Department of Computer Science & Engineering, SDMCET, Dharwad-2

AOOP Assignment Submission Report

[Submitted as part of CTA Assignment No-2]

Course:	Advanced Object-Oriented Programming	Course Code:	18UCSE508
Semester:	V	Division:	A

Submitted by:

USN:	2SD20CS128	Name:	Yogita B Joshi

1. Problem Definition:

Write a Java program to build the GUI application using JavaFX for the following requirements:

a) Read user name and password using appropriate JavaFX controls.

b) Validate the input. If user name and password are matched with the assumed values, then

display the welcome scene with proper text.

c) If user name and password don't match, then raise appropriate exception.

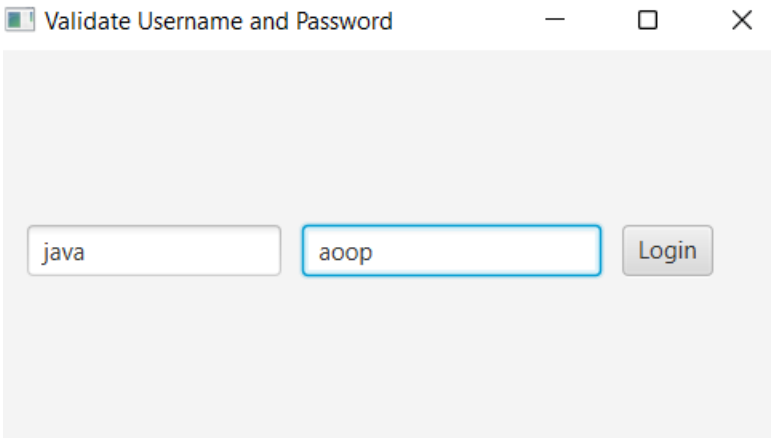
```
package application;

import javafx.application.*;
import javafx.scene.*;
import javafx.stage.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.event.*;
import javafx.geometry.*;


public class Ass1 extends Application {
	Label response1,response2;


	public static void main(String[] args) {
		launch(args);
	}

	// Override the start() method.
	public void start(Stage myStage)throws InvalidCredentialsException {

		// Give the stage a title.
```

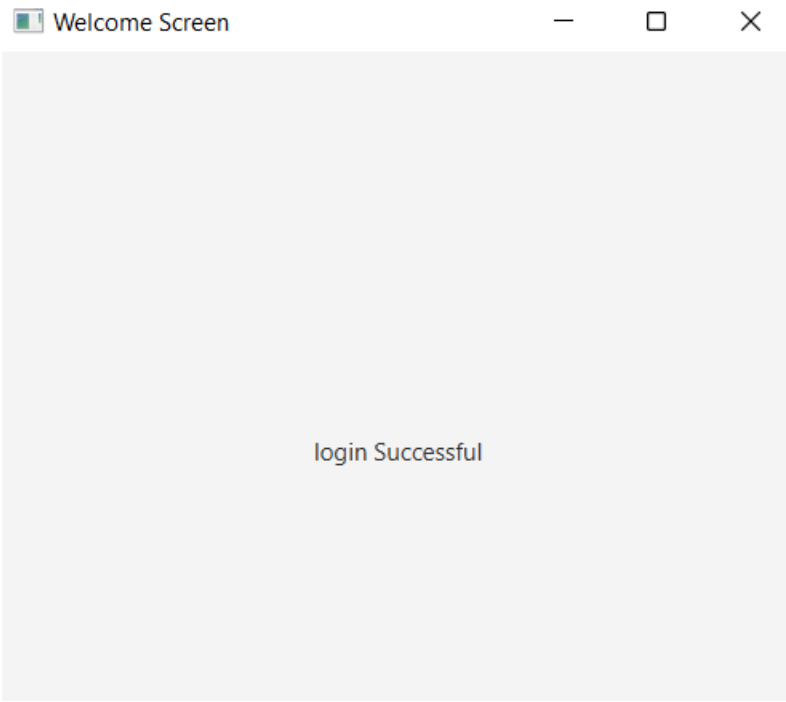
```
myStage.setTitle("Validate Username and Password");


FlowPane rootNode = new FlowPane(10, 20);

// Center the controls in the scene.
rootNode.setAlignment(Pos.CENTER);

// Create a scene.
Scene myScene = new Scene(rootNode, 400, 200);

// Set the scene on the stage.
myStage.setScene(myScene);


response1 = new Label("");

Button busername = new Button("Login");

response2= new Label("");


// Create a text field.
TextField tf1 = new TextField();
TextField tf2= new TextField();


tf1.setPromptText("Username");
tf2.setPromptText("Password");


tf1.setPrefColumnCount(10);
//tf2.setPrefColumnCount(30);

tf1.setOnAction(new EventHandler<ActionEvent>() {
	public void handle(ActionEvent ae) {
		response1.setText("UserName: " + tf1.getText());
	}
});

tf2.setOnAction(new EventHandler<ActionEvent>() {
```

```
                    public void handle(ActionEvent ae) {

                                response1.setText("Password: " + tf2.getText());
                    }
          });


          busernamе.setOnAction(new EventHandler<ActionEvent>() {
                    public void handle(ActionEvent ae) {
                              try {
                                if(tf1.getText().equals(null) || tf2.getText().equals(null)) {
                                          response1.setText("Enter the correct credentials");


                                }

                                else if(tf1.getText().equals("java") &&
tf2.getText().equals("aoop"))
                                {

                                  response1.setText("User Name: " + tf1.getText());
                                  response2.setText("password: " + tf2.getText());

                                  myStage.setTitle("Welcome Screen");



                                  FlowPane rootNode = new FlowPane(10, 10);


                                        rootNode.setAlignment(Pos.CENTER);
                                        Label login=new Label("login Successful");

                                        // Create a scene.
                                        Scene myScene = new Scene(rootNode, 400, 400);

                                        // Set the scene on the stage.
                                        myStage.setScene(myScene);

                                         rootNode.getChildren().addAll(login);
                                }
```

```
                    else
                    {
                        response1.setText("Enter a valid name");
                        response1.setText("Enter a valid password");
                        throw new InvalidCredentialsException();
                    }
                } catch(InvalidCredentialsException ce)
                    {
                        System.out.println("Enter correct credentials");
                        response1.setText(ce.toString());
                    }

                }
            });


            rootNode.getChildren().addAll(tf1,tf2, busernamе,response1,response2);


            myStage.show();
        }

public class InvalidCredentialsException extends Exception {


}
}
```

Test Case 1





2. Problem Definition:

Write a Java program to build the GUI application using JavaFX for the following requirements:

a) Create a Menu control to display the menu items: File, Edit & Help.
b) Create sub menus in the order: File → New, Open & Save. Edit → Cut, Copy & Paste.
Help → Help Centre, About Us
The program must use Mnemonics and Accelerators (wherever appropriate) to Menu Items.

```
package application;

//Demonstrate Menus
import javafx.application.*;
import javafx.scene.*;
import javafx.stage.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.event.*;

public class Ass2 extends Application {
	Label response;

	public static void main(String[] args) {
		// Start the JavaFX application by calling launch().
		launch(args);
	}

	// Override the start() method.
	public void start(Stage myStage) {

		// Give the stage a title.
		myStage.setTitle("Demonstrate Menus");

		// Use a BorderPane for the root node.
		BorderPane rootNode = new BorderPane();
```

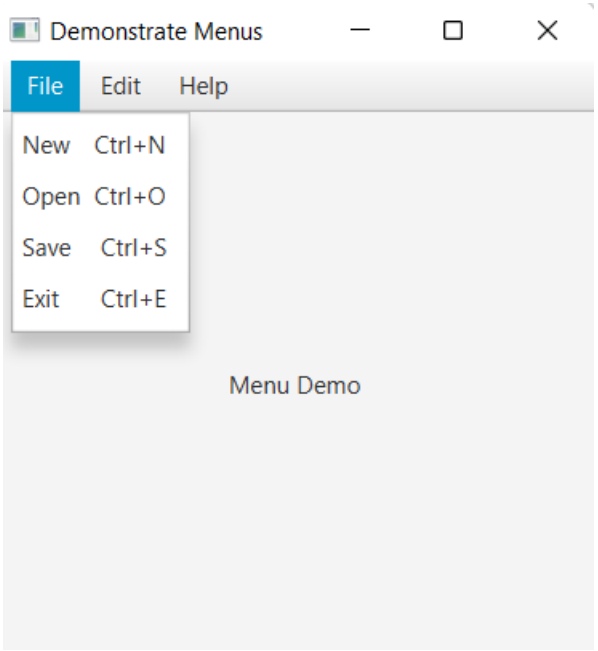
```
// Create a scene.
Scene myScene = new Scene(rootNode, 300, 300);

// Set the scene on the stage.
myStage.setScene(myScene);

// Create a label that will report the selection.
response = new Label("Menu Demo");

// Create the menu bar.
MenuBar mb = new MenuBar();

// Create the File menu.
Menu fileMenu = new Menu("File");  // new defines a mnemonic
MenuItem new1 = new MenuItem("New");
MenuItem open = new MenuItem("Open");
MenuItem save = new MenuItem("Save");
MenuItem exit = new MenuItem("Exit");

fileMenu.getItems().addAll(new1 ,open, save,exit);

// Turn on mnemonic
fileMenu.setMnemonicParsing(true);

// Add keyboard accelerators for the File menu.
new1.setAccelerator(KeyCombination.keyCombination("shortcut+N"));
//shortcut maps to ctrl
open.setAccelerator(KeyCombination.keyCombination("shortcut+O"));
save.setAccelerator(KeyCombination.keyCombination("shortcut+S"));
```

```
exit.setAccelerator(KeyCombination.keyCombination("shortcut+E"));
// Add File menu to the menu bar.
mb.getMenus().add(fileMenu);

// Create the Edit menu.
Menu editMenu = new Menu("Edit");
MenuItem cut = new MenuItem("Cut");
MenuItem copy = new MenuItem("Copy");
MenuItem paste = new MenuItem("Paste");

editMenu.getItems().addAll(cut,copy,paste);
mb.getMenus().add(editMenu);



cut.setAccelerator(KeyCombination.keyCombination("shortcut+X"));
copy.setAccelerator(KeyCombination.keyCombination("shortcut+C"));
paste.setAccelerator(KeyCombination.keyCombination("shortcut+V"));  //shortcut
maps to ctrl

// Create the Help menu.
Menu helpMenu = new Menu("Help");
MenuItem helpcentre = new MenuItem("Help Centre");
MenuItem aboutus = new MenuItem("About Us");
helpMenu.getItems().addAll(helpcentre,aboutus);

// Add Help menu to the menu bar.
mb.getMenus().add(helpMenu);
```

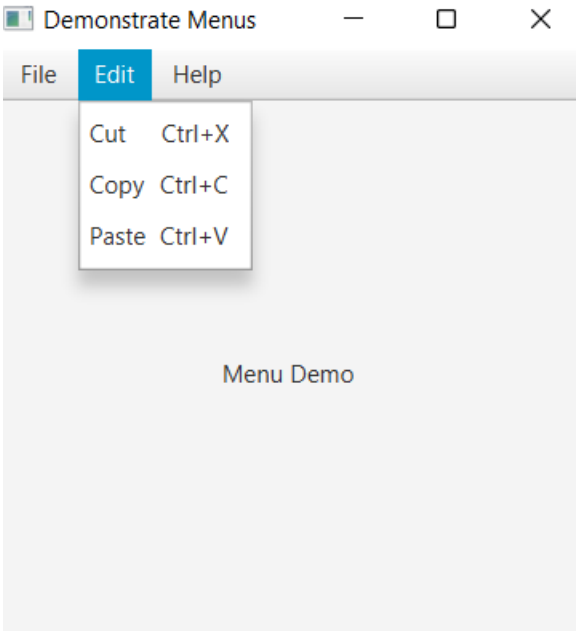
```
// Create one event handler that will handle menu action events.
EventHandler<ActionEvent> MEHandler = new EventHandler<ActionEvent>() {
        public void handle(ActionEvent ae) {
                String name = ((MenuItem) ae.getTarget()).getText();
                // If Exit is chosen, the program is terminated.
                if (name.equals("Exit"))
                        Platform.exit();
                response.setText(name + " selected");
        }
};

// Set action event handlers for the menu items.
open.setOnAction(MEHandler);
new1.setOnAction(MEHandler);
save.setOnAction(MEHandler);
exit.setOnAction(MEHandler);
cut.setOnAction(MEHandler);
copy.setOnAction(MEHandler);
paste.setOnAction(MEHandler);
helpcentre.setOnAction(MEHandler);
aboutus.setOnAction(MEHandler);


// Add the menu bar to the top of the border pane and
// the response label to the center position.
rootNode.setTop(mb);
rootNode.setCenter(response);

// Show the stage and its scene.
```

```
                myStage.show();
        }
}
```

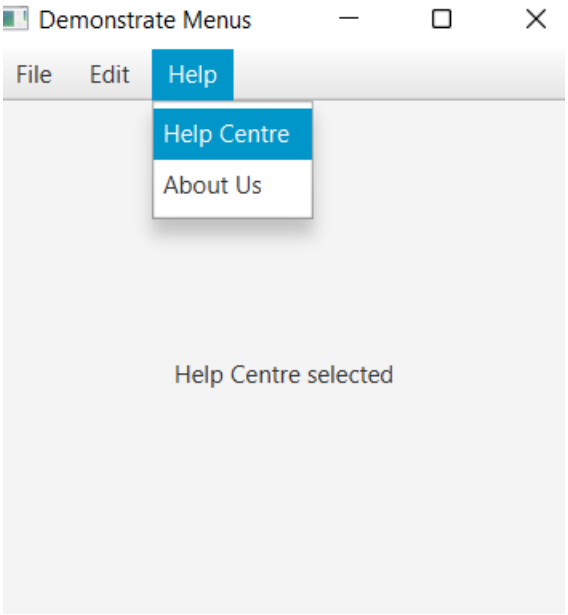
Test Case 1



Test Case 2



Test Case 3



3. Problem Definition:

Write a Java program to build the GUI application using JavaFX for the following requirements:

a) Create Context menu involving the menu items in the order: New & View.
b) Create sub menus for the above main context menu: New → File, Folder & Image.
View → Large, Medium & Small.
The context menu must be displayed on right-click of the mouse button.

```
package application;

//Demonstrate Menus
import javafx.application.*;
import javafx.scene.*;
import javafx.stage.*;
```

```
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.event.*;
import javafx.geometry.Pos;

public class Ass3 extends Application {
	Label response;

	public static void main(String[] args) {
		// Start the JavaFX application by calling launch().
		launch(args);
	}

	// Override the start() method.
	public void start(Stage myStage) {

		// Give the stage a title.
		myStage.setTitle("Demonstrate Menus");

		// Use a BorderPane for the root node.
		BorderPane rootNode = new BorderPane();

		// Create a scene.
		Scene myScene = new Scene(rootNode, 300, 300);

		// Set the scene on the stage.
		myStage.setScene(myScene);

		// Create a label that will report the selection.
		response = new Label("Menu Demo");

		// Create the menu bar.
		MenuBar mb = new MenuBar();

		// Create the File menu.
		Menu fileMenu = new Menu("File");
		MenuItem open = new MenuItem("Open");
		MenuItem close = new MenuItem("Close");
		MenuItem save = new MenuItem("Save");
		MenuItem exit = new MenuItem("Exit");
		fileMenu.getItems().addAll(open, close, save, new SeparatorMenuItem(), exit);
```

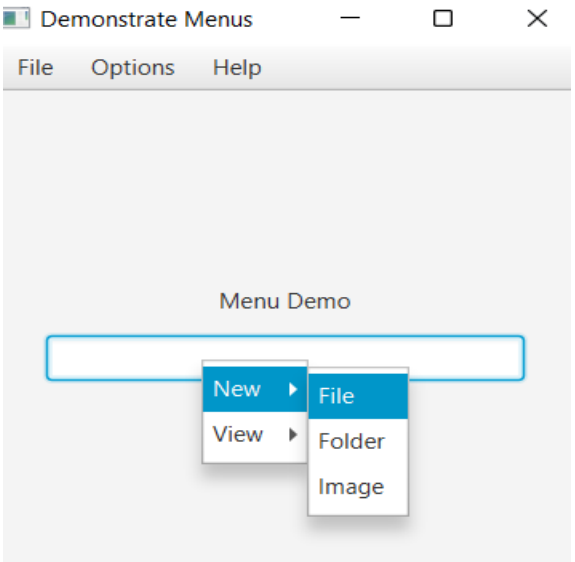
```
// Add File menu to the menu bar.
mb.getMenus().add(fileMenu);

// Create the Options menu.
Menu optionsMenu = new Menu("Options");

// Create the Colors sub-menu.
Menu colorsMenu = new Menu("Colors");
MenuItem red = new MenuItem("Red");
MenuItem green = new MenuItem("Green");
MenuItem blue = new MenuItem("Blue");

colorsMenu.getItems().addAll(red, green, blue);
optionsMenu.getItems().add(colorsMenu);

// Create the Priority sub-menu.
Menu priorityMenu = new Menu("Priority");
MenuItem high = new MenuItem("High");
MenuItem low = new MenuItem("Low");

priorityMenu.getItems().addAll(high, low);
optionsMenu.getItems().add(priorityMenu);

// Add a separator.
optionsMenu.getItems().add(new SeparatorMenuItem());

// Create the Reset menu item.
MenuItem reset = new MenuItem("Reset");
optionsMenu.getItems().add(reset);

// Add Options menu to the menu bar.
mb.getMenus().add(optionsMenu);

// Create the Help menu.
Menu helpMenu = new Menu("Help");
MenuItem about = new MenuItem("About");
helpMenu.getItems().add(about);

// Add Help menu to the menu bar.
```

```
        mb.getMenus().add(helpMenu);

        // Create the context menu items
        Menu newMenu = new Menu("New");
        MenuItem file = new MenuItem("File");
        MenuItem folder = new MenuItem("Folder");
        MenuItem image= new MenuItem("Image");


    newMenu.getItems().addAll(file,folder,image);


  //  final ContextMenu editMenu = new ContextMenu(newMenu);

        Menu viewMenu = new Menu("View");
        MenuItem large = new MenuItem("Large");
        MenuItem medium = new MenuItem("Medium");
        MenuItem small= new MenuItem("Small");


        viewMenu.getItems().addAll(large,medium,small);


        final ContextMenu editMenu= new ContextMenu(newMenu,viewMenu);

        // Create one event handler that will handle menu action events.
        EventHandler<ActionEvent> MEHandler = new EventHandler<ActionEvent>() {
            public void handle(ActionEvent ae) {
                String name = ((MenuItem) ae.getTarget()).getText();
                // If Exit is chosen, the program is terminated.
                if (name.equals("Exit"))
                    Platform.exit();
                response.setText(name + " selected");
            }
        };

        // Set action event handlers for the menu items.
        open.setOnAction(MEHandler);
        close.setOnAction(MEHandler);
        save.setOnAction(MEHandler);
        exit.setOnAction(MEHandler);
```

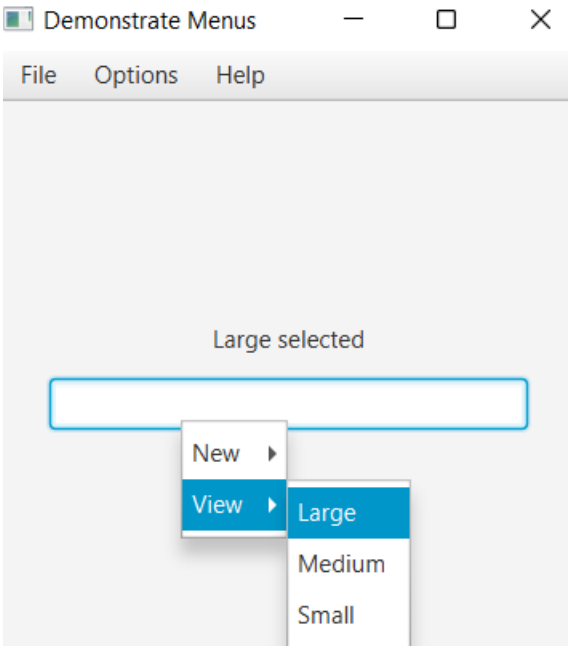
```
red.setOnAction(MEHandler);
green.setOnAction(MEHandler);
blue.setOnAction(MEHandler);
high.setOnAction(MEHandler);
low.setOnAction(MEHandler);
reset.setOnAction(MEHandler);
about.setOnAction(MEHandler);

file.setOnAction(MEHandler);
folder.setOnAction(MEHandler);
image.setOnAction(MEHandler);
large.setOnAction(MEHandler);
medium.setOnAction(MEHandler);
small.setOnAction(MEHandler);

// Create a text field and set its column width to 20.
TextField tf = new TextField();
tf.setPrefColumnCount(20);
//TextField tf1 = new TextField();
//tf.setPrefColumnCount(20);

// Add the context menu to the textfield.
tf.setContextMenu(editMenu);


// Add the menu bar to the top of the border pane and
// the response label to the center position.
rootNode.setTop(mb);

// Create a flow pane that will hold both the response
// label and the text field.
FlowPane fpRoot = new FlowPane(10, 10);

// Center the controls in the scene.
fpRoot.setAlignment(Pos.CENTER);

// Add both the label and the text field to the flow pane.
fpRoot.getChildren().addAll(response, tf);

// Add the flow pane to the center of the border layout.
rootNode.setCenter(fpRoot);
```

```
            // Show the stage and its scene.
            myStage.show();
        }
}
```

Test Case 3



Test Case 4



4. Problem Definition:

Write a JavaFX program that produces the following output when executed and displays Dialog Box.

```
package application;

import javafx.application.*;
import javafx.scene.*;
import javafx.stage.*;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.event.*;
import javafx.geometry.*;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.Alert.AlertType;

public class Ass4 extends Application {


	public static void main(String[] args) {

		launch(args);
	}

	public void start(Stage myStage) {

		myStage.setTitle("JavaFx Registration Form");

		//creating a Scene graph
		GridPane rootNode = new GridPane();

		Scene myScene = new Scene(rootNode,650,525);
		myStage.setScene(myScene);

		myStage.show();
```

```
Label heading = new Label("Employee Registration Form");



Label name =   new Label("Enter Your Name:");
Label gender = new Label("Select Your Gender:");
Label dob =    new Label("Enter Date of Birth:");
Label state =  new Label("Select Your State:");
Label qualification =  new Label("Select Your qualification:");
TextField tname = new TextField();


Button bregister = new Button("Register");
tname.setPromptText("Enter Your Name");
ToggleGroup tg = new ToggleGroup();
RadioButton r1 = new RadioButton("Male");
RadioButton r2 = new RadioButton("Female");


r1.setToggleGroup(tg);
r2.setToggleGroup(tg);

DatePicker dp = new DatePicker();


ObservableList<String> states = FXCollections.observableArrayList("Andhra
Pradesh","Arunachal Pradesh","Assam","Bihar","Chhattisgarh","Goa","Gujarat","Haryana",
"Himachal Pradesh","Karnataka");


ComboBox<String> cbState = new ComboBox<String>(states);
CheckBox c1 = new CheckBox("UG");
CheckBox c2 = new CheckBox("PG");
CheckBox c3 = new CheckBox("PhD");

//creating the HBoxes for each pair of information
HBox hb0 = new HBox(15);
HBox hb1 = new HBox(15);
HBox hb2 = new HBox(15);
HBox hb3 = new HBox(15);
HBox hb4 = new HBox(15);
```

```
HBox hb5 = new HBox(15);
HBox hb6 = new HBox(15);


rootNode.setHgap(15);
rootNode.setVgap(15);

//adding the components into each group of HBoxes
hb0.getChildren().add(heading);
hb1.getChildren().addAll(name,tname);
hb2.getChildren().addAll(gender,r1,r2);
hb3.getChildren().addAll(dob,dp);
hb4.getChildren().addAll(state,cbState);
hb5.getChildren().addAll(qualification,c1,c2,c3);
hb6.getChildren().add(bregister);

rootNode.setAlignment(Pos.TOP_CENTER);

//setting the HBoxes according to the requirement
hb0.setAlignment(Pos.TOP_CENTER);
hb6.setAlignment(Pos.BOTTOM_CENTER);


hb0.setPadding(new Insets(20,0,0,0));

rootNode.add(hb0,1,0,1,1);
rootNode.add(hb1,1,2,1,1);
rootNode.add(hb2,1,3,1,1);
rootNode.add(hb3,1,4,1,1);
rootNode.add(hb4,1,5,1,1);
rootNode.add(hb5,1,6,1,1);
rootNode.add(hb6,1,8,1,1);

//creating an Alert DialogueBox for the response
Alert alert = new Alert(AlertType.INFORMATION);

//setting up the title for the DialogueBox
alert.setTitle("Registration Successful");

//setting the info in the DialogueBox
alert.setHeaderText("Registration Status");
```

```
                alert.setContentText("Employee Registration is Successful!!");


                bregister.setOnAction(new EventHandler<ActionEvent>() {
                        public void handle(ActionEvent ae) {
                                alert.show();
                        }
                });
        }
}
```

Test Case 1