

REPORT

by erangi wanniarachchi

Submission date: 08-Aug-2021 01:15PM (UTC+0800)

Submission ID: 1628942089

File name: report.docx (11.06M)

Word count: 926

Character count: 4846

Tasks

1.

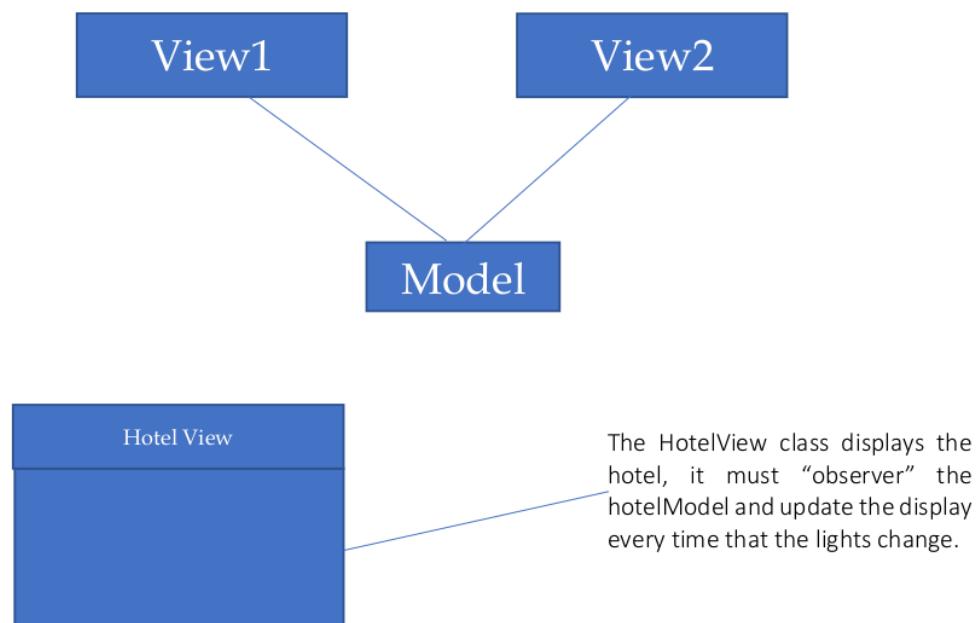
The model-view-controller pattern divides an application into a Model, a View, and a Controller. The Model has nothing to do with the View or the Controller. A Model Observer is the point of view.

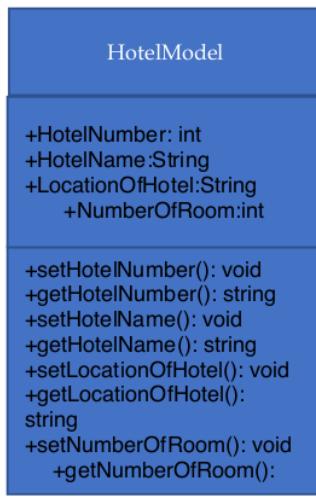
In this case study, there is model the state of the patient and the hotel, and the way the state changes. Patient-Number is auto generated. The model is independent of the view and the controller. None of the code in the model class makes any assumptions about the way that the model is displayed.

It is easy to add alternative views of the same model.

Although a view does depend on the model the views are independent of each other.

Then want to add, display, and exit.





These “getter” methods return values that indicate which hotel is on. The class does not contain any methods to display the hotel.

- The model is unaffected by the viewpoint. No code in the model class makes any assumptions about how the model will be presented.
- It is simple to add more views of the same model.
- Although a view is dependent on the model the views are independent of one another.

CONTROLLER

In this case study, I want to divide the original problem into two sub problems. Those of modelling and viewing the data.

I need to figure out how to translate a patient's mouse click on the Add Hotel and add Patient button into a call to the model's Add Hotel and add Patient function. I'll delegate responsibility for this to a different class known as the controller.

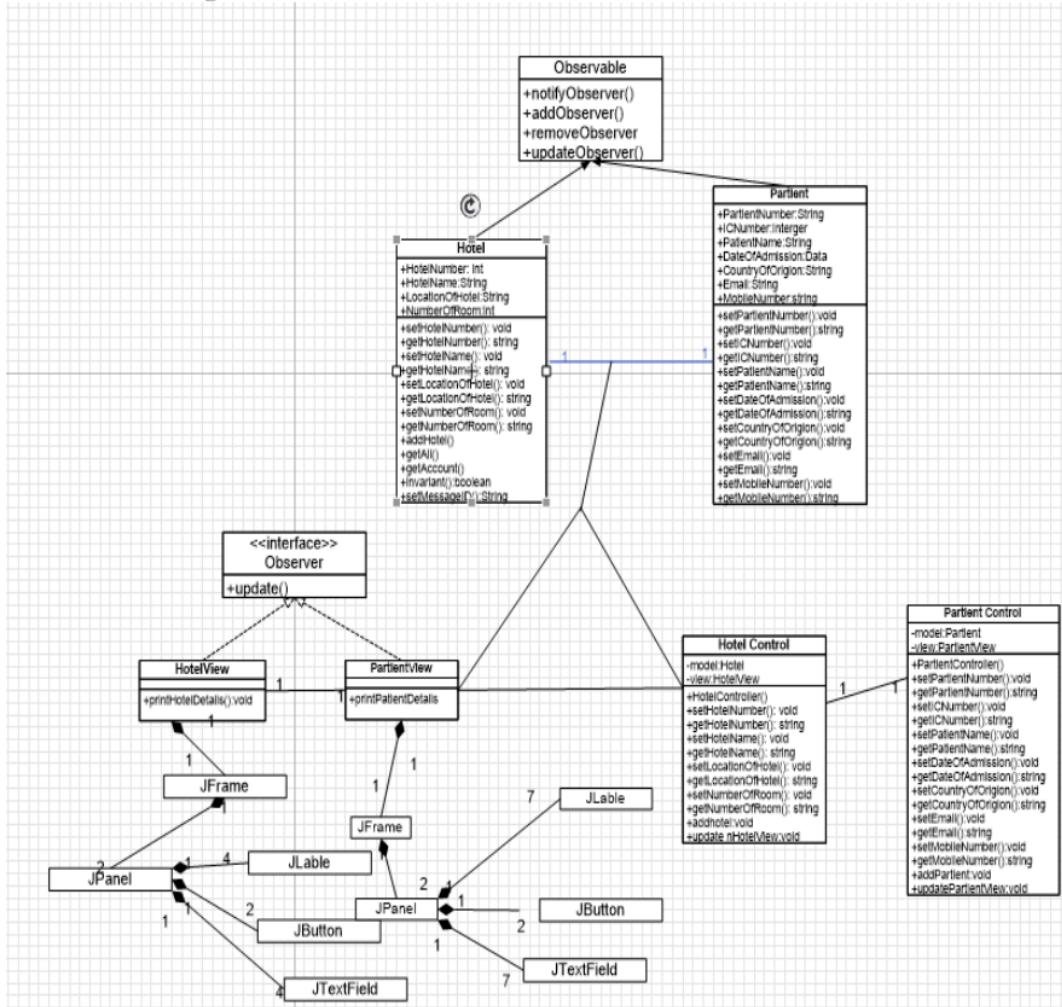
A controller, in general, is a class that analyses events like keystrokes or mouse clicks and turns them into suitable method calls.

- The Observer pattern allows one class to respond to changes in the state of another without requiring the Subject to rely on its Observer.

There are some MVC Features,

- i. Deploy the runtime and framework alongside the application.
- ii. Dependency injection is incorporated into MVC.
- iii. The application is designed as a component-based system, with Model, View, and Controller components. This allows developers to manage the complexities of large-scale projects while still working on specific components.
- iv. There is no need to recompile for each modification. Simply save and reload the browser.
- v. Compilation was completed using the new Roslyn real-time compiler.
- vi. ASP.NET MVC and web API have been combined into a single application

2. UML Class Diagram



3. My patient number is auto generating.

a) Model Classes

This class serves as a link between the control and the view. The user activates a control, which calls a model method to instruct it to change state; once done, it notifies the view that its state has changed. The model does not initiate any activities; instead, it takes and executes orders from the controller.

b) Controller Classes

This class contains all the controls in the GUI. When a control is activated by the user, it calls a method in the Model that instructs it to modify its state accordingly. The listeners defined in the Controller are complex (but tiny) pieces of Java code that detect a user action and call the method in the Model that handles that action.

c) View Classes

This class is in charge of coordinating the look of the GUI. It determines the placement of all controls and displays. When the model notifies the view that the model's state has changed, the view re-displays itself by executing model methods that return all of the information that the view must display.

d) Complete Program

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

// create a method to the HotelAddView
public class HotelAddView extends JPanel {
    // private JTextField numberField = new JTextField("FAX-001", SIZE);
    private JTextField numberField = new JTextField("Hotel No");
    private JTextField nameField = new JTextField("Hotel Name");
    private JTextField locationField = new JTextField("Location Of Hotel");
    private JTextField numberofroomsavailableField = new JTextField("Number Of Rooms Available");
    private JTextField countTextField = new JTextField(SIZE);

    private JLabel nameLabel = new JLabel("Hotel No");
    private JLabel nameLabel = new JLabel("Hotel Name");
    private JLabel locationofhotelLabel = new JLabel("Location Of Hotel");
    private JLabel numberofroomsavailableLabel = new JLabel("Number Of Rooms Available");
    private JButton addButton = new JButton ("Add hotel");
    private JButton resetButton = new JButton ("Reset hotel");
}
```

```
private JTextField countTextField = new JTextField(SIZE);

private JLabel nameLabel = new JLabel("Hotel No");
private JLabel nameLabel = new JLabel("Hotel Name");
private JLabel locationofhotelLabel = new JLabel("Location Of Hotel");
private JLabel numberofroomsavailableLabel = new JLabel("Number Of Rooms Available");
private JButton addButton = new JButton ("Add hotel");
private JButton resetButton = new JButton ("Reset hotel");
```

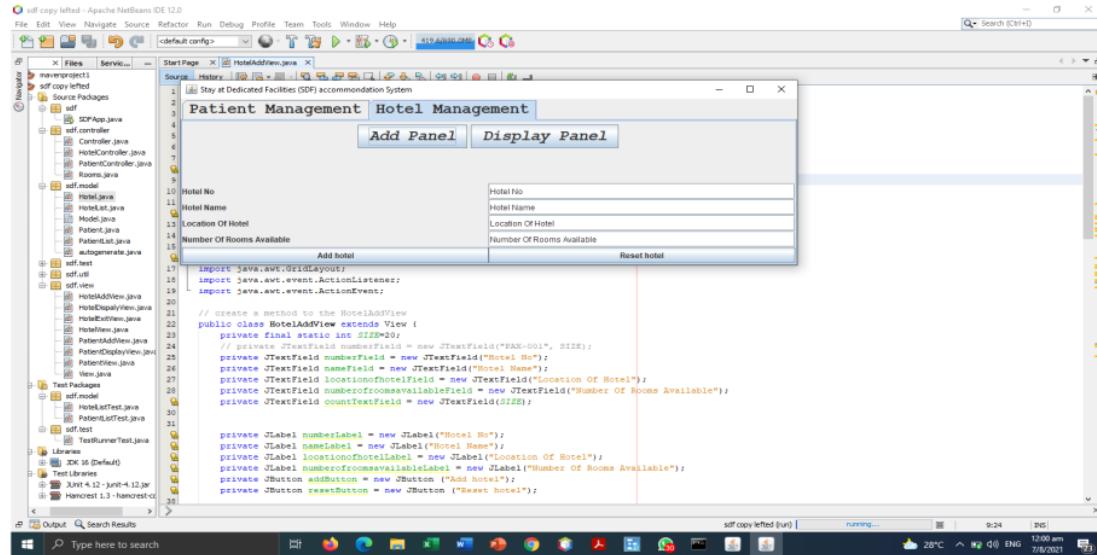
```
sdf copy lefted (run) | running... | 9:24 | INS
```

```
28°C ⚡ ENG 11:59 pm 6/6/2021
```

```
sdf copy lefted (run) | running... | 9:24 | INS Windows (CPU)
```

```
28°C ⚡ ENG 11:48 pm 6/6/2021
```

This is the system of the patient management system GUI for SDF.



This is the system of the hotel management system GUI for SDF.

4. Pre-condition

A pre-condition is a Boolean statement that defines the state that a method must be in for it to operate properly. There are several methods to define the criteria for a function.

Throughout this lesson and in the textbook, we will employ a pair of statements for each function, known as the function's precondition and postcondition.

A TRUE pre-condition indicates that there is no pre-condition. When TRUE is true, the process works.

A FALSE pre-condition would define a process that would never work. The process works only when FALSE is true, which is never satisfied.

Example:

```
Int i=0;           → true
for (Patient p: model.getAll())
{
    tf = new JTextField(SIZE);
```

```
i=i+1;  
String s=String.valueOf(i);  
tf.setText("PAX-"+s);  
contents.add(tf);
```

if $i=0$ value is true, then go to in a for loop. Get a text value. The precondition statement specifies what must be true before calling the function.

Post-condition

post-condition : A post-condition gives a state that should be true when the procedure has finished.

A pre-condition and post-condition pair for a function f specifies that:

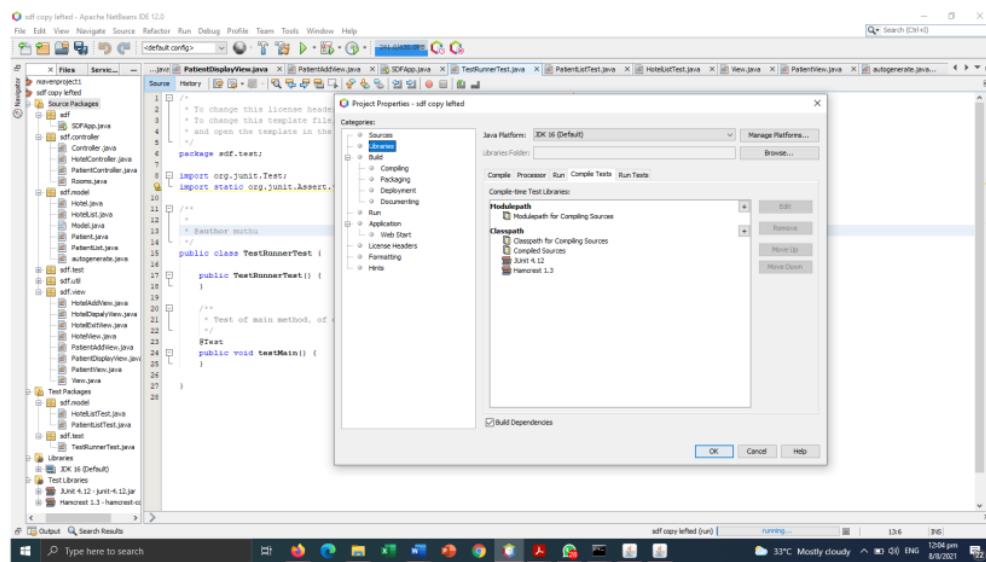
a correct implementation of function f must ensure that when f is called in any state that satisfies the pre-condition, then in any terminating state of f , the post condition is true

Example:

```
Int i=0;  
for (Patient p: model.getAll())  
{  
    tf = new JTextField(SIZE);  
  
    i=i+1;  
    String s=String.valueOf(i);  
    tf.setText("PAX-"+s);  
    contents.add(tf);
```

After that $i=i+1$, get string value and set as PAX-1. When a function completes its execution, it guarantees that the post-condition is true.

5. Any JUnit libraries were listed in the "Add Library..." box in my Project Properties window.



HotelListTest.java

HotelListTest.java

```
1 package sdf.model;
2
3 import java.util.List;
4 import org.junit.After;
5 import org.junit.AfterClass;
6 import org.junit.Assert;
7 import org.junit.Test;
8 import static org.junit.Assert.*;
9
10 import org.junit.Before;
11 import org.junit.Test;
12 import sdf.model.HotelList;
13 import sdf.model.Hotel;
```

HotelListTest.java

```
1 /**
2  * Test case for add, getAll, getCount methods of class PatientList.
3  */
4
5 @Test
6 public void testAddGetAllGetCount() {
7     Hotel hotel1 = new Hotel();
8     Hotel hotel2 = new Hotel();
9     hotel1.setNumber("9123");
10    hotel2.setNumber("9345");
11
12    testInvariant();
13    hotelList.add(hotel1);
14    testInvariant();
15    hotelList.add(hotel2);
16    testInvariant();
17
18    List<Hotel> hotels = hotelList.getAll();
19    assertEquals("Patient1 missing", hotel1, hotels.contains(hotel1));
20    assertEquals("Patient2 missing", hotel2, hotels.contains(hotel2));
21    assertEquals("count error", hotelList.getCount(), 2);
22}
23
24 /**
25  * Test of invariant method, of class PatientList.
26  */
27 @Test
28 public void testInvariant() {
29     assertEquals("patientList is null", hotelList.invariant());
30 }
31
32 private void assertEquals(String count_error, boolean b) {
33     throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
34 }
35
36 /**
37  * Test of add method, of class HotelList.
38  */
39 @Test
40 public void testAdd() {
41 }
```

```
 PatientListTest.java
 1 package sdf.copy.legal;
 2
 3 import org.junit.Assert;
 4 import org.junit.Before;
 5 import org.junit.Test;
 6 import org.junit.AfterClass;
 7 import org.junit.BeforeClass;
 8
 9 import java.util.List;
10 import org.junit.After;
11 import org.junit.AfterClass;
12 import org.junit.Test;
13 import static org.junit.Assert.*;
14 import org.junit.Before;
15
16
17 public class PatientListTest {
18     private PatientList patientList;
19
20     public PatientListTest() {
21
22     }
23
24     /**
25      * @throws Exception
26     */
27     @BeforeClass
28     public static void setUpClass() throws Exception {
29
30     }
31
32     @AfterClass
33     public static void tearDownClass() throws Exception {
34
35     }
36
37     @Before
38     public void setUp() {
39         patientList= new PatientList();
40     }
41
42     /**
43      * Test of add method, of class PatientList.
44      */
45     @Test
46     public void testAdd() {
47
48     }
49
50     /**
51      * Test of getAll method, of class PatientList.
52      */
53     @Test
54     public void testGetAll() {
55
56     }
57
58     /**
59      * Test of getCount method, of class PatientList.
60      */
61     @Test
62     public void testGetCount() {
63
64     }
65
66 }
```

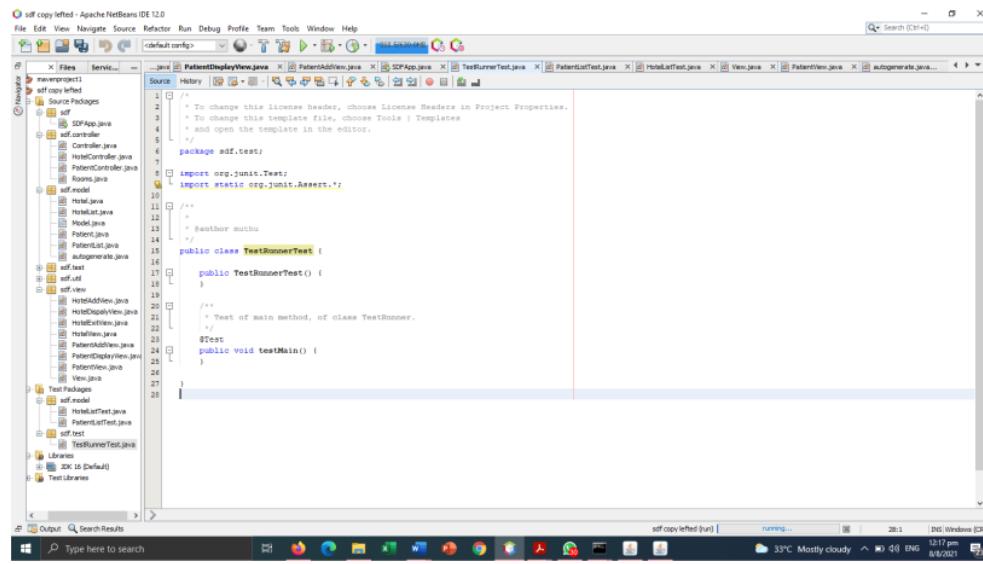
PatientListTest

```
 PatientListTest.java
 1 package sdf.copy.legal;
 2
 3 import org.junit.Assert;
 4 import org.junit.Before;
 5 import org.junit.Test;
 6 import org.junit.AfterClass;
 7 import org.junit.BeforeClass;
 8
 9 import java.util.List;
10 import org.junit.After;
11 import org.junit.AfterClass;
12 import org.junit.Test;
13 import static org.junit.Assert.*;
14 import org.junit.Before;
15
16
17 public class PatientListTest {
18     private PatientList patientList;
19
20     public PatientListTest() {
21
22     }
23
24     /**
25      * @throws Exception
26     */
27     @BeforeClass
28     public static void setUpClass() throws Exception {
29
30     }
31
32     @AfterClass
33     public static void tearDownClass() throws Exception {
34
35     }
36
37     @Before
38     public void setUp() {
39         patientList= new PatientList();
40     }
41
42     /**
43      * Test of add method, of class PatientList.
44      */
45     @Test
46     public void testAdd() {
47
48     }
49
50     /**
51      * Test of getAll method, of class PatientList.
52      */
53     @Test
54     public void testGetAll() {
55
56     }
57
58     /**
59      * Test of getCount method, of class PatientList.
60      */
61     @Test
62     public void testGetCount() {
63
64     }
65
66 }
```

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Title Bar:** "soft copy lefted - Apache NetBeans IDE 12.0"
- Toolbar:** File, Edit, View, Navigator, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Project Explorer:** Shows the project structure with packages like `soft.copy.lefted`, `soft.copy.lefted.model`, `soft.copy.lefted.view`, and `soft.copy.lefted.test`. Under `soft.copy.lefted`, there are files like `SOFApp.java`, `Controller.java`, `HostController.java`, `HostListController.java`, `HostModel.java`, `HostView.java`, `Patient.java`, `PatientList.java`, `PatientListTest.java`, `PatientModel.java`, `PatientView.java`, `soft.copy.lefted.autogenerated.java`, and `soft.copy.lefted.java`.
- Code Editor:** The main window displays the `PatientListTest.java` file. The code includes test cases for adding patients to a `PatientList` and verifying its count and invariant method. The code uses annotations like `@Before` and `@Test` from the JUnit library.
- Toolbars:** Standard Java development tools like Find, Replace, Copy, Paste, etc., are visible along the top.
- Status Bar:** Shows the current file as "soft copy lefted [run]", the build status as "running", and the system time as "12:10 pm 8/20/2023".
- Task List:** A sidebar on the right lists tasks such as "Search (Ctrl+F)", "Run", "Stop", "Run All", "Stop All", and "Run Test".

TestRunnerTest



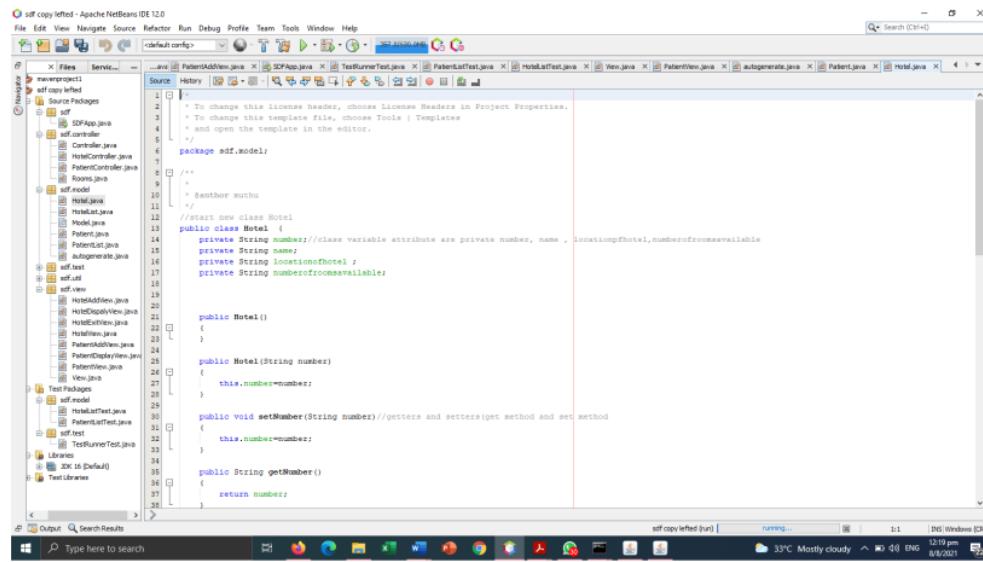
The screenshot shows the Apache NetBeans IDE interface with the title bar "sdf copy lefted - Apache NetBeans IDE 12.0". The left pane displays the project structure under "Navigator" with packages like "soft", "soft.controller", "soft.model", "soft.test", and "soft.view". The right pane shows the code editor for "TestRunnerTest.java". The code defines a class "TestRunnerTest" with a main method:

```
1  * To change this license header, choose License Headers in Project Properties.
2  * To change this template file, choose Tools | Templates
3  * and open the template in the editor.
4
5  package sdf.test;
6
7  import org.junit.Test;
8  import static org.junit.Assert.*;
9
10 /**
11  * ...
12  */
13 /**
14  * Another mutu
15  */
16
17 public class TestRunnerTest {
18
19     /**
20      * Test of main method, of class TestRunner.
21     */
22     @Test
23     public void testMain() {
24
25     }
26
27 }
28
```

7.

MODEL

Hotel.java



The screenshot shows the Apache NetBeans IDE interface with the title bar "sdf copy lefted - Apache NetBeans IDE 12.0". The left pane displays the project structure under "Navigator" with packages like "soft", "soft.controller", "soft.model", "soft.test", and "soft.view". The right pane shows the code editor for "Hotel.java". The code defines a class "Hotel" with private attributes "number", "name", and "locationofhotel", and a constructor and methods for setting and getting "number".

```
1  * To change this license header, choose License Headers in Project Properties.
2  * To change this template file, choose Tools | Templates
3  * and open the template in the editor.
4
5  package sdf.model;
6
7  /**
8  * ...
9  */
10 /**
11  * Another mutu
12  */
13 /**
14  * Start new class Hotel
15  */
16 public class Hotel {
17     private String number;//class variable attribute are private number, name , locationofhotel,numberoffromavailable
18     private String name;
19     private String locationofhotel ;
20     private String numberoffromavailable;
21
22     public Hotel()
23     {
24
25     }
26
27     public Hotel(String number)
28     {
29         this.number=number;
30     }
31
32     public void setNumber(String number)//getters and setters(get method and set method
33     {
34         this.number=number;
35     }
36
37     public String getNumber()
38     {
39         return number;
40     }
41 }
```

Hotel.java

```
public String getName()
{
    return name;
}

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

public String getName()
{
    return name;
}

public void setLocationsofhotel (String locationsofhotel )
{
    this.locationsofhotel = locationsofhotel ;
}

public String getlocationsofhotel ()
{
    return locationsofhotel ;
}

public String getNumberoffromsavailable()
{
    return numberoffromsavailable ;
}

public void setNumberoffromsavailable(String numberoffromsavailable )
{
    this.numberoffromsavailable = numberoffromsavailable;
}
```

HotelList.java

HotelList.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package srf.model;
import java.util.List;
import java.util.ArrayList;
/**
 * Author muthu
 */
public class HotelList extends Model {
    private List<Hotel> hotels = new ArrayList<Hotel>(); //used java array list to store data
    public void add(Hotel h)
    {
        hotels.add(h);
        notifyObserver();
    }

    public List<Hotel> getAll()
    {
        return hotels;
    }

    @Override
    public int getCount()
    {
        return hotels.size();
    }

    public boolean invariant()
    {
        return hotels != null;
    }

    boolean contains(Hotel Hotel2) {
        return hotels.contains(Hotel2);
    }
}
```

HotelList.java

```
/*
 * Author : Sambor mathu
 */
public class HotelList extends Model {
    private List<Hotel> hotels = new ArrayList<Hotel>(); //used java array list to store data

    public void add(Hotel h)
    {
        hotels.add(h);
        notifyObserver();
    }

    public List<Hotel> getAll()
    {
        return hotels;
    }

    @Override
    public int getCount()
    {
        return hotels.size();
    }

    public boolean invariant()
    {
        return hotels != null;
    }

    boolean contains(Hotel Hotel2)
    {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
}
```

Model.java

Model.java

```
/*
 * This model implements gdf.util.Observable
 * It is used as a base class for other concrete models
 */
public abstract class Model implements Observable
{
    private List<Observer> observers = new ArrayList<Observer>();
    private String messageId = null;

    /**
     * Default Constructor
     */
    @Pre
    @Sproc. True
    @Spost. A new class without any parameter being initialized
    */
    public Model() {}

    /**
     * Set the messageId
     * Message ID to notify the registered Observers
     * Pre. True
     * Post. the instance variable messageId is updated
     */
    public void setMessageId(String messageId)
    {
        this.messageId = messageId;
    }
}
```

```
soft copy lefied - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Source Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Test Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Libraries
JDK 16 (Default)
Test Libraries
soft.copy.lefied - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Source Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Test Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Libraries
JDK 16 (Default)
Test Libraries
```

This method implements the register method in `soft.util.Observable`.

```
    /**
     * This method implements the register method in soft.util.Observable.
     *
     * @param observer
     *     Any object that implements the soft.util.Observer interface
     */
    public void register(Observer observer)
    {
        observers.add(observer);
    }
```

This method implements the unregister method in `soft.util.Observable`.

```
    /**
     * This method implements the unregister method in soft.util.Observable.
     *
     * @param observer
     *     A registered observer
     */
    public void unregister(Observer observer)
    {
        observers.remove(observer);
    }
```

This method implements the notifyObserver method in `soft.util.Observable`.

```
    /**
     * This method implements the notifyObserver method in soft.util.Observable.
     *
     * @pre True
     * @post calling all registered observer's update method.
     */
    public void notifyObserver()
    {
        for(Observer observer: observers)
        {
            observer.update(this.messageID);
        }
    }
```

```
soft copy lefied - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Source Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Test Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Libraries
JDK 16 (Default)
Test Libraries
soft.copy.lefied - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Source Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Test Packages
soft.copy.lefied
soft.controller
soft.model
soft.test
soft.view
Libraries
JDK 16 (Default)
Test Libraries
```

```
    /**
     * This method implements the unregister method in soft.util.Observable.
     *
     * @param observer
     *     An observer to remove
     */
    public void unregister(Observer observer)
    {
        observers.remove(observer);
    }
```

All models are expected to maintain a list of entities.

```
    /**
     * All models are expected to maintain a list of entities
     * This method is used to return the number of entities in the list
     *
     * @return The Number of entities in the list
     * @post Return the number of entities in the list
     */
    public abstract int getCount();
```

Patient.java

```
package sdf.model;

public class Patient
{
    private String number;
    private String name;
    private String icNo;
    private String countryOfOrigin;
    private String dateOfAdmission;
    private String email;
    private String mobileNumber;

    public Patient()
    {
    }

    public Patient(String number)
    {
        this.number = number;
    }

    public void setNumber(String number)
    {
        this.number = number;
    }

    public String getNumber()
    {
        return number;
    }

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

    public String getName()
    {
        return name;
    }
```

```
public String getName()
{
    return name;
}

public void setCountryOfOrigin (String countryOfOrigin )
{
    this.countryOfOrigin = countryOfOrigin ;
}

public String getCountryOfOrigin ()
{
    return countryOfOrigin ;
}

public void setDateOfAdmission(
        String dateOfAdmission)
{
    this.dateOfAdmission = dateOfAdmission;
}

public String getDateOfAdmission()
{
    return dateOfAdmission;
}

public void setICNo(String icNo)
{
    this.icNo = icNo;
}

public String getICNo()
{
    return icNo;
}

public void setEmail(String email)
```

Apache NetBeans IDE 12.0

```
public void setEmail(String email)
{
    this.email=email;
}
public String getEmail()
{
    return email;
}
public void setMobileNumber(String mobileNumber)
{
    this.mobileNumber=mobileNumber;
}
public String getMobileNumber()
{
    return mobileNumber;
}
public String toString()
{
    String s="";
    s+="Patient["+
        "name='"+name+"', "
        "idNo='"+idNo+"', "
        "countryOfOrigin='"+countryOfOrigin+"', "
        "dateOfAdmission='"+dateOfAdmission+"', "
        "email='"+email+"', "
        "mobileNumber='"+mobileNumber+"', "
    ];
    return s;
}
public String getMobileNumber(String pno)
{
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}
```

PatientList.java

Apache NetBeans IDE 12.0

```
package sf.model;
import java.util.List;
import java.util.ArrayList;
public class PatientList extends Model
{
    private List<Patient> patients = new ArrayList<Patient>();//using java array list we can get store data
    public void add(Patient p)
    {
        patients.add(p);
        notifyObserver();
    }
    public List<Patient> getAll()
    {
        return patients;
    }
    public int getCount()
    {
        return patients.size();
    }
    public boolean invariant()
    {
        return patients != null;
    }
}
```

Autogenerate.java

The screenshot shows the Apache NetBeans IDE interface with the title bar "sdf copy lefted - Apache NetBeans IDE 12.0". The main window displays the "Source" tab for the "Autogenerate.java" file. The code content is as follows:

```
1  * To change this license header, choose License Headers in Project Properties.
2  * To change this template file, choose Tools | Templates
3  * and open the template in the editor.
4  *
5  * Author: another mathu
6  */
7 package sdf.model;
8
9 /**
10  * Another mathu
11  */
12 class autogenerate {
13 }
14
15
```

The project navigation pane on the left shows several Java files and packages under the "sdf" source package, including SDFApp.java, Controller.java, Model.java, PatientController.java, HotelController.java, Hotel.java, Hospital.java, ModelTest.java, Patient.java, PatientTest.java, and autogenerate.java. The status bar at the bottom indicates "sdf copy lefted (run)" and "15:1 15:1 ING Windows (CRF)".

VIEW

HotelAddView.java

The screenshot shows the Apache NetBeans IDE interface with the title bar "sdf copy lefted - Apache NetBeans IDE 12.0". The main window displays the "Source" tab for the "HotelAddView.java" file. The code content is as follows:

```
1  * To change this license header, choose License Headers in Project Properties.
2  * To change this template file, choose Tools | Templates
3  * and open the template in the editor.
4  *
5  * Author: another mathu
6  */
7 package sdf.view;
8
9 //import classes
10 import java.awt.Container;
11 import sdf.model.Hotel;
12 import sdf.model.HotelList;
13 import sdf.controller.HotelController;
14 import javax.swing.JFrame;
15 import javax.swing.JLabel;
16 import javax.swing.JPanel;
17 import javax.swing.JTextField;
18 import javax.swing.JButton;
19 import javax.swing.border.BorderLayout;
20 import java.awt.GridLayout;
21 import java.awt.event.ActionListener;
22 import java.awt.event.ActionEvent;
23
24 // create a method to the HotelAddView
25 public class HotelAddView extends View {
26     private JPanel panel;
27     private JTextField numberField = new JTextField("0001", 5);
28     private JTextField numberField = new JTextField("0001");
29     private JTextField nameField = new JTextField("Hotel Name");
30     private JTextField locationField = new JTextField("Location Of Hotel");
31     private JTextField numbersAvailableField = new JTextField("0000");
32     private JTextField countTextField = new JTextField("0000");
33
34     private JLabel numberLabel = new JLabel("Hotel No.");
35     private JLabel nameLabel = new JLabel("Hotel Name");
36     private JLabel locationLabel = new JLabel("Location Of Hotel");
37     private JLabel numbersAvailableLabel = new JLabel("Number Of Rooms Available");
38     private JButton addButton = new JButton("Add Hotel");
39     private JButton resetButton = new JButton("Reset Hotel");
40 }
```

The project navigation pane on the left shows several Java files and packages under the "sdf" source package, including SDFApp.java, Controller.java, Model.java, PatientController.java, HotelController.java, Hotel.java, Hospital.java, ModelTest.java, Patient.java, PatientTest.java, and autogenerate.java. The status bar at the bottom indicates "sdf copy lefted (run)" and "20:1 20:1 ING Windows (CRF)".

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Title Bar:** sif copied - Apache NetBeans IDE 12.0
- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help
- Toolbar:** Standard Java development tools like New, Open, Save, Cut, Copy, Paste, Find, etc.
- Project Explorer:** Shows the project structure with packages like `sif`, `sif.model`, `sif.controller`, `sif.view`, and `Test Packages`. Under `sif`, there are files like `SIFPanel.java`, `Controller.java`, `HotelController.java`, `PatientController.java`, `Rooms.java`, `Hotel.java`, `Model.java`, `Patient.java`, `PatientController.java`, `autogenerated.java`, `sif.util`, and `sif.view`. Under `sif.model`, there are `HotelList.java`, `PatientList.java`, `ResultList.java`, `PatientDisplayView.java`, `PatientAddView.java`, `HotelAddView.java`, `HotelEditView.java`, `HotelDeleteView.java`, `Hotels.java`, `PatientAddView.java`, `PatientDisplayView.java`, `PatientEditView.java`, and `View.java`. Under `Test Packages`, there are `TestUtil.java`, `HotelListTest.java`, `PatientListTest.java`, and `TestRunnerTest.java`.
- Code Editor:** The main window displays the `HotelAddView.java` file. The code implements a `JFrame` for adding a new hotel. It includes a constructor that initializes a `GridLayout(0,2)` layout and adds various `JLabel` and `JTextField` components. It also includes action listeners for the `add` and `reset` buttons.
- Output and Search:** Bottom-left panel for Output and Search Results. Bottom-right panel showing running status, system info (33°C, Mostly cloudy), and system tray icons.

```
public HotelAddView(HotelList model, HotelController controller)
{
    // Code here runs when a new
    super(model, controller);
    setLayout(new GridLayout(0,2));
    add(numberLabel); add(numberField);
    add(nameLabel); add(nameField);
    add(locationLabel); add(locationofhotelField);
    add(numbersfromavailableLabel); add(numbersfromavailableField);
    add(addButton); add(resetButton);
    resetButton.addActionListener(
        new ActionListener ()
    {
        @Override
        public void actionPerformed (ActionEvent e)
        {
            //numberField.setText("");
            numberField.setText(")");
            nameField.setText(")");
            locationofhotelField.setText(")");
            numbersfromavailableField.setText(")");
        }
    });
    addButton.addActionListener(
        new ActionListener ()
    {
        public void actionPerformed (ActionEvent e)
        {
            //String number = "1";
            String number = numberField.getText();
            String name = nameField.getText();
            String location = locationofhotelField.getText();
            String numbersfromavailable = numbersfromavailableField.getText();
        }
    });
}
```

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Title Bar:** sff copy-lefted - Apache NetBeans (IDE 12.0)
- Menu Bar:** File, Edit, View, Navigate, Sources, Refactor, Run, Debug, Profile, Team, Tools, Window, Help
- Toolbar:** Standard Java development tools like New, Open, Save, Cut, Copy, Paste, Find, etc.
- Project Explorer (Left):** Shows the project structure under "src":
 - src/main/java:
 - sff
 - sff/copy-lefted
 - sff/controller
 - HotelController.java
 - HotelListController.java
 - sff/model
 - Hotel.java
 - HotelList.java
 - HotelListController.java
 - Patent.java
 - PatentList.java
 - PatentListController.java
 - autoGenerate.java
 - Hotel.java
 - HotelList.java
 - HotelListController.java
 - Patent.java
 - PatentList.java
 - PatentListController.java
 - autoGenerate.java
 - sff/view
 - HotelAddView.java
 - HotelDisplayView.java
 - HotelEditView.java
 - HotelList.java
 - PatentAddView.java
 - PatentDisplayView.java
 - PatentEditView.java
 - View.java
 - Source Editor (Center):** Displays the Java code for HotelList.java. The code implements an ActionListener for an add button, reading values from text fields and calling methods from the HotelListController and HotelList model.
 - Output (Bottom Left):** Shows the command "sff copy-lefted (run)" and status "running...".
 - Task List (Bottom Left):** Shows tasks like "TestRunnerTest.java" and "TestRunnerTest.java".
 - Search Results (Bottom Left):** Shows search results for "copy-lefted".
 - System Tray (Bottom Right):** Shows system icons for battery, network, volume, and date/time.
 - Desktop Icons (Bottom):** Standard Windows icons for File Explorer, Control Panel, Task View, Start, and others.

HotelDisplayView.java

```

vif copy-left - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
< default config >
Source History < HotelAdmView.java > HotelDisplayView.java < HotelEditView.java > HotelList.java < PatientList.java > Result.java < PatientDisplayView.java > PatientAddView.java ...
< HotelDisplayView.java >
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package vif.view;
7
8  import vif.view.View;
9  import vif.model.Hotel;
10 import vif.model.HotelList;
11 import vif.controller.HotelController;
12 import java.awt.Container;
13 import javax.swing.JLabel;
14 import javax.swing.JTextField;
15 import java.awt.BorderLayout;
16 import java.awt.GridLayout;
17 import java.awt(JFrame);
18 import javax.swing.JPanel;
19 import vif.controller.HotelController;
20 import vif.model.Hotel;
21 import vif.model.HotelList;
22
23 public class HotelDisplayView extends View //create view
24 {
25     private JPanel content;
26     private final static int SIZE=20;
27
28     private JLabel numberLabel = new JLabel("Hotel-#");
29     private JLabel locationOfHotelLabel = new JLabel("Location Of Hotel");
30     private JLabel numberFromAvailableLabel = new JLabel("Number Of Roomsavailable");
31     private JLabel countLabel = new JLabel("Count");
32     private JTextField countTextField = new JTextField(SIZE);
33
34     public HotelDisplayView(HotelList model, HotelController controller)
35     {
36         super(model, controller);
37         setLayout(new BorderLayout());
38         displayContent();
39         add(countLabel, BorderLayout.WEST);
40         add(countTextField, BorderLayout.EAST);
41     }
42
43     public void displayContent()
44     {
45         HotelList model = (HotelList)model();
46         Container content = getContentPane();
47         content.setLayout(new GridLayout(0,4));
48         content.add(numberLabel);
49         content.add(countLabel);
50         content.add(locationOfHotelLabel);
51         content.add(numberFromAvailableLabel);
52         JTextField tf;
53
54         for (Hotel h: model.getAll())
55         {
56             tf = new JTextField(SIZE);
57             tf.setText(h.getNumber());
58             content.add(tf);
59
60             tf = new JTextField(SIZE);
61             tf.setText(h.getName());
62             content.add(tf);
63
64             tf = new JTextField(SIZE);
65             tf.setText(" "+h.getLocationofHotel());
66             content.add(tf);
67
68             tf = new JTextField(SIZE);
69
70         }
71     }
72 }

```

```

vif copy-left - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
< default config >
Source History < HotelAdmView.java > HotelDisplayView.java < HotelEditView.java > HotelList.java < PatientList.java > Result.java < PatientDisplayView.java > PatientAddView.java ...
< HotelDisplayView.java >
31
32     public HotelDisplayView(HotelList model, HotelController controller)
33     {
34         super(model, controller);
35         setLayout(new BorderLayout());
36         displayContent();
37         add(countLabel, BorderLayout.WEST);
38         add(countTextField, BorderLayout.EAST);
39         displayContent();
40     }
41
42     public void displayContent()
43     {
44         HotelList model = (HotelList)model();
45         Container content = getContentPane();
46         content.setLayout(new GridLayout(0,4));
47         content.add(numberLabel);
48         content.add(countLabel);
49         content.add(locationOfHotelLabel);
50         content.add(numberFromAvailableLabel);
51         JTextField tf;
52
53         for (Hotel h: model.getAll())
54         {
55             tf = new JTextField(SIZE);
56             tf.setText(h.getNumber());
57             content.add(tf);
58
59             tf = new JTextField(SIZE);
60             tf.setText(h.getName());
61             content.add(tf);
62
63             tf = new JTextField(SIZE);
64             tf.setText(" "+h.getLocationofHotel());
65             content.add(tf);
66
67             tf = new JTextField(SIZE);
68
69         }
70     }
71 }

```

Apache NetBeans IDE 12.0

```
HotelExitView.java
```

```
content.add(tf);

tf = new JTextField(SIZE);
tf.setText(" "+h.getlocationsofhotel());
content.add(tf);

tf = new JTextField(SIZE);
tf.setText(h.getNumberoffroomsavailable ());
content.add(tf);

}

add(content, BorderLayout.NORTH);

}

public void displayCount()
{
    countTextField.setText(""+getModel().getCount());
}

@Override
public void update(String messageId)
{
    remove(countField);
    displayContent();
    displayCount();
}

JFrame topFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
topFrame.pack();
topFrame.repaint();
}
```

HotelExitView.java

Apache NetBeans IDE 12.0

```
HotelExitView.java
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package mdf.view;

/**
 *
 * @author sumbu
 */
public class HotelExitView {
```

HotelView.java

```
sd copy lefted - Apache Netbeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
default config - HotelView.java HotelDisplayView.java HotelExitView.java HotelView.java HotelList.java Result.java PatientDisplayView.java PatientAddView.java
Project Sources Packages Tools Window Help
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History
1 /* To change this license header, choose License Headers in Project Properties.
2 * To change this template file, choose Tools | Templates
3 * and open the template in the editor.
4 */
5
6 package sdf.view;
7 import sdf.view.View;
8 import sdf.view.HotelExitView;
9 import sdf.view.HotelDisplayView;
10 import sdf.view.HotelView;
11 import sdf.model.HotelList;
12 import sdf.controller.HotelController;
13 import java.awt.BorderLayout;
14 import javax.swing.JFrame;
15 import javax.swing.JPanel;
16 import javax.swing.CardLayout;
17 import javax.swing.JButton;
18 import javax.swing.JPanel;
19 import javax.swing.border.BorderLayout;
20 import javax.swing.event.ActionListener;
21 import javax.swing.event.ActionEvent;
22
23 public class HotelView extends View
24 {
25     private JButton addition = new JButton("Add Panel");
26     private JButton displayButton = new JButton("Display Panel");
27     private JButton exitButton = new JButton("Exit");
28     private HotelAddView addView;
29     private HotelDisplayView displayView;
30     //private HotelExitView exitView;
31
32     private JPanel cards = new JPanel(new CardLayout());
33     private JPanel buttons = new JPanel();
34     private static final String ADD = "ADD";
35     private static final String DISPLAY = "DISPLAY";
36     //private static final String EXIT = "EXIT";
37
38     public HotelView(HotelList model, HotelController controller)
39     {
40         super(model, controller);
41
42         addView = new HotelAddView(model, controller);
43         displayView = new HotelDisplayView(model, controller);
44         exitView = new HotelExitView(model, controller);
45         buttons.add(addButton);
46         buttons.add(displayButton);
47         buttons.add(exitButton);
48
49         // Set Buttons Font
50         addButton.setFont(View.btfFont);
51         displayButton.setFont(View.btfFont);
52         exitButton.setFont(View.btfFont);
53
54         setLayout(new BorderLayout());
55         add(buttons, BorderLayout.NORTH);
56         add(new JLabel(""), BorderLayout.CENTER);
57         cards.add(displayView, DISPLAY);
58         cards.add(addView, ADD);
59         //cards.add(exitView, EXIT);
60         add(cards, BorderLayout.SOUTH);
61
62         showAddView();
63
64         addButton.addActionListener(
65             new ActionListener()
66             {
67                 public void actionPerformed (ActionEvent e)
68                 {
69                     showAddView();
70                 }
71             });
72
73         displayButton.addActionListener(
74             new ActionListener()
75             {
76                 public void actionPerformed (ActionEvent e)
77                 {
78                     showAddView();
79                 }
80             });
81
82     }
83 }
```

```
sd copy lefted - Apache Netbeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
default config - HotelView.java HotelDisplayView.java HotelExitView.java HotelView.java HotelList.java Result.java PatientDisplayView.java PatientAddView.java
Project Sources Packages Tools Window Help
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History
34     public HotelView(HotelList model, HotelController controller)
35     {
36         super(model, controller);
37
38         addView = new HotelAddView(model, controller);
39         displayView = new HotelDisplayView(model, controller);
40         exitView = new HotelExitView(model, controller);
41         buttons.add(addButton);
42         buttons.add(displayButton);
43         buttons.add(exitButton);
44
45         // Set Buttons Font
46         addButton.setFont(View.btfFont);
47         displayButton.setFont(View.btfFont);
48         exitButton.setFont(View.btfFont);
49
50         setLayout(new BorderLayout());
51         add(buttons, BorderLayout.NORTH);
52         add(new JLabel(""), BorderLayout.CENTER);
53         cards.add(displayView, DISPLAY);
54         cards.add(addView, ADD);
55         //cards.add(exitView, EXIT);
56         add(cards, BorderLayout.SOUTH);
57
58         showAddView();
59
60         addButton.addActionListener(
61             new ActionListener()
62             {
63                 public void actionPerformed (ActionEvent e)
64                 {
65                     showAddView();
66                 }
67             });
68
69         displayButton.addActionListener(
70             new ActionListener()
71             {
72                 public void actionPerformed (ActionEvent e)
73                 {
74                     showAddView();
75                 }
76             });
77
78         exitButton.addActionListener(
79             new ActionListener()
80             {
81                 public void actionPerformed (ActionEvent e)
82                 {
83                     showAddView();
84                 }
85             });
86
87     }
88 }
```

```
1 package mdf.view;
2
3 import mdf.model.Patient;
4 import mdf.model.PatientList;
5 import mdf.controller.PatientController;
6 import java.awt.Container;
7 import javax.swing.JFrame;
8 import javax.swing.JPanel;
9 import javax.swing.JButton;
10 import javax.swing.JTextField;
11 import javax.swing.JScrollPane;
12 import javax.swing.JTextPane;
13 import java.awt.event.ActionListener;
14 import java.awt.event.ActionEvent;
```

```
15
16 public class PatientDisplayView extends JPanel implements ActionListener
17 {
18     // private JTextField nameField = new JTextField("FAX-001", SIZE);
19     private JTextField nameField = new JTextField("Patient Name", SIZE);
20     private JTextField idField = new JTextField("0", SIZE);
21     private JTextField countryField = new JTextField("Patient Country Of Origin", SIZE);
22     private JTextField dateOfAdmissionField = new JTextField("Date Of Admission", SIZE);
23     private JTextField emailField = new JTextField("E-mail", SIZE);
24     private JTextField mobileNumberField = new JTextField("Mobile Number", SIZE);
25
26     private JLabel numberLabel = new JLabel("Patient Number");
27     private JLabel nameLabel = new JLabel("Patient Name");
28     private JLabel idLabel = new JLabel("IC No.");
29     private JLabel countryOfOriginLabel = new JLabel("Country Of Origin");
30     private JLabel dateLabel = new JLabel("Date");
31     private JLabel emailLabel = new JLabel("E-mail");
32     private JLabel mobileNumberLabel = new JLabel("Mobile Number");
33     private JButton addButton = new JButton ("Add Patient");
34     private JButton resetButton = new JButton ("Reset Patient");
35
36     public PatientDisplayView(PatientList model, PatientController controller)
37     {
38         super(model, controller);
39     }
40 }
```

PatientAddView.java

```
1 package mdf.view;
2
3 import mdf.model.View;
4 import mdf.model.Patient;
5 import mdf.controller.PatientController;
6 import javax.swing.JFrame;
7 import javax.swing.JPanel;
8 import javax.swing.JButton;
9 import javax.swing.JTextField;
10 import javax.swing.JScrollPane;
11 import javax.swing.JTextPane;
12 import java.awt.Container;
13 import java.awt.event.ActionListener;
14 import java.awt.event.ActionEvent;
```

```
15
16 public class PatientAddView extends View implements ActionListener
17 {
18     // private JTextField numberField = new JTextField("FAX-001", SIZE);
19     private JTextField nameField = new JTextField("Patient Name", SIZE);
20     private JTextField idField = new JTextField("0", SIZE);
21     private JTextField countryField = new JTextField("Patient Country Of Origin", SIZE);
22     private JTextField dateOfAdmissionField = new JTextField("Date Of Admission", SIZE);
23     private JTextField emailField = new JTextField("E-mail", SIZE);
24     private JTextField mobileNumberField = new JTextField("Mobile Number", SIZE);
25
26     private JLabel numberLabel = new JLabel("Patient Number");
27     private JLabel nameLabel = new JLabel("Patient Name");
28     private JLabel idLabel = new JLabel("IC No.");
29     private JLabel countryOfOriginLabel = new JLabel("Country Of Origin");
30     private JLabel dateLabel = new JLabel("Date");
31     private JLabel emailLabel = new JLabel("E-mail");
32     private JLabel mobileNumberLabel = new JLabel("Mobile Number");
33     private JButton addButton = new JButton ("Add Patient");
34     private JButton resetButton = new JButton ("Reset Patient");
35
36     public PatientAddView(PatientList model, PatientController controller)
37     {
38         super(model, controller);
39     }
40 }
```

```
oif copy lefted - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Source History Project Files Recent Servers Tools Help
private JLabel nameLabel = new JLabel("Patient Name");
private JLabel idNoLabel = new JLabel("IC No.");
private JLabel countryOfOriginLabel = new JLabel("Country Of Origin");
private JLabel dateOfAdmissionLabel = new JLabel("Date Of Admission");
private JLabel emailLabel = new JLabel("Email");
private JLabel mobileNumberLabel = new JLabel("Mobile Number");
private JButton addButton = new JButton("Add Patient");
private JButton resetButton = new JButton("Reset Patient");

public PatientAddView(PatientList model, PatientController controller)
{
    super(model, controller);
    setLayout(new GridLayout(0,2));
    nameLabel.add(nameField);
    add(nameLabel); add(nameField);
    add(idNoLabel); add(idNoField);
    add(countryOfOriginLabel); add(countryOfOriginField);
    add(dateOfAdmissionLabel); add(dateOfAdmissionField);
    add(emailLabel); add(emailField);
    add(mobileNumberLabel); add(mobileNumberField);
    add(addButton); add(resetButton);
    resetButton.addActionListener(
        new ActionListener () {
            public void actionPerformed (ActionEvent e)
            {
                //nameField.setText("");
                nameField.setText(" ");
                idNoField.setText(" ");
                countryOfOriginField.setText(" ");
                dateOfAdmissionField.setText(" ");
                emailField.setText(" ");
                mobileNumberField.setText(" ");
            }
        });
    addButton.addActionListener(
        new ActionListener () {
            public void actionPerformed (ActionEvent e)
            {
                //Patient number = " ";
                String name = nameField.getText();
                String idNo = idNoField.getText();
                String countryOfOrigin = countryOfOriginField.getText();
                String dateOfAdmission = dateOfAdmissionField.getText();
                String email = emailField.getText();
                String mobileNumber = mobileNumberField.getText();
                Patient p = new Patient();
                p.setNumber(mobileNumber);
                p.setName(name);
                p.setIdNo(idNo);
                p.setCountryOfOrigin(countryOfOrigin);
                p.setDateOfAdmission(dateOfAdmission);
                p.setEmail(email);
                p.setMobileNumber(mobileNumber);
                PatientController controller = (PatientController) getController();
                controller.addPatient(p);
                System.out.println("**p** added to patientList");
            }
        });
}
```

```
oif copy lefted - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Source History Project Files Recent Servers Tools Help
private JLabel nameLabel = new JLabel("Patient Name");
private JLabel idNoLabel = new JLabel("IC No.");
private JLabel countryOfOriginLabel = new JLabel("Country Of Origin");
private JLabel dateOfAdmissionLabel = new JLabel("Date Of Admission");
private JLabel emailLabel = new JLabel("Email");
private JLabel mobileNumberLabel = new JLabel("Mobile Number");
private JButton addButton = new JButton("Add Patient");
private JButton resetButton = new JButton("Reset Patient");

public PatientAddView(PatientList model, PatientController controller)
{
    super(model, controller);
    setLayout(new GridLayout(0,2));
    nameLabel.add(nameField);
    add(nameLabel); add(nameField);
    add(idNoLabel); add(idNoField);
    add(countryOfOriginLabel); add(countryOfOriginField);
    add(dateOfAdmissionLabel); add(dateOfAdmissionField);
    add(emailLabel); add(emailField);
    add(mobileNumberLabel); add(mobileNumberField);
    add(addButton); add(resetButton);
    resetButton.addActionListener(
        new ActionListener () {
            public void actionPerformed (ActionEvent e)
            {
                //nameField.setText("");
                nameField.setText(" ");
                idNoField.setText(" ");
                countryOfOriginField.setText(" ");
                dateOfAdmissionField.setText(" ");
                emailField.setText(" ");
                mobileNumberField.setText(" ");
            }
        });
    addButton.addActionListener(
        new ActionListener () {
            public void actionPerformed (ActionEvent e)
            {
                //Patient number = " ";
                String name = nameField.getText();
                String idNo = idNoField.getText();
                String countryOfOrigin = countryOfOriginField.getText();
                String dateOfAdmission = dateOfAdmissionField.getText();
                String email = emailField.getText();
                String mobileNumber = mobileNumberField.getText();
                Patient p = new Patient();
                p.setNumber(mobileNumber);
                p.setName(name);
                p.setIdNo(idNo);
                p.setCountryOfOrigin(countryOfOrigin);
                p.setDateOfAdmission(dateOfAdmission);
                p.setEmail(email);
                p.setMobileNumber(mobileNumber);
                PatientController controller = (PatientController) getController();
                controller.addPatient(p);
                System.out.println("**p** added to patientList");
            }
        });
}
```

PatientDisplayView.java

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Title Bar:** File copy lefted - Apache NetBeans IDE 12.0
- Toolbar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Project Explorer:** Shows a project named "soft copy lefted" with packages like "soft", "soft.model", "soft.controller", "soft.view", and "soft.util".
- Code Editor:** The main window displays the code for `PatientDisplayView.java`. The code is a Java Swing application for displaying patient information. It imports various Java Swing components and defines a `PatientDisplayView` class extending `View`. The class constructor takes a `PatientList` model and a `PatientController` controller. It sets up a `JPanel` content pane, adds labels for patient number, name, address, etc., and a `JTextfield` for count. It also defines a `displayContents()` method.
- Status Bar:** Shows the current file as `soft.copy.lefted/run`, the status as `running...`, the time as 12:31 pm, the date as 8/6/2021, and the system as ING Windows [CPU].

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Toolbar:** Run, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Search Bar:** Search (Ctrl+F) at the top right.
- Project Explorer (Left):** Shows the project structure with packages like `com.saf`, `saf copy-lefted`, and `com.saf`. Under `com.saf`, there are sub-packages `model`, `controller`, `view`, and `util`. The `view` package contains files such as `HotelDisplayView.java`, `HotelEditView.java`, `HotelList.java`, `PatientAddView.java`, `PatientDisplayView.java`, `PatientEditView.java`, and `PatientList.java`.
- Code Editor (Right):** Displays the Java code for `PatientDisplayView.java`. The code implements the `View` interface and extends `JPanel`. It contains methods for displaying patient contents and handling button actions. A `PatModel` object is used to get patient data.
- Bottom Status Bar:** Shows the current file (`saf copy-lefted (run)`), memory usage (19.5M), and system status (28.3°C, Mostly cloudy).

Apache NetBeans IDE 12.0

```
public void addContents(JPanel contents)
{
    tf = new JTextField(SIZE);
    tf.setText(p.getFirstName());
    contents.add(tf);

    tf = new JTextField(SIZE);
    tf.setText(p.getLastName());
    contents.add(tf);

    tf = new JTextField(SIZE);
    tf.setText(p.getMiddleName());
    contents.add(tf);

    tf = new JTextField(SIZE);
    tf.setText(p.getEmail());
    contents.add(tf);

    tf = new JTextField(SIZE);
    tf.setText(p.getMobileNumber());
    contents.add(tf);

    add(contents, BorderLayout.NORTH);
}

public void displayCount()
{
    countTextField.setText(""+getModel().getCount());
}

public void update(String messageId)
{
    remove(contents);
    displayContents();
    displayCount();
}

JFrame topFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
topFrame.pack();
topFrame.repaint();
}
```

Output Search Results

type here to search

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Project Navigator

Source Packages

src

- soft copy khaled
- src/main/java
- src/test/java
- src/main/resources
- src/test/resources
- src/main/resources/META-INF
- src/test/resources/META-INF
- src/main/resources/ICDFApp.properties
- src/test/resources/ICDFApp.properties
- src/main/resources/softcopy.khaled
- src/test/resources/softcopy.khaled
- src/main/java/soft/controller
- src/main/java/soft/model
- src/main/java/soft/view
- src/main/java/soft/test
- src/main/java/TestRunner
- src/main/java/soft
- src/main/java/soft/controller/HotelController.java
- src/main/java/soft/controller/PatientController.java
- src/main/java/soft/controller/RoomsController.java
- src/main/java/soft/model/Hotel.java
- src/main/java/soft/model/HotelList.java
- src/main/java/soft/model/Model.java
- src/main/java/soft/model/Patient.java
- src/main/java/soft/model/PatientList.java
- src/main/java/soft/model/Rooms.java
- src/main/java/soft/view/HotelAddView.java
- src/main/java/soft/view/HotelDisplayView.java
- src/main/java/soft/view/HotelEditView.java
- src/main/java/soft/view/PatientAddView.java
- src/main/java/soft/view/PatientDisplayView.java
- src/main/java/soft/view/PatientEditView.java
- src/main/java/soft/view/View.java
- src/main/java/TestRunnerTest.java
- src/main/java/soft/test/TestRunnerTest.java
- src/main/java/soft/controller/TestRunnerController.java
- src/main/java/soft/model/TestRunnerModel.java
- src/main/java/soft/view/TestRunnerView.java
- src/main/java/soft/controller/TestRunnerController.java
- src/main/java/soft/model/TestRunnerModel.java
- src/main/java/soft/view/TestRunnerView.java

Output Search Results

type here to search

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Project Navigator

Source Packages

src

- soft copy khaled
- src/main/java
- src/test/java
- src/main/resources
- src/test/resources
- src/main/resources/META-INF
- src/test/resources/META-INF
- src/main/resources/ICDFApp.properties
- src/test/resources/ICDFApp.properties
- src/main/resources/softcopy.khaled
- src/test/resources/softcopy.khaled
- src/main/java/soft/controller
- src/main/java/soft/model
- src/main/java/soft/view
- src/main/java/soft/test
- src/main/java/TestRunner
- src/main/java/soft
- src/main/java/soft/controller/HotelController.java
- src/main/java/soft/controller/PatientController.java
- src/main/java/soft/controller/RoomsController.java
- src/main/java/soft/model/Hotel.java
- src/main/java/soft/model/HotelList.java
- src/main/java/soft/model/Model.java
- src/main/java/soft/model/Patient.java
- src/main/java/soft/model/PatientList.java
- src/main/java/soft/model/Rooms.java
- src/main/java/soft/view/HotelAddView.java
- src/main/java/soft/view/HotelDisplayView.java
- src/main/java/soft/view/HotelEditView.java
- src/main/java/soft/view/PatientAddView.java
- src/main/java/soft/view/PatientDisplayView.java
- src/main/java/soft/view/PatientEditView.java
- src/main/java/soft/view/View.java
- src/main/java/TestRunnerTest.java
- src/main/java/soft/test/TestRunnerTest.java
- src/main/java/soft/controller/TestRunnerController.java
- src/main/java/soft/model/TestRunnerModel.java
- src/main/java/soft/view/TestRunnerView.java

Output Search Results

type here to search

PatientView.java

Apache NetBeans IDE 12.0

```
public class PatientView extends View
{
    private JButton addButton = new JButton ("Add Panel");
    private JButton displayButton = new JButton ("Display Panel");
    private PatientAddView addView;
    private PatientDisplayView displayView;
    //private PatientExitView exitView;
    private JPanel cards = new JPanel(new CardLayout());
    private JPanel buttons = new JPanel();
    private static final String DISPLAY = "DISPLAY";
    //private static final String EXIT = "EXIT";
    public PatientView(PatientList model, PatientController controller)
    {
        super (model, controller);
        addView = new PatientAddView(model, controller);
        displayView = new PatientDisplayView(model, controller);
        //exitView = new PatientExitView(model, controller);
        buttons = new JPanel();
        buttons.add(displayButton);
        //buttons.add(exitButton);
        // Set Button Font
        addButton.setFont(View.buttonFont);
        displayButton.setFont(View.buttonFont);
        exitButton.setFont(View.buttonFont);
    }
}
```

Output Search Results

type here to search

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Project Navigator

Source Packages

src

- soft copy khaled
- src/main/java
- src/test/java
- src/main/resources
- src/test/resources
- src/main/resources/META-INF
- src/test/resources/META-INF
- src/main/resources/ICDFApp.properties
- src/test/resources/ICDFApp.properties
- src/main/resources/softcopy.khaled
- src/test/resources/softcopy.khaled
- src/main/java/soft/controller
- src/main/java/soft/model
- src/main/java/soft/view
- src/main/java/soft/test
- src/main/java/TestRunner
- src/main/java/soft
- src/main/java/soft/controller/HotelController.java
- src/main/java/soft/controller/PatientController.java
- src/main/java/soft/controller/RoomsController.java
- src/main/java/soft/model/Hotel.java
- src/main/java/soft/model/HotelList.java
- src/main/java/soft/model/Model.java
- src/main/java/soft/model/Patient.java
- src/main/java/soft/model/PatientList.java
- src/main/java/soft/model/Rooms.java
- src/main/java/soft/view/HotelAddView.java
- src/main/java/soft/view/HotelDisplayView.java
- src/main/java/soft/view/HotelEditView.java
- src/main/java/soft/view/PatientAddView.java
- src/main/java/soft/view/PatientDisplayView.java
- src/main/java/soft/view/PatientEditView.java
- src/main/java/soft/view/View.java
- src/main/java/TestRunnerTest.java
- src/main/java/soft/test/TestRunnerTest.java
- src/main/java/soft/controller/TestRunnerController.java
- src/main/java/soft/model/TestRunnerModel.java
- src/main/java/soft/view/TestRunnerView.java

Output Search Results

type here to search

sdf copy lefted - Apache NetBeans IDE 12.0

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config>
```

Source PatientDisplayView.java PatientAddView.java SDFAppl.java TestRunnerTestJava PatientListTest.java HotelListTest.java View.java PatientNew.java autogenerated.java

Navigation

Project

src

- mavenproject1
- sdf copy lefted
- Source Packages
- sdff
- SDFAppl.java
- Controller.java
- HotelController.java
- HotelListController.java
- Rooms.java
- sdff.model
- Hotel.java
- HotelList.java
- Patient.java
- PatientList.java
- autogenerated.java
- sdff.test
- sdff.all
- sdff.view
- HotelAddView.java
- HotelDisplayView.java
- HotelListView.java
- PatientAddView.java
- PatientDisplayView.java
- PatientList.java
- View.java
- Test Packages
- sdff.model
- HotelListTest.java
- PatientListTest.java
- sdff.test
- TestRunnerTest.java
- Libraries
- JDK 16 (Default)
- Test Libraries

Source PatientDisplayView.java

```
31 // Set Button Font
32 displayButton.setFont(View.btnFont);
33 displayButton.setFont(View.btnFont);
34
35 setLayout(new BorderLayout());
36 AddButtons, BorderLayout.NORTH);
37 add(new JLabel(""), BorderLayout.CENTER);
38 cards.add(displayView, DISPLAY);
39 cards.add(addView, ADD);
40 />add(buttonView, EDIT);
41 add(cards, BorderLayout.CENTER);
42
43 showAddView();
44
45 addButton.addActionListener(
46     new ActionListener () {
47         public void actionPerformed (ActionEvent e)
48         {
49             showAddView();
50         }
51     });
52
53 displayButton.addActionListener(
54     new ActionListener () {
55         public void actionPerformed (ActionEvent e)
56         {
57             showDisplayView();
58         }
59     });
60
61 }
62
63 private void showAddView()
64 {
65 }
```

Output Search Results

32°C Mostly cloudy 14:1 ING Windows (EN) 32°C Mostly cloudy 14:1 ING Windows (EN) 8/6/2021

sdf copy lefted - Apache NetBeans IDE 12.0

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config>
```

Source PatientDisplayView.java PatientAddView.java SDFAppl.java TestRunnerTestJava PatientListTest.java HotelListTest.java View.java PatientNew.java autogenerated.java

Navigation

Project

src

- mavenproject1
- sdf copy lefted
- Source Packages
- sdff
- SDFAppl.java
- Controller.java
- HotelController.java
- HotelListController.java
- Rooms.java
- sdff.model
- Hotel.java
- HotelList.java
- Patient.java
- PatientList.java
- autogenerated.java
- sdff.test
- sdff.all
- sdff.view
- HotelAddView.java
- HotelDisplayView.java
- HotelListView.java
- PatientAddView.java
- PatientDisplayView.java
- PatientList.java
- View.java
- Test Packages
- sdff.model
- HotelListTest.java
- PatientListTest.java
- sdff.test
- TestRunnerTest.java
- Libraries
- JDK 16 (Default)
- Test Libraries

Source PatientDisplayView.java

```
61 displayButton.addActionListener(
62     new ActionListener () {
63         public void actionPerformed (ActionEvent e)
64         {
65             showDisplayView();
66         }
67     });
68
69 }
70
71 private void showAddView()
72 {
73     CardLayout cl = (CardLayout)(cards.getLayout());
74     cl.show(cards, ADD);
75 }
76
77 private void showDisplayView()
78 {
79     CardLayout cl = (CardLayout)(cards.getLayout());
80     cl.show(cards, DISPLAY);
81 }
```

Output Search Results

32°C Mostly cloudy 14:1 ING Windows (EN) 32°C Mostly cloudy 14:1 ING Windows (EN) 8/6/2021

View.java

```
package sdf.view;

import sdf.model.Model;
import sdf.controller.Controller;
import sdf.util.Observer;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Font;

public class View extends JPanel implements Observer {

    private Model model;
    private Controller controller;
    private JFrame frame;

    /**
     * This attribute is to be used as a default font through the application
     * frame. True
     * font, a font with "Courier New", bold, size is 24
     */
    public static Font generalFont = new Font("Courier New", Font.BOLD, 24);

    /**
     * This attribute is to be used as a default font for all "Buttons"
     * through the application
     * frame. True
     * font, a font with "Courier New", Italic, size is 24
     */
    public static Font btnFont = new Font("Courier New", Font.ITALIC|Font.BOLD, 24);

    /**
     * This attribute is to be used as a default font for all "Labels"
     * through the application
     * frame. True
     * font, a font with "Courier New", Italic, size is 24
     */
    public static Font lblFont = new Font("Courier New", Font.BOLD, 24);

    /**
     * This attribute is to be used as a default font for all "TextFields"
     * through the application
     * frame. True
     * font, a font with "Arial", Plain, size is 24
     */
    public static Font fieldFont = new Font("Arial", Font.PLAIN, 24);

    /**
     * This attribute is to be used as a default font for all "Tables"
     * through the application
     * frame. True
     * font, a font with "Arial", Plain, size is 24
     */
    public static Font tableFont = new Font("Arial", Font.PLAIN, 24);

    /**
     * defaul constructor
     * frame. True
     * font, a new object with null model and controller
     */
    public View() {
    }

    /**
     * defaul constructor
     * frame. True
     * font, a new object with null model and controller
     */
    public View(Model model, Controller controller) {
        this.model = model;
        this.controller = controller;
    }
}
```

```
package sdf.view;

import sdf.model.Model;
import sdf.controller.Controller;
import sdf.util.Observer;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Font;

public class View extends JPanel implements Observer {

    private Model model;
    private Controller controller;
    private JFrame frame;

    /**
     * This attribute is to be used as a default font for all "Labels"
     * through the application
     * frame. True
     * font, a font with "Courier New", Italic, size is 24
     */
    public static Font generalFont = new Font("Courier New", Font.ITALIC, 24);

    /**
     * This attribute is to be used as a default font for all "Buttons"
     * through the application
     * frame. True
     * font, a font with "Courier New", Italic, size is 24
     */
    public static Font btnFont = new Font("Courier New", Font.ITALIC|Font.BOLD, 24);

    /**
     * This attribute is to be used as a default font for all "Labels"
     * through the application
     * frame. True
     * font, a font with "Arial", Plain, size is 24
     */
    public static Font lblFont = new Font("Arial", Font.PLAIN, 24);

    /**
     * This attribute is to be used as a default font for all "TextFields"
     * through the application
     * frame. True
     * font, a font with "Arial", Plain, size is 24
     */
    public static Font fieldFont = new Font("Arial", Font.PLAIN, 24);

    /**
     * This attribute is to be used as a default font for all "Tables"
     * through the application
     * frame. True
     * font, a font with "Arial", Plain, size is 24
     */
    public static Font tableFont = new Font("Arial", Font.PLAIN, 24);

    /**
     * defaul constructor
     * frame. True
     * font, a new object with null model and controller
     */
    public View() {
    }

    /**
     * defaul constructor
     * frame. True
     * font, a new object with null model and controller
     */
    public View(Model model, Controller controller) {
        this.model = model;
        this.controller = controller;
    }
}
```

```
oaf copy lefted - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> PatientDisplayView.java PatientAddView.java SOFApp.java TestRunnerTest.java PatientListTest.java HotelListTest.java View.java PatientView.java autogenerated.java
Search (Ctrl+F) X
Navigation Files Services... Source History Tools Window Help
PatientDisplayView.java PatientAddView.java SOFApp.java TestRunnerTest.java PatientListTest.java HotelListTest.java View.java PatientView.java autogenerated.java
Source Packages
oaf.copy.lefted
  oaf.controller
    Controller.java
    HotelController.java
    RoomController.java
  oaf.model
    Hotel.java
    HotelList.java
    Room.java
    Patient.java
    PatientList.java
    autogenerated.java
  oaf.test
  oaf.view
    HotelAddView.java
    HotelDisplayView.java
    HotelListView.java
    PatientAddView.java
    PatientDisplayView.java
    PatientListView.java
    View.java
  Test Packages
  Test Model
    HotelListTest.java
    PatientListTest.java
  Test Utilities
    TestRunnerTest.java
  Libraries
  JXK 16 (Default)
  Test Libraries
PatientDisplayView.java
1  /**
2  * This class represents the view for displaying patient information.
3  */
4
5  package oaf.copy.lefted;
6
7  import oaf.util.Observable;
8  import oaf.util.Observer;
9
10 import javax.swing.JFrame;
11
12 /**
13  * Spans model
14  *   The model for this view
15  * Spans controller
16  *   The controller for this view
17  * Spans frame
18  *   Spans the model and controller are initialized
19  */
20 public View(Model model, Controller controller)
21 {
22     this.model = model;
23     model.register(this);
24     this.controller = controller;
25     controller.setView(this);
26 }
27
28 /**
29  * Return the model of this view
30  * Spans, return the model of this view
31  */
32 public Model getModel()
33 {
34     return model;
35 }
36
37 /**
38  * Return the frame this view resides
39  * Spans, return the frame this view resides
40  */
41 public JFrame getFrame()
42 {
43     return frame;
44 }
45
46 /**
47  * Spans frame
48  *   The frame this view resides
49  * Spans, the instance variable frame is updated
50  */
51 private JFrame frame;
52
53 /**
54  * Return the controller of this view
55  * Spans, return the controller of this view
56  */
57 public Controller getController()
58 {
59     return controller;
60 }
61
62 /**
63  * This is the call back method from oaf.util.Observable
64  * subClass handling the event by overriding this method
65  * Spans message
66  *   A message id sent from the Observable
67  *   For which the Observer understands
68  *   Spans. Observable executes notifyObserver() method
69  *   after certain event occurs
70  * Spans, the event is handled
71  */
72 public void update(String messageId)
73 {
74     // Override this method in the subclass
75 }
```

```
oaf copy lefted - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> PatientDisplayView.java PatientAddView.java SOFApp.java TestRunnerTest.java PatientListTest.java HotelListTest.java View.java PatientView.java autogenerated.java
Search (Ctrl+F) X
Navigation Files Services... Source History Tools Window Help
PatientDisplayView.java PatientAddView.java SOFApp.java TestRunnerTest.java PatientListTest.java HotelListTest.java View.java PatientView.java autogenerated.java
Source Packages
oaf.copy.lefted
  oaf.controller
    Controller.java
    HotelController.java
    RoomController.java
  oaf.model
    Hotel.java
    HotelList.java
    Room.java
    Patient.java
    PatientList.java
    autogenerated.java
  oaf.test
  oaf.view
    HotelAddView.java
    HotelDisplayView.java
    HotelListView.java
    PatientAddView.java
    PatientDisplayView.java
    PatientListView.java
    View.java
  Test Packages
  Test Model
    HotelListTest.java
    PatientListTest.java
  Test Utilities
    TestRunnerTest.java
  Libraries
  JXK 16 (Default)
  Test Libraries
PatientDisplayView.java
1  /**
2  * This class represents the view for displaying patient information.
3  */
4
5  package oaf.copy.lefted;
6
7  import oaf.util.Observable;
8  import oaf.util.Observer;
9
10 import javax.swing.JFrame;
11
12 /**
13  * Spans model
14  *   The model for this view
15  * Spans controller
16  *   The controller for this view
17  * Spans frame
18  *   Spans the model and controller are initialized
19  */
20 public View(Model model, Controller controller)
21 {
22     this.model = model;
23     model.register(this);
24     this.controller = controller;
25     controller.setView(this);
26 }
27
28 /**
29  * Return the model of this view
30  * Spans, return the model of this view
31  */
32 public Model getModel()
33 {
34     return model;
35 }
36
37 /**
38  * Return the frame this view resides
39  * Spans, return the frame this view resides
40  */
41 public JFrame getFrame()
42 {
43     return frame;
44 }
45
46 /**
47  * Spans frame
48  *   The frame this view resides
49  * Spans, the instance variable frame is updated
50  */
51 private JFrame frame;
52
53 /**
54  * Return the controller of this view
55  * Spans, return the controller of this view
56  */
57 public Controller getController()
58 {
59     return controller;
60 }
61
62 /**
63  * This is the call back method from oaf.util.Observable
64  * subClass handling the event by overriding this method
65  * Spans message
66  *   A message id sent from the Observable
67  *   For which the Observer understands
68  *   Spans. Observable executes notifyObserver() method
69  *   after certain event occurs
70  * Spans, the event is handled
71  */
72 public void update(String messageId)
73 {
74     // Override this method in the subclass
75 }
```

Controller

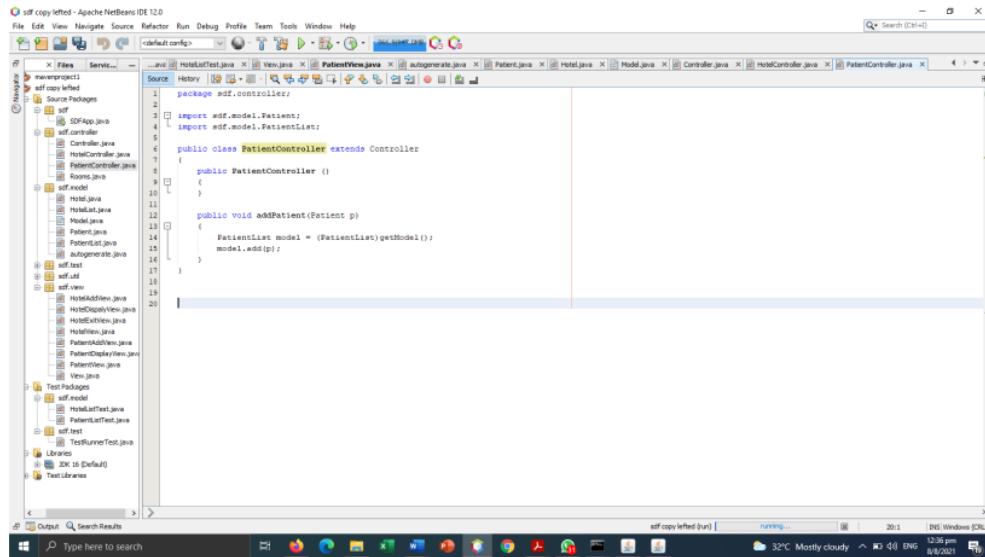
Controller.java

```
© sf copy left - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
codeconfig config ...
Source History ...
...-and TestRunnerTest.java PatientListTest.java HtmlListTest.java View.java PatientView.java autogenerate.java Patient.java Hotel.java Model.java Controller.java ...
Search (Ctrl+F) ...
...
1 package sdf.controller;
2 import sdf.model.Model;
3 import sdf.view.View;
4
5 public class Controller {
6     private Model model;
7     private View view;
8
9     public Controller () {
10 }
11
12     public void setModel(Model model)
13     {
14         this.model=model;
15     }
16
17     public void setView(View view)
18     {
19         this.view=view;
20     }
21
22     public Model getModel()
23     {
24         return model;
25     }
26
27     public View getView()
28     {
29         return view;
30     }
31
32 }
33
34
35
36
37 }
```

HotelController.java

```
© sf copy left - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
codeconfig config ...
Source History ...
...-and PatientTest.java HotelTest.java View.java PatientView.java autogenerate.java Patient.java Hotel.java Model.java Controller.java HotelController.java ...
Search (Ctrl+F) ...
...
1 /**
2  * To change this license header, choose License Headers in Project Properties,
3  * and open the template in the editor.
4  */
5 package sdf.controller;
6
7 import sdf.model.Hotel;
8 import sdf.model.HotelList;
9
10 public class HotelController extends Controller {
11     public HotelController ()
12     {
13     }
14
15     public void addHotel(Hotel h)
16     {
17         HotelList model = (HotelList)getModel();
18         model.add(h);
19     }
20
21 }
22
```

PatientController.java

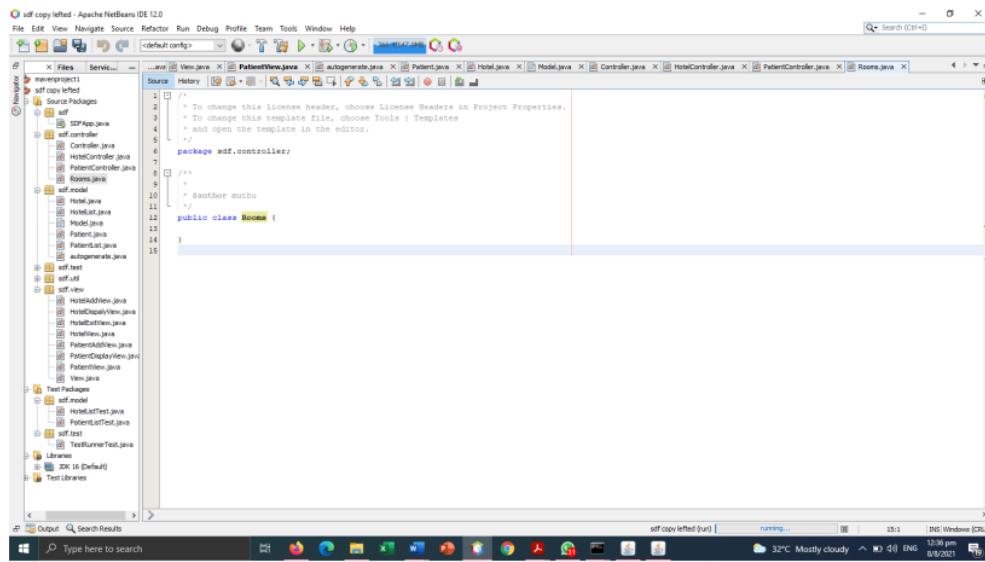


Apache NetBeans IDE 12.0

```
package sdf.controller;
import sdf.model.Patient;
import sdf.model.PatientList;
public class PatientController extends Controller
{
    public PatientController ()
    {
    }
    public void addPatient(Patient p)
    {
        PatientList model = (PatientList)getModel();
        model.add(p);
    }
}
```

The screenshot shows the Apache NetBeans IDE interface with the PatientController.java file open in the editor. The code defines a PatientController class that extends the Controller class. It contains a constructor and a method named addPatient that adds a patient to a PatientList model.

Rooms.java



Apache NetBeans IDE 12.0

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package sdf.controller;
public class Rooms {
}
```

The screenshot shows the Apache NetBeans IDE interface with the Rooms.java file open in the editor. The code defines a class named Rooms that is currently empty.

GUI- Patient Management

The screenshot shows a window titled "Patient Management". The interface includes tabs for "Patient Management" and "Hotel Management". Below the tabs are two buttons: "Add Panel" and "Display Panel". The main area contains six input fields arranged in two columns of three. The left column contains: "Patient Name", "IC No", "Country Of Origin", "Date Of Admission", "E-mail", and "Mobile Number". The right column contains: "Patient Name", "IC No", "Country Of Origin", "Date Of Admission", "E-mail", and "Mobile Number". At the bottom are two buttons: "Add Patient" and "Reset Patient".

GUI- Hotel Management

The screenshot shows a window titled "Patient Management". The interface includes tabs for "Patient Management" and "Hotel Management". Below the tabs are two buttons: "Add Panel" and "Display Panel". The main area contains four input fields arranged in two columns of two. The left column contains: "Hotel No", "Hotel Name", "Location Of Hotel", and "Number Of Rooms Available". The right column contains: "Hotel No", "Hotel Name", "Location Of Hotel", and "Number Of Rooms Available". At the bottom are two buttons: "Add hotel" and "Reset hotel".

REPORT

ORIGINALITY REPORT

0
%

SIMILARITY INDEX

0
%

INTERNET SOURCES

0
%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes Off

Exclude bibliography On

Exclude matches < 1%