
                    JFrame frame = r.getFrame();
                    Login login = new Login();
```

```
    JPanel panel = login.getMainPanel();
    frame.add(panel);
    frame.setVisible(true);
}
break;
case "Book": {
    if (!Booking.getUnbookedItems().isEmpty()) {
        Parent_JFrame.getMainFrame().setEnabled(false);
        Booking_BookItem ac = new Booking_BookItem();
        ac.setVisible(true);
    } else {
        JOptionPane.showMessageDialog(null, "No UnBooked Items are available !");
    }
}
break;
case "Unbook": {
    if (!Booking.getBookedItems().isEmpty()) {
        Parent_JFrame.getMainFrame().setEnabled(false);
        Booking_UnBookItem ac = new Booking_UnBookItem();
        ac.setVisible(true);
    } else {
```

```
                JOptionPane.showMessageDialog(null, "No Booked Items found !");
            }
        }
        break;
        case "Search by Customer ID": {
            String customerID = CustomerID_TextField.getText().trim();
            if (!customerID.isEmpty()) {
                if (Customer.isIDvalid(customerID)) {
                    Customer customer = Customer.SearchByID(Integer.parseInt(customerID));
                    if (customer != null) {
                        ArrayList<Booking> bookings =
Booking.SearchByCustomerID(Integer.parseInt(customerID));
                        if (!bookings.isEmpty()) {
                            JOptionPane.showMessageDialog(null, bookings.toString());
                        } else {
                            JOptionPane.showMessageDialog(null, "This Customer has not booked
any items yet !");
                        }
                    } else {
                        JOptionPane.showMessageDialog(null, "Customer ID not found !");
                    }
                } else {
```

```
            JOptionPane.showMessageDialog(null, "Invalid Customer ID !");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Enter Customer ID first !");
    }
    CustomerID_TextField.setText("");
}
break;
case "Search by Item id": {
    String itemRegNo = ItemRegNo_TextField.getText().trim();
    System.out.println("Searching by regnumber 215"+itemRegNo);
    if (!itemRegNo.isEmpty()) {
        System.out.println("Searching by regnumber 217"+itemRegNo);
        if (Item.isRegNoValid(itemRegNo)) {
            Item item = Item.SearchByID(Integer.parseInt(itemRegNo));
            if (item != null) {
                ArrayList<Booking> bookings = Booking.SearchByItemRegNo(itemRegNo);
                if (!bookings.isEmpty()) {
                    JOptionPane.showMessageDialog(null, bookings.toString());
                } else {
                    JOptionPane.showMessageDialog(null, "This Item is not booked yet !");
```

```
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Id no. not found !");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Invalid Id no !");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Enter Item id No first !");
        }
        CustomerID_TextField.setText("");
      }
      break;
    }
  }
}
```

Booking_UnBookItem.java

```
package GUI;
```

```
import BackendCode.Booking;
import BackendCode.Item;
import BackendCode.Customer;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.*;
import static javax.swing.JOptionPane.OK_CANCEL_OPTION;

/**
 *
 * @author @AbdullahShahid01
 */
public class Booking_UnBookItem extends JFrame {

    JButton UnBook_Button, Cancel_Button;
    JLabel ItemID_Label, ItemIDValidity_Label;
    JTextField ItemID_TextField;

    private Item item;
```

```
public Booking_UnBookItem() {
    super("UnBook Item");
    setLayout(new FlowLayout());
    setSize(new Dimension(300, 145));
    setResizable(false);
    setLocationRelativeTo(this);
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        }
    });

    UnBook_Button = new JButton("UnBook");
    Cancel_Button = new JButton("Cancel");

    ItemID_Label = new JLabel("Enter Item ID to be UnBooked");
    ItemIDValidity_Label = new JLabel();
```

```
ItemID_TextField = new JTextField();


ItemID_TextField.setPreferredSize(new Dimension(240, 22));
ItemIDValidity_Label.setPreferredSize(new Dimension(415, 9));


UnBook_Button.setPreferredSize(new Dimension(100, 22));
Cancel_Button.setPreferredSize(new Dimension(100, 22));


ItemIDValidity_Label.setForeground(Color.red);


add(ItemID_Label);
add(ItemID_TextField);
add(ItemIDValidity_Label);


add(UnBook_Button);
add(Cancel_Button);


UnBook_Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

```
String itemID = ItemID_TextField.getText().trim();
if (!itemID.isEmpty()) {
    try {
        if (Integer.parseInt(itemID) > 0) {
            ItemIDValidity_Label.setText("");
            item = Item.SearchByID(Integer.parseInt(itemID));
            if (item != null) {
                if (item.isRented()) {
                    ItemIDValidity_Label.setText("");
                } else {
                    item = null;
                    JOptionPane.showMessageDialog(null, "This item is not booked !");
                }
            } else {
                item = null;
                JOptionPane.showMessageDialog(null, "Item ID does not exists !");
            }
        } else {
            itemID = null;
            ItemIDValidity_Label.setText("                                                    "
                    + "ID cannot be '0' or negative !");
```

```
                }
            } catch (NumberFormatException ex) {
                itemID = null;
                ItemIDValidity_Label.setText("                                    "
                        + "Invalid ID !");
            }
        } else {
            itemID = null;
            ItemIDValidity_Label.setText("                                    "
                    + "Enter Item ID !");
        }

        if (itemID != null && item != null) {
            setEnabled(false);
            int showConfirmDialog = JOptionPane.showConfirmDialog(null, "You are about
to UnBook this Item\n" + item.toString()
                    + "\n Are you sure you want to continue ??", "UnBook Confirmation",
OK_CANCEL_OPTION);
            if (showConfirmDialog == 0) {

                ArrayList<Booking> booking =
Booking.SearchByItemID(Integer.parseInt(itemID));
```

```
            Booking last = booking.get((booking.size() - 1));
            last.setReturnTime(System.currentTimeMillis());
            last.Update();

            int bill = last.calculateBill();

            Customer customer = last.getCustomer();
            customer.setBill(customer.getBill()+bill);
            customer.Update();

            Parent_JFrame.getMainFrame().getContentPane().removeAll();
            Booking_Details cd = new Booking_Details();
            Parent_JFrame.getMainFrame().add(cd.getMainPanel());
            Parent_JFrame.getMainFrame().getContentPane().revalidate();
            JOptionPane.showMessageDialog(null, "Item Successfully UnBooked !");
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        } else {
            setEnabled(true);
        }
    }
```

```
        }
    }
    );
    Cancel_Button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        }
    });
  }

}
```

Customer_Add.java

```
package GUI;


import BackendCode.Customer;
import java.awt.*;
import java.awt.event.*;
```

```
import javax.swing.*;

import org.netbeans.lib.awtextra.AbsoluteConstraints;

import org.netbeans.lib.awtextra.AbsoluteLayout;


public class Customer_Add {


    JButton Add_Button, Cancel_Button;

    JLabel Name_Label,Address_Label,Password_Label;

    JLabel CNIC_Label,  Contact_Label, Email_Label, UserName_Label,  CNICValidity_Label, contactValidity_Label, NameValidity_Label, AddressValidity_Label, UserNameValidity_Label, PasswordValidity_Label;

    JTextField CNIC_TextField, Name_TextField, Contact_TextField, Address_TextField, UserName_TextField, Password_TextField;

    JFrame frame = new JFrame();


    public Customer_Add() {

        frame.setTitle("Add Customer");

        frame.setLayout(new AbsoluteLayout());

        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

        frame.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {

                Parent_JFrame.getMainFrame().setEnabled(true);
```

```
            frame.dispose();
        }
    });

    frame.setSize(new Dimension(450, 290));
    frame.setResizable(false);
    frame.setLocationRelativeTo(Parent_JFrame.getMainFrame());

    Add_Button = new JButton("Add");
    Cancel_Button = new JButton("Cancel");

//      CNIC_Label = new JLabel("Enter CNIC (without dashes)");
    Name_Label = new JLabel("Enter Name");
//      Phone_label = new JLabel("Enter Phone");

    Contact_Label = new JLabel("Enter Contact");
    Address_Label = new JLabel("Enter Address");
//      UserName_Label = new JLabel("Enter Username");
    Password_Label = new JLabel("Enter Password");

//      CNICValidity_Label = new JLabel();
```

```
        NameValidity_Label = new JLabel();
        AddressValidity_Label = new JLabel();
//      UserNameValidity_Label = new JLabel();
        PasswordValidity_Label = new JLabel();
        contactValidity_Label = new JLabel();

//      CNIC_TextField = new JTextField();
        Name_TextField = new JTextField();
        Contact_TextField = new JTextField();
        Address_TextField = new JTextField();
//      UserName_TextField = new JTextField();
        Password_TextField = new JTextField();

//      CNIC_TextField.setPreferredSize(new Dimension(240, 22));
        Name_TextField.setPreferredSize(new Dimension(240, 22));
        Contact_TextField.setPreferredSize(new Dimension(240, 22));
        Address_TextField.setPreferredSize(new Dimension(240, 22));
//      UserName_TextField.setPreferredSize(new Dimension(240, 22));
        Password_TextField.setPreferredSize(new Dimension(240, 22));

//      CNIC_Label.setPreferredSize(new Dimension(175, 22));
```

```
        Name_Label.setPreferredSize(new Dimension(175, 22));
//      Contact_Label.setPreferredSize(new Dimension(175, 22));
//      Email_Label.setPreferredSize(new Dimension(175, 22));
//      UserName_Label.setPreferredSize(new Dimension(175, 22));
        Password_Label.setPreferredSize(new Dimension(175, 22));
//      CNICValidity_Label.setPreferredSize(new Dimension(240, 9));
        contactValidity_Label.setPreferredSize(new Dimension(240, 9));
        NameValidity_Label.setPreferredSize(new Dimension(240, 9));
//      EmailValidity_Label.setPreferredSize(new Dimension(240, 9));
//      UserNameValidity_Label.setPreferredSize(new Dimension(240, 9));
        PasswordValidity_Label.setPreferredSize(new Dimension(240, 9));


//      CNICValidity_Label.setForeground(Color.red);
        contactValidity_Label.setForeground(Color.red);
        NameValidity_Label.setForeground(Color.red);
        AddressValidity_Label.setForeground(Color.red);
//      UserNameValidity_Label.setForeground(Color.red);
        PasswordValidity_Label.setForeground(Color.red);
//

        frame.add(Name_Label, new AbsoluteConstraints(10, 5));
        frame.add(Name_TextField, new AbsoluteConstraints(195, 5));
```

```
    frame.add(NameValidity_Label, new AbsoluteConstraints(195, 30));


    frame.add(Address_Label, new AbsoluteConstraints(10, 42));
    frame.add(Address_TextField, new AbsoluteConstraints(195, 42));
    frame.add(AddressValidity_Label, new AbsoluteConstraints(195, 66));

    frame.add(Contact_Label, new AbsoluteConstraints(10, 77));
    frame.add(Contact_TextField, new AbsoluteConstraints(195, 77));
    frame.add(contactValidity_Label, new AbsoluteConstraints(195, 102));




    frame.add(Add_Button, new AbsoluteConstraints(100, 225, 100, 22));
    frame.add(Cancel_Button, new AbsoluteConstraints(250, 225, 100, 22));

    Add_Button.addActionListener(new Customer_Add_ActionListener());

    Cancel_Button.addActionListener(new Customer_Add_ActionListener());
}

 private class Customer_Add_ActionListener implements ActionListener {
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        switch (e.getActionCommand()) {
            case "Add": {
//                String cnic = CNIC_TextField.getText().trim();
                String name = Name_TextField.getText().trim();
                String contact = Contact_TextField.getText().trim();
                String address = Address_TextField.getText().trim();



//                if (Customer.isCNICValid(cnic)) {
//                    Customer customer = Customer.SearchByCNIC(cnic);
//                    if (customer == null) {
//                    if (customer == null) {
//                        if (Customer.isNameValid(name)) {
//                            if (Customer.isContactNoValid(contact)) {
                                new Customer(0, 0, name, contact,address).Add(); // ID is Auto
                                Parent_JFrame.getMainFrame().getContentPane().removeAll();
                                Customer_Details cd = new Customer_Details();
                                Parent_JFrame.getMainFrame().add(cd.getMainPanel());
```

```
                        Parent_JFrame.getMainFrame().getContentPane().revalidate();
                        Parent_JFrame.getMainFrame().setEnabled(true);
                        JOptionPane.showMessageDialog(null, "Customer added successfully
!");
                        frame.dispose();
//                      } else {
//                         JOptionPane.showMessageDialog(null, "Invalid contact no. !");
//                      }
//                    } else {
//                      JOptionPane.showMessageDialog(null, "Invalid Name !");
//                    }
//                  } else {
//                    JOptionPane.showMessageDialog(null, "This CNIC is already registered !");
//                  }
//                } else {
//                  JOptionPane.showMessageDialog(null, "Invalid CNIC");
//                }
                  break;
              }
              case "Cancel": {
                  Parent_JFrame.getMainFrame().setEnabled(true);
```

```
                frame.dispose();
                break;
            }
        }
    }


  }
}
```

Customer_Details.java

```
package GUI;


import BackendCode.Booking;
import BackendCode.Customer;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
```

```
import javax.swing.table.DefaultTableCellRenderer;

import org.netbeans.lib.awtextra.AbsoluteConstraints;

import org.netbeans.lib.awtextra.AbsoluteLayout;


public class Customer_Details implements ActionListener {


    private JTextField SearchID_TextField;

    private JButton SearchID_Button, SearchName_Button, Update_Button, Add_Button,
Remove_Button, Back_Button, Logout_Button, ClearBill_Button;

    private JScrollPane jScrollPane1;

    private JTable jTable1;

    private JTextField SearchName_TextField;

    static DefaultTableModel tablemodel;

    private JPanel MainPanel;


    public Customer_Details() {

        MainPanel = new JPanel();

        Parent_JFrame.getMainFrame().setTitle("Customer Details - Lease-A-Item Management
System");

        MainPanel.setLayout(new AbsoluteLayout());

        MainPanel.setMinimumSize(new Dimension(1366, 730));
```

```
SearchID_Button = new JButton("Search ID");

Update_Button = new JButton("Update");

Add_Button = new JButton("Add");

Remove_Button = new JButton("Remove");

Back_Button = new JButton("Back");

Logout_Button = new JButton("Logout");

SearchName_Button = new JButton("Search Name");

ClearBill_Button = new JButton("Clear Bill");

SearchID_TextField = new JTextField();

SearchName_TextField = new JTextField();

jScrollPane1 = new JScrollPane();

jTable1 = new JTable();


String[] columns = {"Sr#", "ID", "Name", "Contact Number","Address", "Item Rented",
"Bill"};

tablemodel = new DefaultTableModel(columns, 0) {


    @Override

    public boolean isCellEditable(int row, int column) {

        //all cells false

        return false;
```

```
        }
    };

    jTable1 = new JTable(tablemodel);

    jTable1.setSize(new Dimension(1330, 550));
    jScrollPane1 = new JScrollPane();
    jScrollPane1.setViewportView(jTable1);
    jTable1.setFillsViewportHeight(true);// makes the size of table equal to that of scroll pane to fill the table in the scrollpane
    ArrayList<Customer> Customer_objects = Customer.View();
    for (int i = 0; i < Customer_objects.size(); i++) {

        int ID = Customer_objects.get(i).getID();
        String Address = Customer_objects.get(i).getAddress();
        String Name = Customer_objects.get(i).getName();
        String ContactNo = Customer_objects.get(i).getContact_No();
        int Bill = Customer_objects.get(i).getBill();

        // getting booked Items for customer
        ArrayList<Booking> bookings = Booking.SearchByCustomerID(ID);
```

```
            String bookedItems = "";
            if (!bookings.isEmpty()) {
                for (int j = 0; j < bookings.size(); j++) {
                    if (bookings.get(j).getReturnTime() == 0) {
                        bookedItems += bookings.get(j).getItem().getID() + ": " +
bookings.get(j).getItem().getID() + "\n";
                    } else {
                        bookedItems = "No Items Booked !";
                    }
                }
            } else {
                bookedItems = "No Items Booked !";
            }
            String[] one_s_Record = {(i + 1) + "", "" + ID, Address, Name, ContactNo, bookedItems,
Bill + ""};
            tablemodel.addRow(one_s_Record);
        }

        // center aligning the text in all the columns
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(JLabel.CENTER);
        jTable1.getColumnModel().getColumn(0).setCellRenderer(centerRenderer);
```

```
        jTable1.getColumnModel().getColumn(1).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(2).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(3).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(4).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(5).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(6).setCellRenderer(centerRenderer);
//      jTable1.getColumnModel().getColumn(7).setCellRenderer(centerRenderer);
//      jTable1.getColumnModel().getColumn(8).setCellRenderer(centerRenderer);
//      jTable1.getColumnModel().getColumn(9).setCellRenderer(centerRenderer);

        // adjusting size of each column
        jTable1.getColumnModel().getColumn(0).setPreferredWidth(70);
        jTable1.getColumnModel().getColumn(1).setPreferredWidth(150);
        jTable1.getColumnModel().getColumn(2).setPreferredWidth(170);
        jTable1.getColumnModel().getColumn(3).setPreferredWidth(110);
        jTable1.getColumnModel().getColumn(4).setPreferredWidth(180);
        jTable1.getColumnModel().getColumn(5).setPreferredWidth(140);
        jTable1.getColumnModel().getColumn(6).setPreferredWidth(100);
//      jTable1.getColumnModel().getColumn(7).setPreferredWidth(130);
//      jTable1.getColumnModel().getColumn(8).setPreferredWidth(110);
//      jTable1.getColumnModel().getColumn(9).setPreferredWidth(110);
```

```
//      jScrollPane1.setViewportView(jTable1);
        MainPanel.add(SearchID_Button, new AbsoluteConstraints(390, 10, 130, 22));
        MainPanel.add(SearchID_TextField, new AbsoluteConstraints(525, 10, 240, 22));
        MainPanel.add(SearchName_Button, new AbsoluteConstraints(10, 10, 130, 22));
        MainPanel.add(SearchName_TextField, new AbsoluteConstraints(145, 10, 240, 22));
        MainPanel.add(jScrollPane1, new AbsoluteConstraints(10, 50, 1330, 550));
        MainPanel.add(Update_Button, new AbsoluteConstraints(579, 625, 130, 22));
        MainPanel.add(Add_Button, new AbsoluteConstraints(420, 625, 130, 22));
        MainPanel.add(Remove_Button, new AbsoluteConstraints(735, 625, 130, 22));
        MainPanel.add(Back_Button, new AbsoluteConstraints(1106, 625, 100, 22));
        MainPanel.add(Logout_Button, new AbsoluteConstraints(1236, 625, 100, 22));
        MainPanel.add(ClearBill_Button, new AbsoluteConstraints(10, 625, 200, 22));

        SearchID_Button.addActionListener(this);
        SearchName_Button.addActionListener(this);
        Remove_Button.addActionListener(this);
        Add_Button.addActionListener(this);
        Update_Button.addActionListener(this);
        Back_Button.addActionListener(this);
        Logout_Button.addActionListener(this);
```

```
        ClearBill_Button.addActionListener(this);
    }

//    public static void main(String args[]) {
//        new Customer_Details().setVisible(true);
//
//    }
    public JPanel getMainPanel() {
        return MainPanel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        switch (e.getActionCommand()) {
            case "Search ID": {
                String id = SearchID_TextField.getText().trim();
                if (!id.isEmpty()) {
                    if (Customer.isIDvalid(id)) {
                        Customer co = Customer.SearchByID(Integer.parseInt(id));
                        if (co != null) {
                            JOptionPane.showMessageDialog(null, co.toString());
```

```
                SearchID_TextField.setText("");
            } else {
                JOptionPane.showMessageDialog(null, "Required person not found");
                SearchID_TextField.setText("");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Invalid ID !");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Please Enter ID first !");
    }
}
break;
case "Search Name": {
    String name = SearchName_TextField.getText().trim();
    if (!name.isEmpty()) {
        if (Customer.isNameValid(name)) {
            ArrayList<Customer> customerArrayList = Customer.SearchByName(name);
            String record = "";
            for (int i = 0; i < customerArrayList.size(); i++) {
                record += customerArrayList.get(i).toString() + "\n";
```

```
            }
            if (!customerArrayList.isEmpty()) {
                JOptionPane.showMessageDialog(null, record);
                SearchName_TextField.setText("");
            } else {
                JOptionPane.showMessageDialog(null, "Required person not found");
                SearchName_TextField.setText("");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Invalid Name !");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Please Enter Name first !");
    }
}
break;
case "Add": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    Customer_Add aco = new Customer_Add();
    aco.frame.setVisible(true);
}
```

```
        break;
        case "Remove": {
            Parent_JFrame.getMainFrame().setEnabled(false);
            new Customer_Remove().frame.setVisible(true);
        }
        break;
        case "Update": {
            Parent_JFrame.getMainFrame().setEnabled(false);
            new Customer_Update().frame.setVisible(true);
        }
        break;
        case "Back": {
            Parent_JFrame.getMainFrame().setTitle("Lease-A-Item Management System
[REBORN]");
            MainMenu mm = new MainMenu();
            Parent_JFrame.getMainFrame().getContentPane().removeAll();
            Parent_JFrame.getMainFrame().add(mm.getMainPanel());
            Parent_JFrame.getMainFrame().getContentPane().revalidate();
        }
        break;
        case "Logout": {
```

```
            Parent_JFrame.getMainFrame().dispose();
            Runner r = new Runner();
            JFrame frame = Runner.getFrame();
            Login login = new Login();
            JPanel panel = login.getMainPanel();
            frame.add(panel);
            frame.setVisible(true);
        }
        break;
        case "Clear Bill": {
            ArrayList<Customer> View = Customer.View();//Creating an arrayList that contains Objects of all Customers
            if (!View.isEmpty()) {
                ArrayList<String> IDsArray = new ArrayList<>(0);
                for (int i = 0; i < View.size(); i++) { // getting IDs of all the Customers with Bill > 0
                    if (View.get(i).getBill() != 0) {
                        IDsArray.add(View.get(i).getID() + "");
                    }
                }
                Object showInputDialog = JOptionPane.showInputDialog(null, "Select ID for Customer whose bill you want to clear.", "Clear Bill",
                        JOptionPane.PLAIN_MESSAGE, null, IDsArray.toArray(), null);
```

```
            System.out.println(showInputDialog);


            if (showInputDialog != null) {
                Customer customer =
Customer.SearchByID((Integer.parseInt(showInputDialog + "")));


                int showConfirmDialog = JOptionPane.showConfirmDialog(null, "You are
about to clear the balance for the following Customer\n"
                        + customer + "\nAre you sure you want to continue ?", "Clear Bill
Confirmation",
                        JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE,
null);
                if (showConfirmDialog == 0) {
                    customer.setBill(0);
                    customer.Update();
                    Parent_JFrame.getMainFrame().getContentPane().removeAll();
                    Customer_Details cd = new Customer_Details();
                    Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                    Parent_JFrame.getMainFrame().getContentPane().revalidate();
                    JOptionPane.showMessageDialog(null, "Bill Successfully Cleared !");
                }
            }
        } else {
```

```
                JOptionPane.showMessageDialog(null, "No Customer Currently Registered !");
            }
        }
        break;
    }
  }
}
```

Customer_Remove.java

```
package GUI;


import BackendCode.Booking;
import BackendCode.Customer;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.*;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;
```

```
public class Customer_Remove {


    JButton Remove_Button, Cancel_Button;
    JLabel ID_Label, IDValidity_Label;
    JTextField ID_TextField;
    JFrame frame = new JFrame();


    public Customer_Remove() {
        frame.setTitle("Remove Customer");
        frame.setLayout(new AbsoluteLayout());
        frame.setSize(new Dimension(450, 290));
        frame.setResizable(false);
        frame.setLocationRelativeTo(Parent_JFrame.getMainFrame());
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                Parent_JFrame.getMainFrame().setEnabled(true);
                frame.dispose();
            }
        });
```

```
        Remove_Button = new JButton("Remove");
        Cancel_Button = new JButton("Cancel");

        ID_Label = new JLabel("Enter ID (without dashes)");
        IDValidity_Label = new JLabel();
        ID_TextField = new JTextField();

        ID_TextField.setPreferredSize(new Dimension(240, 22));

        ID_Label.setPreferredSize(new Dimension(175, 22));
        IDValidity_Label.setPreferredSize(new Dimension(240, 9));
        IDValidity_Label.setForeground(Color.red);
        frame.add(ID_Label, new AbsoluteConstraints(10, 5));
        frame.add(ID_TextField, new AbsoluteConstraints(195, 5));
//        IDValidity_Label.setText("Invalid ID !");
        frame.add(IDValidity_Label, new AbsoluteConstraints(195, 30));
        frame.add(Remove_Button, new AbsoluteConstraints(100, 225, 100, 22));
        frame.add(Cancel_Button, new AbsoluteConstraints(250, 225, 100, 22));

        Remove_Button.addActionListener(new Customer_Remove_ActionListener());
```

```
        Cancel_Button.addActionListener(new Customer_Remove_ActionListener());
    }

    private class Customer_Remove_ActionListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            switch (e.getActionCommand()) {
                case "Remove": {
                    String id = ID_TextField.getText().trim();
                    if (Customer.isIDvalid(id)) {
                        Customer customer = Customer.SearchByID(Integer.parseInt(id));
                        if (customer != null) {
                            int showConfirmDialog = JOptionPane.showConfirmDialog(frame, "You are
about to remove the following Customer.\n"
                                    + customer.toString() + " \nAll the data including Booked Items and
Balance for this Customer will also be deleted  !"
                                    + "\n Are you sure you want to continue ??", "Remove Customer",
JOptionPane.OK_CANCEL_OPTION);
                            if (showConfirmDialog == 0) {
                                // Deleting all the booking records of customer
```

```
    ArrayList<Booking> bookings = Booking.View();
    for (int i = 0; i < bookings.size(); i++) {
        if (customer.getID() == bookings.get(i).getCustomer().getID()) {
            bookings.get(i).Remove();
        }
    }
    // ** Delete all Items for this Customer **
    customer.Remove();

    System.out.println("Customer deleted !");
    Parent_JFrame.getMainFrame().getContentPane().removeAll();
    Customer_Details cd = new Customer_Details();
    Parent_JFrame.getMainFrame().add(cd.getMainPanel());
    Parent_JFrame.getMainFrame().getContentPane().revalidate();
    JOptionPane.showMessageDialog(null, "Record successfully Removed !");
    Parent_JFrame.getMainFrame().setEnabled(true);
    frame.dispose();
} else {

    frame.setEnabled(true);
}
```

```
                } else {
                    JOptionPane.showMessageDialog(null, "This ID does not exists !");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Enter a valid ID !\n(A valid ID is an integer number greater than 0)");
            }
            break;
        }
        case "Cancel": {
            Parent_JFrame.getMainFrame().setEnabled(true);
            frame.dispose();
            break;
        }
    }
  }
}
```

Customer_Update.java

```
package GUI;


import BackendCode.Customer;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

import org.netbeans.lib.awtextra.AbsoluteConstraints;

import org.netbeans.lib.awtextra.AbsoluteLayout;


public class Customer_Update {


   JButton OK_Button, Cancel_Button;

   JLabel ID_Label, IDValidity_Label;

   JTextField ID_TextField;


   JFrame frame = new JFrame();

   static Customer customer; // this customer object is used in UpdateCustomer_Inner class to obtain the record for entered ID


   public Customer_Update() {
```

```
frame.setTitle("Update Customer");
frame.setLayout(new AbsoluteLayout());
frame.setSize(new Dimension(450, 290));
frame.setResizable(false);
frame.setLocationRelativeTo(Parent_JFrame.getMainFrame());
frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
frame.addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        Parent_JFrame.getMainFrame().setEnabled(true);
        frame.dispose();
    }
});

OK_Button = new JButton("OK");
Cancel_Button = new JButton("Cancel");

ID_Label = new JLabel("Enter ID to be Updated");
IDValidity_Label = new JLabel();
ID_TextField = new JTextField();
```

```
ID_TextField.setPreferredSize(new Dimension(240, 22));


ID_Label.setPreferredSize(new Dimension(175, 22));
IDValidity_Label.setPreferredSize(new Dimension(240, 9));
IDValidity_Label.setForeground(Color.red);
frame.add(ID_Label, new AbsoluteConstraints(10, 5));
frame.add(ID_TextField, new AbsoluteConstraints(195, 5));
frame.add(IDValidity_Label, new AbsoluteConstraints(195, 30));
frame.add(OK_Button, new AbsoluteConstraints(100, 225, 100, 22));
frame.add(Cancel_Button, new AbsoluteConstraints(250, 225, 100, 22));


OK_Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Customer CO = new Customer();
        String ID = ID_TextField.getText().trim();
        if (!ID_TextField.getText().isEmpty()) {
            if (Customer.isIDvalid(ID)) {
                CO.setID(Integer.parseInt(ID));
                customer = Customer.SearchByID(Integer.parseInt(ID)); // the ID of this object
is used in UpdateManage_GUI_B class. that is why it is kept static
```

```
                if (customer != null) {
                    Parent_JFrame.getMainFrame().setEnabled(false);
                    frame.dispose();
                    new UpdateCustomer_Inner().setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "Required ID is not found !");
                }
            } else {
                IDValidity_Label.setText("Invalid ID !");
            }
        } else {
            IDValidity_Label.setText("Enter ID !");
        }
    }
});

Cancel_Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Parent_JFrame.getMainFrame().setEnabled(true);
        frame.dispose();
```

```
        }
    });
  }

  public class UpdateCustomer_Inner extends JFrame {

    JButton Update_Button, Cancel_Button;
    JLabel Name_Label,Address_Label,Contact_Label,Password_Label;
    JLabel  contactValidity_Label, NameValidity_Label, AddressValidity_Label,
PasswordValidity_Label;
    JTextField Name_TextField, Contact_TextField, Address_TextField;

    public UpdateCustomer_Inner() {
      super("Update Customer");
      System.out.println("95");
      setLayout(new AbsoluteLayout());
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
System.out.println("98");
      setSize(new Dimension(450, 290));
      setResizable(false);
      setLocationRelativeTo(this);
```

```
        Update_Button = new JButton("Update");
        Cancel_Button = new JButton("Cancel");
System.out.println("103");

        Name_Label = new JLabel("Enter Name");
        Contact_Label = new JLabel("Enter Contact");
        Address_Label = new JLabel("Enter Address");

        NameValidity_Label = new JLabel();
        AddressValidity_Label = new JLabel();
        contactValidity_Label = new JLabel();
System.out.println("114");
        Name_TextField = new JTextField();
        Contact_TextField = new JTextField();
        Address_TextField = new JTextField();

        Name_TextField.setPreferredSize(new Dimension(240, 22));
        Contact_TextField.setPreferredSize(new Dimension(240, 22));
        Address_TextField.setPreferredSize(new Dimension(240, 22));
System.out.println("122");
        Name_Label.setPreferredSize(new Dimension(175, 22));
```

```
        Contact_Label.setPreferredSize(new Dimension(175, 22));
        NameValidity_Label.setPreferredSize(new Dimension(240, 9));
        contactValidity_Label.setPreferredSize(new Dimension(240, 9));
        AddressValidity_Label.setPreferredSize(new Dimension(240, 9));
        System.out.println("128");

        contactValidity_Label.setForeground(Color.red);
        NameValidity_Label.setForeground(Color.red);
        AddressValidity_Label.setForeground(Color.red);
System.out.println("134");
        add(Name_Label, new AbsoluteConstraints(10, 5));
        add(Name_TextField, new AbsoluteConstraints(195, 5));
        add(NameValidity_Label, new AbsoluteConstraints(195, 30));

        add(Address_Label, new AbsoluteConstraints(10, 42));
        add(Address_TextField, new AbsoluteConstraints(195, 42));
        add(AddressValidity_Label, new AbsoluteConstraints(195, 66));
System.out.println("142");
        add(Contact_Label, new AbsoluteConstraints(10, 77));
        add(Contact_TextField, new AbsoluteConstraints(195, 77));
        add(contactValidity_Label, new AbsoluteConstraints(195, 102));
```

```
        add(Update_Button, new AbsoluteConstraints(100, 225, 100, 22));
        add(Cancel_Button, new AbsoluteConstraints(250, 225, 100, 22));
System.out.println("150");
        Update_Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                String name = Name_TextField.getText().trim();
                String contact = Contact_TextField.getText().trim();
                String address = Address_TextField.getText().trim();
                if (!name.isEmpty()) {
                    if (Customer.isNameValid(name)) {
                        System.out.println("valid Customer name !");
                    } else {
                        name = null;
                        NameValidity_Label.setText("Invalid Name !");
                    }
                } else {
                    name = null;
                    NameValidity_Label.setText("Enter Name !");
```

```
            }
            if (contact.isEmpty()) {
                contact = null;
                contactValidity_Label.setText("Enter Contact Number !");
            }

            if ( name != null && contact != null) {
                customer = new Customer(customer.getBill(), customer.getID(), name,
contact,address);
                System.out.println(customer.toString());
                customer.Update();
                Parent_JFrame.getMainFrame().getContentPane().removeAll();
                Customer_Details cd = new Customer_Details();
                Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                Parent_JFrame.getMainFrame().getContentPane().revalidate();
                JOptionPane.showMessageDialog(null, "Record Successfully Updated !");
                Parent_JFrame.getMainFrame().setEnabled(true);
                dispose();
            }
        }
    });
```

```
        Cancel_Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Parent_JFrame.getMainFrame().setEnabled(true);
                dispose();
            }
        });
    }


}
```

Item_Add.java

```
package GUI;


import BackendCode.Item;
import java.awt.*;
import java.awt.event.*;
```

```
import java.util.Date;
import javax.swing.*;


/**
 *
 * @author @AbdullahShahid01
 */
public class Item_Add extends JFrame {


//    int ID;
//    String Name, Model, Type,Specifications;
//    int length,breadth, height,weight,RentPerHour;
//
    JButton Add_Button, Cancel_Button;
    JLabel Name_Label,Model_Label,Type_Label, Specifications_Label,
RentPerHour_Label,Length_Label,Height_Label,Breadth_Label,Weight_Label,
            NameValidity_Label, ModelValidity_Label, TypeValidity_Label,
SpecificationValidity_Label,RentPerHourValidity_Label;
    JTextField Name_TextField, Model_TextField,Type_TextField,Specifications_TextField,
RentPerHour_TextField,Length_TextField,Breadth_TextField,Height_TextField,Weight_TextFi
eld;
    JComboBox<String> Type_ComboBox, Model_ComboBox;
//    JSpinner SeatingCapacity_Spinner;
```

```
    public Item_Add() {
        super("Add Item");
        System.out.println("Item Add worked");
        setLayout(new FlowLayout());
        setSize(new Dimension(450, 475));
        setResizable(false);
        setLocationRelativeTo(this);
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                Parent_JFrame.getMainFrame().setEnabled(true);
                dispose();
            }
        });
System.out.println("Add 43");
        Add_Button = new JButton("Add");
        Cancel_Button = new JButton("Cancel");

        Name_Label = new JLabel("Name");
```

```
Model_Label = new JLabel("Model");

Type_Label = new JLabel("Item type");

Specifications_Label = new JLabel("Specification");

RentPerHour_Label = new JLabel("Rent Per Hour (in PKR)");

Length_Label = new JLabel("Length ");

Breadth_Label = new JLabel("Breadth");

Height_Label = new JLabel("Height");

Weight_Label = new JLabel("Weight");



NameValidity_Label = new JLabel();

ModelValidity_Label = new JLabel();

TypeValidity_Label = new JLabel();

SpecificationValidity_Label= new JLabel();

RentPerHourValidity_Label = new JLabel();


Name_TextField = new JTextField();

Model_TextField = new JTextField();

Type_TextField = new JTextField();

Specifications_TextField = new JTextField();

RentPerHour_TextField = new JTextField();
```

```
Length_TextField = new JTextField();

Breadth_TextField = new JTextField();

Height_TextField = new JTextField();

Weight_TextField = new JTextField();




Name_TextField.setPreferredSize(new Dimension(240, 22));

Model_TextField.setPreferredSize(new Dimension(240, 22));

Type_TextField.setPreferredSize(new Dimension(240, 22));

Specifications_TextField.setPreferredSize(new Dimension(240, 22));

RentPerHour_TextField.setPreferredSize(new Dimension(240, 22));


Length_TextField.setPreferredSize(new Dimension(240, 22));

Breadth_TextField.setPreferredSize(new Dimension(240, 22));

Height_TextField.setPreferredSize(new Dimension(240, 22));

Weight_TextField.setPreferredSize(new Dimension(240, 22));
```

```
Name_Label.setPreferredSize(new Dimension(175, 22));
Model_Label.setPreferredSize(new Dimension(175, 22));
Type_Label.setPreferredSize(new Dimension(175, 22));
Specifications_Label.setPreferredSize(new Dimension(175, 22));
RentPerHour_Label.setPreferredSize(new Dimension(175, 22));
Length_Label.setPreferredSize(new Dimension(175, 22));
Height_Label.setPreferredSize(new Dimension(175, 22));
Breadth_Label.setPreferredSize(new Dimension(175, 22));
Weight_Label.setPreferredSize(new Dimension(175, 22));

NameValidity_Label.setPreferredSize(new Dimension(415, 9));
ModelValidity_Label.setPreferredSize(new Dimension(415, 9));
TypeValidity_Label.setPreferredSize(new Dimension(415, 9));
SpecificationValidity_Label.setPreferredSize(new Dimension(415, 9));
RentPerHourValidity_Label.setPreferredSize(new Dimension(415, 9));

NameValidity_Label.setForeground(Color.red);
ModelValidity_Label.setForeground(Color.red);
TypeValidity_Label.setForeground(Color.red);
```

```
SpecificationValidity_Label.setForeground(Color.red);

RentPerHourValidity_Label.setForeground(Color.red);


add(Name_Label);

add(Name_TextField);

add(NameValidity_Label);


add(Model_Label);

add(Model_TextField);

add(ModelValidity_Label);


add(Type_Label);

add(Type_TextField);

add(TypeValidity_Label);


add(Specifications_Label);

add(Specifications_TextField);

add(SpecificationValidity_Label);



add(RentPerHour_Label);
```

```
add(RentPerHour_TextField);

add(RentPerHourValidity_Label);


add(Length_Label);

add(Length_TextField);


add(Height_Label);

add(Height_TextField);


add(Breadth_Label);

add(Breadth_TextField);


add(Weight_Label);

add(Weight_TextField);


add(Add_Button);

add(Cancel_Button);


System.out.println("Add 127");

Add_Button.addActionListener(new ActionListener() {
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("Add button worked");
            String name = Name_TextField.getText().trim(),
                    model = Model_TextField.getText().trim(),
                    type = Type_TextField.getText().trim(),
                    specification = Specifications_TextField.getText().trim(),
                    rentPerHour = RentPerHour_TextField.getText().trim();

            String length = Length_TextField.getText().trim();
            String breadth = Breadth_TextField.getText().trim();
            String height = Height_TextField.getText().trim();
            String weight = Weight_TextField.getText().trim();
            System.out.println("Add button value read complete");

            if (!name.isEmpty()) {
                if (Item.isNameValid(Name_TextField.getText().trim())) {
                    NameValidity_Label.setText("");
//                      name = Name_TextField.getText().trim();
                } else {
```

```
            name = null;
            NameValidity_Label.setText("Invalid  Item Name !");
        }
    } else {
        name = null;
        NameValidity_Label.setText("Enter Item Name !");
    }

    if (!model.isEmpty()) {
        if (Item.isNameValid(model)) {
            ModelValidity_Label.setText("");

        } else {
            model = null;
            ModelValidity_Label.setText("Invalid Model Name !");
        }
    } else {
        model = null;
        ModelValidity_Label.setText("Enter Model Name !");
    }
```

```
if (type.isEmpty()) {


    type = null;

    TypeValidity_Label.setText("Enter Type!");

}


if (!specification.isEmpty()) {

    if (Item.isNameValid(specification)) {

        SpecificationValidity_Label.setText("");


    } else {

        model = null;

        SpecificationValidity_Label.setText("Invalid specification label !");

    }

} else {

    model = null;

    SpecificationValidity_Label.setText("Enter specification label !");

}


 if (!rentPerHour.isEmpty()) {

    if (Item.isNameValid(rentPerHour)) {
```

```
            RentPerHourValidity_Label.setText("");


        } else {
            rentPerHour = null;
            RentPerHourValidity_Label.setText("Invalid rentPerHour label !");
        }
    } else {
        rentPerHour = null;
        RentPerHourValidity_Label.setText("Enter rentPerHour label !");
    }




    try {
        if (name != null && model != null && type != null && rentPerHour != null) {
            Item item = Item.SearchByRegNo(model);



                if (item == null) {

                    //Item(id, Maker, Name, Colour, Type, SeatingCapacity, Model, Condition,
RegNo, RentPerHour, itemOwner)
```

```
                        // id is auto
//                          this.ID = ID;
//      this.Name = Name;
//      this.Model = Model;
//      this.Type = Type;
//      this.Specifications = Specifications;
//      this.length = length;
//      this.breadth = breadth;
//      this.height = height;
//      this.weight = weight;
//      this.RentPerHour = RentPerHour;
//    public Item(int ID, String Name, String Model, String Type, String Specifications, int
length, int breadth, int height, int weight, int RentPerHour) {
                        System.out.println("Constructor reached");
                        item = new Item(0,name,
model,type,specification,length,breadth,height,weight, rentPerHour);
                        item.Add();

                        Parent_JFrame.getMainFrame().getContentPane().removeAll();
                        Item_Details cd = new Item_Details();
                        Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                        Parent_JFrame.getMainFrame().getContentPane().revalidate();
```

```
                    JOptionPane.showMessageDialog(null, "Record Successfully Added !");
                    Parent_JFrame.getMainFrame().setEnabled(true);
                    dispose();
                } else {
                    JOptionPane.showMessageDialog(null, "This item Registeration no is already registered !");
                }

            }
        } catch (HeadlessException | NumberFormatException ex) {
            System.out.println(ex);
        }
    }
}
);
Cancel_Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Parent_JFrame.getMainFrame().setEnabled(true);
        dispose();
    }
```

```
        });
    }
}
```

Item_Details.java

```
package GUI;


import BackendCode.Booking;
import BackendCode.Item;
//import BackendCode.ItemOwner;
import java.awt.Dimension;
import javax.swing.table.DefaultTableModel;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableCellRenderer;

public class Item_Details {

    private static DefaultTableModel tablemodel; // it is made static so that it can be accessed in add GUI class to update the Jtable when a new record is added

    private static JButton SearchName_Button, SearchRegNo_Button, Add_Button,
            Update_Button, Remove_Button, BackButton, LogoutButton;
    private static JTextField SearchName_TextField, SearchRegNo_TextField;
    private static JScrollPane jScrollPane1;
    private static JTable jTable1;
    private JPanel MainPanel;

    /**
     * @return the tablemodel
```

```
     */
    public static DefaultTableModel getTablemodel() {
        return tablemodel;
    }

    public JPanel getMainPanel() {
        return MainPanel;
    }

    public Item_Details() {
        MainPanel = new JPanel();
        Parent_JFrame.getMainFrame().setTitle("Item Details - Lease-A-Item Management System");
        MainPanel.setLayout(new AbsoluteLayout());
        MainPanel.setMinimumSize(new Dimension(1366, 730));
//
//        SearchRegNo_Button = new JButton("Search Reg_No");
//        SearchRegNo_TextField = new JTextField();

        SearchName_Button = new JButton("Search Name");
        SearchName_TextField = new JTextField();
```

```
        Add_Button = new JButton("Add");
        Update_Button = new JButton("Update");
        Remove_Button = new JButton("Remove");
        BackButton = new JButton("Back");
        LogoutButton = new JButton("Logout");



        jScrollPane1 = new JScrollPane();
        jTable1 = new JTable();

//        this.Name = Name;
//        this.Model = Model;
//        this.Type = Type;
//        this.Specifications = Specifications;
//        this.length = length;
//        this.breadth = breadth;
//        this.height = height;
//        this.weight = weight;
//        this.RentPerHour = RentPerHour;
```

```
//"Sr#", "ID", "Name", "Model", "Type", "Specifications", "Length", "Breadth","Height",
"Weight", "Rent/hr"
        String[] columns =  {"Sr#", "ID", "Name", "Model", "Type", "Specifications", "length",
"breadth", "height",
            "weight", "Rent/hour"};
        tablemodel = new DefaultTableModel(columns, 0) {


            @Override
            public boolean isCellEditable(int row, int column) {
                //all cells false
                return false;
            }
        };


        jTable1 = new JTable(getTablemodel());
        jTable1.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
//        jTable1.setPreferredScrollableViewportSize(new Dimension(2000, 550));
        jScrollPane1 = new JScrollPane();
        jScrollPane1.setViewportView(jTable1);
        jTable1.setFillsViewportHeight(true);// makes the size of table equal to that of scroll
pane to fill the table in the scrollpane
        ArrayList<Item> Item_objects = Item.View();
```

```
      for (int i = 0; i < Item_objects.size(); i++) {
//ID, Maker, Name, Colour, Type, SeatingCapacity, Model, Condition, RegNo,
//RentPerHour, IsRented RentDate, itemOwner customer
//    int ID;
//    String Name, Model, Type,Specifications;
//    int length,breadth, height,weight,RentPerHour;
//
            int ID = Item_objects.get(i).getID();
            String Name = Item_objects.get(i).getName();
            String model = Item_objects.get(i).getModel();
            String type = Item_objects.get(i).getType();
            String specification = Item_objects.get(i).getSpecifications();
            String length = Item_objects.get(i).getLength();
            String breadth = Item_objects.get(i).getBreadth();
            String height = Item_objects.get(i).getHeight();
            String weight = Item_objects.get(i).getWeight();
            String rentPerHour = Item_objects.get(i).getRentPerHour();

            String customerID = "";
            String customerName = "";
            String[] one_s_Record = {
```

```
        ((i + 1) + ""),
        "" + ID,
        Name,
        model,
        type,
        specification+"",
        length+"",
        breadth+"",
        height+"",
        weight+"",
        rentPerHour };
    tablemodel.addRow(one_s_Record);
}

// center aligning the text in all the columns
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment(JLabel.CENTER);
jTable1.getColumnModel().getColumn(0).setCellRenderer(centerRenderer);
jTable1.getColumnModel().getColumn(1).setCellRenderer(centerRenderer);
jTable1.getColumnModel().getColumn(2).setCellRenderer(centerRenderer);
jTable1.getColumnModel().getColumn(3).setCellRenderer(centerRenderer);
```

```
        jTable1.getColumnModel().getColumn(4).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(5).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(6).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(7).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(8).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(9).setCellRenderer(centerRenderer);
        jTable1.getColumnModel().getColumn(10).setCellRenderer(centerRenderer);
//        jTable1.getColumnModel().getColumn(11).setCellRenderer(centerRenderer);


        // adjusting size of each column
        jTable1.getColumnModel().getColumn(0).setMinWidth(20);
        jTable1.getColumnModel().getColumn(1).setMinWidth(20);
        jTable1.getColumnModel().getColumn(2).setMinWidth(170);
        jTable1.getColumnModel().getColumn(3).setMinWidth(170);
        jTable1.getColumnModel().getColumn(4).setMinWidth(140);
        jTable1.getColumnModel().getColumn(5).setMinWidth(150);
        jTable1.getColumnModel().getColumn(6).setMinWidth(20);
        jTable1.getColumnModel().getColumn(7).setMinWidth(20);
        jTable1.getColumnModel().getColumn(8).setMinWidth(20);
        jTable1.getColumnModel().getColumn(9).setMinWidth(20);
```

```
        jTable1.getColumnModel().getColumn(10).setMinWidth(30);
//      jTable1.getColumnModel().getColumn(11).setMinWidth(150);



        jTable1.getTableHeader().setReorderingAllowed(false);

//      MainPanel.add(SearchRegNo_Button, new AbsoluteConstraints(10, 15, 130, 22));
//      MainPanel.add(SearchRegNo_TextField, new AbsoluteConstraints(145, 15, 240, 22));
        MainPanel.add(SearchName_Button, new AbsoluteConstraints(390, 15, 130, 22));
        MainPanel.add(SearchName_TextField, new AbsoluteConstraints(525, 15, 240, 22));
        MainPanel.add(jScrollPane1, new AbsoluteConstraints(10, 60, 1330, 550));
        MainPanel.add(Remove_Button, new AbsoluteConstraints(785, 625, 130, 22));
        MainPanel.add(Add_Button, new AbsoluteConstraints(450, 625, 130, 22));
        MainPanel.add(Update_Button, new AbsoluteConstraints(620, 625, 130, 22));
        MainPanel.add(BackButton, new AbsoluteConstraints(1106, 625, 100, 22));
        MainPanel.add(LogoutButton, new AbsoluteConstraints(1236, 625, 100, 22));

        SearchName_Button.addActionListener(new Item_Details_ActionListener());
//      SearchRegNo_Button.addActionListener(new Item_Details_ActionListener());
        Add_Button.addActionListener(new Item_Details_ActionListener());
        Update_Button.addActionListener(new Item_Details_ActionListener());
```

```
    Remove_Button.addActionListener(new Item_Details_ActionListener());
    BackButton.addActionListener(new Item_Details_ActionListener());
    LogoutButton.addActionListener(new Item_Details_ActionListener());

}

private class Item_Details_ActionListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {

        switch (e.getActionCommand()) {

            case "Search Name": {
                String name = SearchName_TextField.getText().trim();
                if (!name.isEmpty()) {
                    if (Item.isNameValid(name)) {

                        ArrayList<Item> item = Item.SearchByName(name);

                        if (!item.isEmpty()) {
```

```
                JOptionPane.showMessageDialog(null, item.toString());
                SearchName_TextField.setText("");
            } else {
                JOptionPane.showMessageDialog(null, "Required item not found");
                SearchName_TextField.setText("");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Invalid Name !");
            SearchName_TextField.setText("");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Please Enter item Name first !");
    }

}
break;
case "Add": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    Item_Add ac = new Item_Add();
    ac.setVisible(true);
}
```

```
            break;
            case "Update": {
                Parent_JFrame.getMainFrame().setEnabled(false);
                Item_Update ac = new Item_Update();
                ac.setVisible(true);
            }
            break;
            case "Remove": {
                Parent_JFrame.getMainFrame().setEnabled(false);
                Item_Remove ac = new Item_Remove();
                ac.setVisible(true);
            }
            break;
            case "Back": {
                Parent_JFrame.getMainFrame().setTitle("Rent-A-Item Management System
[REBORN]");
                MainMenu mm = new MainMenu();
                Parent_JFrame.getMainFrame().getContentPane().removeAll();
                Parent_JFrame.getMainFrame().add(mm.getMainPanel());
                Parent_JFrame.getMainFrame().getContentPane().revalidate();
            }
```

```
            break;
            case "Logout": {
                Parent_JFrame.getMainFrame().dispose();
                Runner r = new Runner();
                JFrame frame = r.getFrame();
                Login login = new Login();
                JPanel panel = login.getMainPanel();
                frame.add(panel);
                frame.setVisible(true);
            }
            break;
            case "Book": {
                if (!Booking.getUnbookedItems().isEmpty()) {
                    Parent_JFrame.getMainFrame().setEnabled(false);
                    Booking_BookItem ac = new Booking_BookItem();
                    ac.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "No UnBooked Items are available !");
                }
            }
            break;
```

```
            case "Unbook": {
                if (!Booking.getBookedItems().isEmpty()) {
                    Parent_JFrame.getMainFrame().setEnabled(false);
                    Booking_UnBookItem ac = new Booking_UnBookItem();
                    ac.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "No Booked Items found !");
                }
            }
            break;
        }
    }
  }
}
```

Item_Remove.java

```
package GUI;


import BackendCode.Item;
import BackendCode.Item;
```

```
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.*;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;


/**
 *
 * @author @AbdullahShahid01
 */
public class Item_Remove extends JFrame {

    JButton Remove_Button, Cancel_Button;
    JLabel ItemID_Label, ItemIDValidity_Label;
    JTextField ItemID_TextField;

    private Item item;

    public Item_Remove() {
```

```
super("Remove Item");
setLayout(new FlowLayout());
setSize(new Dimension(300, 140));
setResizable(false);
setLocationRelativeTo(this);
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        Parent_JFrame.getMainFrame().setEnabled(true);
        dispose();
    }
});

Remove_Button = new JButton("Remove");
Cancel_Button = new JButton("Cancel");

ItemID_Label = new JLabel("Enter Item ID to be removed");
ItemIDValidity_Label = new JLabel();
ItemID_TextField = new JTextField();

ItemID_TextField.setPreferredSize(new Dimension(240, 22));
```

```
//      ItemID_Label.setPreferredSize(new Dimension(175, 22));
        ItemIDValidity_Label.setPreferredSize(new Dimension(415, 9));
        Remove_Button.setPreferredSize(new Dimension(100, 22));
        Cancel_Button.setPreferredSize(new Dimension(100, 22));

        ItemIDValidity_Label.setForeground(Color.red);

        add(ItemID_Label);
        add(ItemID_TextField);
        add(ItemIDValidity_Label);

        add(Remove_Button);
        add(Cancel_Button);

        Remove_Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String itemID = ItemID_TextField.getText().trim();
                if (!itemID.isEmpty()) {
                    try {
                        if (Integer.parseInt(itemID) > 0) {
```

```
                ItemIDValidity_Label.setText("");
//                  if (itemID != null) {
                Item item = Item.SearchByID(Integer.parseInt(itemID));
                if (item != null) {
                    int showConfirmDialog = JOptionPane.showConfirmDialog(null, "You are about to remove this item \n "
                            + item.toString() + "\n Are you sure you want to continue ??", "Confirmation",
                            JOptionPane.OK_CANCEL_OPTION);
                    if (showConfirmDialog == 0) {
                        item.Remove();
                        Parent_JFrame.getMainFrame().getContentPane().removeAll();
                        Item_Details cd = new Item_Details();
                        Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                        Parent_JFrame.getMainFrame().getContentPane().revalidate();

                        Parent_JFrame.getMainFrame().setEnabled(true);
                        dispose();
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Item ID not found !");
                }
```

```
                } else {
                    itemID = null;
                    ItemIDValidity_Label.setText("ID cannot be '0' or negative !");
                }
            } catch (NumberFormatException ex) {
                itemID = null;
                ItemIDValidity_Label.setText("Invalid ID !");
            }
        } else {
            itemID = null;
            ItemIDValidity_Label.setText("Enter Item ID !");
        }
        /*ID, Maker, Name, Colour, Type, SeatingCapacity, Model, Condition, RegNo,
RentPerHour, IsRented RentDate, ItemItem, customer*/


    }
}
);
Cancel_Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

```
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        }
    });
  }
}
```

Item_Update.java

```
package GUI;

import BackendCode.Item;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.*;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;

/**
```

```
*
* @author @AbdullahShahid01
*/
public class Item_Update extends JFrame {

    JButton Update_Button, Cancel_Button;
    JLabel ItemID_Label, ItemIDValidity_Label;
    JTextField ItemID_TextField;

    private Item item;

    public Item_Update() {
        super("Update Item");
        setLayout(new FlowLayout());
        setSize(new Dimension(300, 140));
        setResizable(false);
        setLocationRelativeTo(this);
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                Parent_JFrame.getMainFrame().setEnabled(true);
```

```
            dispose();
        }
    });

    Update_Button = new JButton("Update");
    Cancel_Button = new JButton("Cancel");

    ItemID_Label = new JLabel("Enter Item ID to be updated");
    ItemIDValidity_Label = new JLabel();
    ItemID_TextField = new JTextField();

    ItemID_TextField.setPreferredSize(new Dimension(240, 22));
//      ItemID_Label.setPreferredSize(new Dimension(175, 22));
    ItemIDValidity_Label.setPreferredSize(new Dimension(415, 9));
    Update_Button.setPreferredSize(new Dimension(100, 22));
    Cancel_Button.setPreferredSize(new Dimension(100, 22));

    ItemIDValidity_Label.setForeground(Color.red);

    add(ItemID_Label);
    add(ItemID_TextField);
```

```
        add(ItemIDValidity_Label);


        add(Update_Button);
        add(Cancel_Button);


        Update_Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {


                String itemID = ItemID_TextField.getText().trim();


                if (!itemID.isEmpty()) {
                    try {
                        if (Integer.parseInt(itemID) > 0) {
                            ItemIDValidity_Label.setText("");
                        } else {
                            itemID = null;
                            ItemIDValidity_Label.setText("                                    ID cannot be
'0' or negative !");
                        }
                    } catch (NumberFormatException ex) {
```

```
            itemID = null;
            ItemIDValidity_Label.setText("                                    Invalid ID !");
        }
    } else {
        itemID = null;
        ItemIDValidity_Label.setText("                                    Enter Item ID !");
    }

    if (itemID != null) {
        item = Item.SearchByID(Integer.parseInt(itemID));
        if (item != null) {
            Item_UpdateInner cui = new Item_UpdateInner();
            cui.setVisible(true);
            dispose();

        } else {
            JOptionPane.showMessageDialog(null, "Item ID not found !");
        }
    } else {
        ItemIDValidity_Label.setText("                                    Enter Item ID !");
    }
```

```
        }
    }
    );
    Cancel_Button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        }
    });
}

private class Item_UpdateInner extends JFrame {

    JButton Update_Button, Cancel_Button;
    JLabel Name_Label,Model_Label,Type_Label, Specifications_Label,
RentPerHour_Label,Length_Label,Height_Label,Breadth_Label,Weight_Label,
        NameValidity_Label, ModelValidity_Label, TypeValidity_Label,
SpecificationValidity_Label,RentPerHourValidity_Label;
    JTextField Name_TextField, Model_TextField,Type_TextField,Specifications_TextField,
RentPerHour_TextField,Length_TextField,Breadth_TextField,Height_TextField,Weight_TextFi
eld;
```

```
public Item_UpdateInner() {

    super("Update Item");

    setLayout(new FlowLayout());
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    setSize(new Dimension(450, 475));
    setResizable(false);
    setLocationRelativeTo(this);

    Update_Button = new JButton("Update");
    Cancel_Button = new JButton("Cancel");

    Name_Label = new JLabel("Name");
    Model_Label = new JLabel("Model");
    Type_Label = new JLabel("Item type");
    Specifications_Label = new JLabel("Specification");
    RentPerHour_Label = new JLabel("Rent Per Hour (in PKR)");
    Length_Label = new JLabel("Length ");
```

```
Breadth_Label = new JLabel("Breadth");

Height_Label = new JLabel("Height");

Weight_Label = new JLabel("Weight");



NameValidity_Label = new JLabel();

ModelValidity_Label = new JLabel();

TypeValidity_Label = new JLabel();

SpecificationValidity_Label= new JLabel();

RentPerHourValidity_Label = new JLabel();


Name_TextField = new JTextField();

Model_TextField = new JTextField();

Type_TextField = new JTextField();

Specifications_TextField = new JTextField();

RentPerHour_TextField = new JTextField();



Length_TextField = new JTextField();

Breadth_TextField = new JTextField();

Height_TextField = new JTextField();
```

```
Weight_TextField = new JTextField();




Name_TextField.setPreferredSize(new Dimension(240, 22));

Model_TextField.setPreferredSize(new Dimension(240, 22));

Type_TextField.setPreferredSize(new Dimension(240, 22));

Specifications_TextField.setPreferredSize(new Dimension(240, 22));

RentPerHour_TextField.setPreferredSize(new Dimension(240, 22));


Length_TextField.setPreferredSize(new Dimension(240, 22));

Breadth_TextField.setPreferredSize(new Dimension(240, 22));

Height_TextField.setPreferredSize(new Dimension(240, 22));

Weight_TextField.setPreferredSize(new Dimension(240, 22));




Name_Label.setPreferredSize(new Dimension(175, 22));

Model_Label.setPreferredSize(new Dimension(175, 22));

Type_Label.setPreferredSize(new Dimension(175, 22));
```

```
Specifications_Label.setPreferredSize(new Dimension(175, 22));

RentPerHour_Label.setPreferredSize(new Dimension(175, 22));

Length_Label.setPreferredSize(new Dimension(175, 22));

Height_Label.setPreferredSize(new Dimension(175, 22));

Breadth_Label.setPreferredSize(new Dimension(175, 22));

Weight_Label.setPreferredSize(new Dimension(175, 22));


NameValidity_Label.setPreferredSize(new Dimension(415, 9));

ModelValidity_Label.setPreferredSize(new Dimension(415, 9));

TypeValidity_Label.setPreferredSize(new Dimension(415, 9));

SpecificationValidity_Label.setPreferredSize(new Dimension(415, 9));

RentPerHourValidity_Label.setPreferredSize(new Dimension(415, 9));


NameValidity_Label.setForeground(Color.red);

ModelValidity_Label.setForeground(Color.red);

TypeValidity_Label.setForeground(Color.red);

SpecificationValidity_Label.setForeground(Color.red);

RentPerHourValidity_Label.setForeground(Color.red);



add(Name_Label);
```

```
add(Name_TextField);
add(NameValidity_Label);

add(Model_Label);
add(Model_TextField);
add(ModelValidity_Label);

add(Type_Label);
add(Type_TextField);
add(TypeValidity_Label);

add(Specifications_Label);
add(Specifications_TextField);
add(SpecificationValidity_Label);


add(RentPerHour_Label);
add(RentPerHour_TextField);
add(RentPerHourValidity_Label);

add(Length_Label);
```

```
        add(Length_TextField);


        add(Height_Label);
        add(Height_TextField);


        add(Breadth_Label);
        add(Breadth_TextField);


        add(Weight_Label);
        add(Weight_TextField);


        add(Update_Button);
        add(Cancel_Button);


        Update_Button.addActionListener(new ActionListener() {
            @Override


            public void actionPerformed(ActionEvent e) {
System.out.println("Add button worked");
            String name = Name_TextField.getText().trim(),
                    model = Model_TextField.getText().trim(),
```

```
                type = Type_TextField.getText().trim(),
                specification = Specifications_TextField.getText().trim(),
                rentPerHour = RentPerHour_TextField.getText().trim();

            String length = Length_TextField.getText().trim();
            String breadth = Breadth_TextField.getText().trim();
            String height = Height_TextField.getText().trim();
            String weight = Weight_TextField.getText().trim();
            System.out.println("Add button value read complete");

            if (!name.isEmpty()) {
                if (Item.isNameValid(Name_TextField.getText().trim())) {
                    NameValidity_Label.setText("");
//                      name = Name_TextField.getText().trim();
                } else {
                    name = null;
                    NameValidity_Label.setText("Invalid  Item Name !");
                }
            } else {
                name = null;
                NameValidity_Label.setText("Enter Item Name !");
```

```
}

if (!model.isEmpty()) {
    if (Item.isNameValid(model)) {
        ModelValidity_Label.setText("");

    } else {
        model = null;
        ModelValidity_Label.setText("Invalid Model Name !");
    }
} else {
    model = null;
    ModelValidity_Label.setText("Enter Model Name !");
}

if (type.isEmpty()) {

    type = null;
    TypeValidity_Label.setText("Enter Type!");
}
```

```
if (!specification.isEmpty()) {
    if (Item.isNameValid(specification)) {
        SpecificationValidity_Label.setText("");

    } else {
        model = null;
        SpecificationValidity_Label.setText("Invalid specification label !");
    }
} else {
    model = null;
    SpecificationValidity_Label.setText("Enter specification label !");
}


if (!rentPerHour.isEmpty()) {
    if (Item.isNameValid(rentPerHour)) {
        RentPerHourValidity_Label.setText("");

    } else {
        rentPerHour = null;
        RentPerHourValidity_Label.setText("Invalid rentPerHour label !");
```

```
            }
        } else {
            rentPerHour = null;
            RentPerHourValidity_Label.setText("Enter rentPerHour label !");
        }



            try {
                if (name != null && model != null && type != null && rentPerHour != null) {
//new item(ID, Maker, Name, Colour, Type, seatingCapacity, model, Condition, RegNo, RentPerHour, itemOwner)
                    item = new Item(item.getID(),name, model,type,specification,length,breadth,height,weight, rentPerHour);
                    item.Update();


                    Parent_JFrame.getMainFrame().getContentPane().removeAll();
                    Item_Details cd = new Item_Details();
                    Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                    Parent_JFrame.getMainFrame().getContentPane().revalidate();


                    JOptionPane.showMessageDialog(null, "Record Successfully Updated !");
```

```
                    Parent_JFrame.getMainFrame().setEnabled(true);
                    dispose();

                }
            } catch (HeadlessException | NumberFormatException ex) {
                System.out.println(ex);
            }
        }
    }
    );
    Cancel_Button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Parent_JFrame.getMainFrame().setEnabled(true);
            dispose();
        }
    });
  }
 }
}
```

Login.java

```
package GUI;


import java.awt.Color;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Font;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;
```

```
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;

/**
 *
 * @author @AbdullahShahid01
 */
public class Login {

    private final JPanel MiniPanel, MainPanel;
    private final JButton Close_Button, Login_Button;
    private final JLabel PW_Label, UN_Label, Image_jLabel, info_Label;
    private final JTextField UN_TextField;
    private final JPasswordField Password_Field;

    public Login() {

        MiniPanel = new JPanel();
        MainPanel = new JPanel();

        Close_Button = new JButton("Close");
```

```
Login_Button = new JButton("Login");


PW_Label = new JLabel("Password");

UN_Label = new JLabel("Username");

info_Label = new JLabel("Please Enter your Login Details");

Image_jLabel = new JLabel();


UN_TextField = new JTextField("admin");

Password_Field = new JPasswordField("123");


MiniPanel.setBackground(Color.BLUE);

MiniPanel.setForeground(Color.WHITE);

MiniPanel.setLayout(new FlowLayout());


MainPanel.setMinimumSize(new Dimension(1366, 730));

MainPanel.setLayout(new AbsoluteLayout());


Login_Button.setPreferredSize(new Dimension(80, 20));

Close_Button.setPreferredSize(new Dimension(80, 20));


info_Label.setFont(new Font("Consolas", 1, 18)); // Consolas, Bold , 18pt
```

```
info_Label.setForeground(Color.WHITE);


UN_Label.setFont(new Font("Consolas", 0, 18));
UN_Label.setForeground(Color.WHITE);
UN_Label.setPreferredSize(new Dimension(100, 20));

PW_Label.setFont(new Font("Consolas", 0, 18));
PW_Label.setForeground(Color.WHITE);
PW_Label.setPreferredSize(new Dimension(100, 20));




Image_jLabel.setMinimumSize(new Dimension(1366, 730));
Image_jLabel.setIcon(new ImageIcon("LoginImage.jpg"));

UN_TextField.setPreferredSize(new Dimension(200, 20));
Password_Field.setPreferredSize(new Dimension(200, 20));

MiniPanel.add(info_Label);
MiniPanel.add(UN_Label);
MiniPanel.add(UN_TextField);
```

```
    MiniPanel.add(PW_Label);
    MiniPanel.add(Password_Field);
    MiniPanel.add(Login_Button);
    MiniPanel.add(Close_Button);

    MainPanel.add(MiniPanel, new AbsoluteConstraints(50, 150, 350, 125));
    MainPanel.add(Image_jLabel, new AbsoluteConstraints(0, 0));

    Login_Button.addActionListener(new LoginActionListener());
    Close_Button.addActionListener(new LoginActionListener());
}

/**
 * @return the MainPanel
 */
public JPanel getMainPanel() {
    return MainPanel;
}

private class LoginActionListener implements ActionListener {
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        switch (e.getActionCommand()) {
            case "Close": {
                int showConfirmDialog = JOptionPane.showConfirmDialog(null, "You are about to terminate the program.\n"
                        + " Are you sure you want to continue ?", "Close Confirmation", JOptionPane.OK_CANCEL_OPTION, JOptionPane.WARNING_MESSAGE, null);
                if (showConfirmDialog == 0) {
                    System.exit(0);
                }
                break;
            }
            case "Login": {
                if (UN_TextField.getText().trim().equalsIgnoreCase("admin")
                        && String.valueOf(Password_Field.getPassword()).equals("123")) {
                    UN_TextField.setText("");
                    Password_Field.setText("");
                    Runner.getFrame().dispose();
                    Parent_JFrame frame = new Parent_JFrame();
                    MainMenu menu = new MainMenu();
                    JFrame mainFrame = Parent_JFrame.getMainFrame();
```

```
//                    JPanel mainPanel = menu.getMainPanel();
                    mainFrame.add(menu.getMainPanel());
                    mainFrame.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid UserName/Password", "Error",
JOptionPane.ERROR_MESSAGE);
                }
                break;
            }
        }
    }
  }

}
```

MainMenu.java

```
package GUI;


import java.awt.Color;
import java.awt.Dimension;
```

```
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import org.netbeans.lib.awtextra.AbsoluteConstraints;
import org.netbeans.lib.awtextra.AbsoluteLayout;

/**
 *
 * @author @AbdullahShahid01
 */
public class MainMenu implements ActionListener {

    private static JLabel Image_Label;
    private static JButton ItemsButton, CustomerButton,  BookingButton, LogoutButton;
    private JPanel MainPanel;
```

```
    public JPanel getMainPanel() {
        return MainPanel;
    }

    public MainMenu() {
        MainPanel = new JPanel();

        MainPanel.setLayout(new AbsoluteLayout());
        MainPanel.setMinimumSize(new Dimension(1366, 730));

        CustomerButton = new JButton("Customer");
        ItemsButton = new JButton("Items");
//        OwnerButton = new JButton("Pyhsical");
        BookingButton = new JButton("Booking Details");
        LogoutButton = new JButton("Logout");

        Image_Label = new JLabel();

        LogoutButton.setFont(new Font("Tahoma", 1, 14));
        CustomerButton.setFont(new Font("Tahoma", 1, 14));
        ItemsButton.setFont(new Font("Tahoma", 1, 14));
```

```
//      OwnerButton.setFont(new Font("Tahoma", 1, 14));
        BookingButton.setFont(new Font("Tahoma", 1, 14));

        Image_Label.setIcon((new ImageIcon("MainMenuImage.jpeg")));

        CustomerButton.setBackground(new Color(240,240,240));
        ItemsButton.setBackground(new Color(240,240,240));
//      OwnerButton.setBackground(new Color(240,240,240));
        LogoutButton.setBackground(new Color(240,240,240));
        BookingButton.setBackground(new Color(240,240,240));

        MainPanel.add(LogoutButton, new AbsoluteConstraints(1166, 80, 100, 25));
        MainPanel.add(CustomerButton, new AbsoluteConstraints(70, 220, 200, 99));
        MainPanel.add(ItemsButton, new AbsoluteConstraints(70, 360, 200, 99));
//      MainPanel.add(OwnerButton, new AbsoluteConstraints(70, 360, 200, 99));
        MainPanel.add(BookingButton, new AbsoluteConstraints(70, 80, 200, 99));
        MainPanel.add(Image_Label, new AbsoluteConstraints(0, 0, 1370, 710));

        BookingButton.addActionListener(this);
        CustomerButton.addActionListener(this);
//      OwnerButton.addActionListener(this);
```

```
        LogoutButton.addActionListener(this);
        ItemsButton.addActionListener(this);
//        Parent_JFrame.getMainFrame().add(MainPanel);
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        switch (e.getActionCommand()) {
            case "Items": {
                Parent_JFrame.getMainFrame().getContentPane().removeAll();
                Item_Details cd = new Item_Details();
                Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                Parent_JFrame.getMainFrame().getContentPane().revalidate();
            }
            break;
            case "Customer": {
                Parent_JFrame.getMainFrame().getContentPane().removeAll();
                Customer_Details cd = new Customer_Details();
                Parent_JFrame.getMainFrame().add(cd.getMainPanel());
                Parent_JFrame.getMainFrame().getContentPane().revalidate();
```

```
            }
            break;
//            case "Pyhsical": {
//                Parent_JFrame.getMainFrame().getContentPane().removeAll();
//                ItemOwner_Details cd = new ItemOwner_Details();
//                Parent_JFrame.getMainFrame().add(cd.getMainPanel());
//                Parent_JFrame.getMainFrame().getContentPane().revalidate();
//            }
//            break;
            case "Logout": {
                Parent_JFrame.getMainFrame().dispose();
                Runner r = new Runner();
                JFrame frame = r.getFrame();
                Login login = new Login();
                JPanel panel = login.getMainPanel();
                frame.add(panel);
                frame.setVisible(true);
            }
            break;
            case "Booking Details": {
                Parent_JFrame.getMainFrame().getContentPane().removeAll();
```

```
            Booking_Details cd = new Booking_Details();
            Parent_JFrame.getMainFrame().add(cd.getMainPanel());
            Parent_JFrame.getMainFrame().getContentPane().revalidate();
        }
        break;
      }
   }
}
```

Parent_JFrame

```
package GUI;


import BackendCode.Booking;
import BackendCode.Item;
import java.awt.Desktop;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
```

```
import java.io.IOException;
import java.util.ArrayList;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;


/**
 *
 * @author @AbdullahShahid01
 */
public class Parent_JFrame {


    private static JFrame MainFrame;
    private final JMenuBar menu_Bar;
    private final JMenu File, ItemMenu, CustomerMenu,  HelpMenu;
    private final JMenuItem Exit, addItem, updateItem, removeItem, ViewUnbookedItems,
ViewbookedItems,
            addCustomer, updateCustomer, removeCustomer,
            ViewJavaDoc, ViewDocumentation, About;
```

```
    public Parent_JFrame() {
        MainFrame = new JFrame("Lease-A-Item Management System");
        MainFrame.setSize(1366, 730);
        MainFrame.setVisible(true);



        MainFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        MainFrame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                int showConfirmDialog = JOptionPane.showConfirmDialog(null, "You are about to terminate the program.\n"
                        + " Are you sure you want to continue ?", "Close Confirmation", JOptionPane.OK_CANCEL_OPTION, JOptionPane.WARNING_MESSAGE, null);
                if (showConfirmDialog == 0) {
                    System.exit(0);
                }
            }
        });

        menu_Bar = new JMenuBar();
```

```
File = new JMenu("File");

ItemMenu = new JMenu("Items");

CustomerMenu = new JMenu("Customer");

HelpMenu = new JMenu("Help");


Exit = new JMenuItem("Exit");

addItem = new JMenuItem("Add Item");

updateItem = new JMenuItem("Update Item");

removeItem = new JMenuItem("Remove Item");

ViewbookedItems = new JMenuItem("View booked Items");

ViewUnbookedItems = new JMenuItem("View Unbooked Items");


addCustomer = new JMenuItem("Add Customer");

updateCustomer = new JMenuItem("Update Customer");

removeCustomer = new JMenuItem("Remove  Customer");


ViewJavaDoc = new JMenuItem("View JavaDoc");

ViewDocumentation = new JMenuItem("View Documentation");

About = new JMenuItem("About");
```

```
File.add(Exit);

ItemMenu.add(addItem);

ItemMenu.add(updateItem);

ItemMenu.add(removeItem);

ItemMenu.add(ViewbookedItems);

ItemMenu.add(ViewUnbookedItems);


CustomerMenu.add(addCustomer);

CustomerMenu.add(updateCustomer);

CustomerMenu.add(removeCustomer);


HelpMenu.add(ViewJavaDoc);

HelpMenu.add(ViewDocumentation);

HelpMenu.add(About);


menu_Bar.add(File);

menu_Bar.add(ItemMenu);

menu_Bar.add(CustomerMenu);

menu_Bar.add(HelpMenu);


MainFrame.setJMenuBar(menu_Bar);
```

```
    Exit.addActionListener(new Parent_JFrame_ActionListner());
    addItem.addActionListener(new Parent_JFrame_ActionListner());
    updateItem.addActionListener(new Parent_JFrame_ActionListner());
    removeItem.addActionListener(new Parent_JFrame_ActionListner());
    ViewbookedItems.addActionListener(new Parent_JFrame_ActionListner());
    ViewUnbookedItems.addActionListener(new Parent_JFrame_ActionListner());

    addCustomer.addActionListener(new Parent_JFrame_ActionListner());
    updateCustomer.addActionListener(new Parent_JFrame_ActionListner());
    removeCustomer.addActionListener(new Parent_JFrame_ActionListner());

    ViewJavaDoc.addActionListener(new Parent_JFrame_ActionListner());
    ViewDocumentation.addActionListener(new Parent_JFrame_ActionListner());
    About.addActionListener(new Parent_JFrame_ActionListner());

}

public static JFrame getMainFrame() {
    return MainFrame;
}
```

```
    private class Parent_JFrame_ActionListner implements ActionListener {


        @Override
        public void actionPerformed(ActionEvent e) {
            switch (e.getActionCommand()) {
                case "Exit": {
                    int showConfirmDialog = JOptionPane.showConfirmDialog(null, "You are about to terminate the program.\n"
                            + " Are you sure you want to continue ?", "Close Confirmation", JOptionPane.OK_CANCEL_OPTION, JOptionPane.WARNING_MESSAGE, null);
                    if (showConfirmDialog == 0) {
                        System.exit(0);
                    }
                }
                break;
                case "Add Item": {
                    Parent_JFrame.getMainFrame().setEnabled(false);
                    Item_Add ac = new Item_Add();
                    ac.setVisible(true);
                }
                break;
```

```
case "Update Item": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    Item_Update ac = new Item_Update();
    ac.setVisible(true);
}
break;
case "Remove Item": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    Item_Remove ac = new Item_Remove();
    ac.setVisible(true);
}
break;
case "View booked Items": {
    ArrayList<Item> SearchBookedItems_Array = Booking.getBookedItems();
    String result = "";
    if (!SearchBookedItems_Array.isEmpty()) {
        for (int i = 0; i < SearchBookedItems_Array.size(); i++) {
            result += (i + 1) + " : " + SearchBookedItems_Array.get(i) + "\n";
        }
    } else {
        result = "No Items are Booked !";
```

```
    }
    JOptionPane.showMessageDialog(null, result);
}
break;
case "View Unbooked Items": {
    ArrayList<Item> SearchUnBookedItems_Array = Booking.getUnbookedItems();
    String result = "";
    if (!SearchUnBookedItems_Array.isEmpty()) {
        for (int i = 0; i < SearchUnBookedItems_Array.size(); i++) {
            result += (i + 1) + " : " + SearchUnBookedItems_Array.get(i) + "\n";
        }
    } else {
        result = "No UnBooked Items are available !";
    }
    JOptionPane.showMessageDialog(null, result);
}
break;
case "Add Customer": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    Customer_Add aco = new Customer_Add();
    aco.frame.setVisible(true);
```

```
}
break;
case "Update Customer": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    new Customer_Update().frame.setVisible(true);
}
break;
case "Remove  Customer": {
    Parent_JFrame.getMainFrame().setEnabled(false);
    new Customer_Remove().frame.setVisible(true);
}
break;

case "View JavaDoc": {
    if (Desktop.isDesktopSupported()) {
        try {
            File myFile = new File("JavaDoc_Documentation_About.pdf");
            if (myFile.exists()) {
                Desktop.getDesktop().open(myFile);
            } else {
                JOptionPane.showMessageDialog(null, "JavaDoc not found !");
```

```
            }
        } catch (IOException ex) {


        }
    }
}
break;
case "View Documentation": {
    if (Desktop.isDesktopSupported()) {
        try {
            File myFile = new File("JavaDoc_Documentation_About.pdf");
            if (myFile.exists()) {
                Desktop.getDesktop().open(myFile);
            } else {
                JOptionPane.showMessageDialog(null, "JavaDoc not found !");
            }
        } catch (IOException ex) {


        }
    }
}
```

```
            break;
            case "About": {
               JOptionPane.showMessageDialog(null, "THIS PROGRAM IS WRITTEN AS AN ASSIGNMENT OF OBJECT ORIENTED PROGRAMMING PROGRAMMIG!");
            }
            break;

        }
    }
  }
}
```

Runner

```
package GUI;

import java.awt.Dimension;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
/**
 *
 * @author @AbdullahShahid01
 */
public class Runner {

    private static final JFrame FRAME = new JFrame();
    private final ImageIcon icon;
    private final JLabel L1;

    public static JFrame getFrame() {
        return FRAME;
    }

    public Runner() {

        icon = new ImageIcon("WelcomeImage.jpg");
        L1 = new JLabel(icon);
        FRAME.setUndecorated(true);
        FRAME.setSize(new Dimension(1000, 534));
        FRAME.setLocationRelativeTo(null);
```

```
        FRAME.add(L1);
    }

    public static void main(String[] args) {
        Runner runner = new Runner();
        Runner.FRAME.setVisible(true);

        try {
            Thread.sleep(300);
            Login LoginObject = new Login();
            Runner.FRAME.getContentPane().removeAll();
            Runner.FRAME.add(LoginObject.getMainPanel());
            Runner.FRAME.getContentPane().revalidate();

        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}
```