Lab 5: Swing and JavaFX

Objectives:

- Use Different layouts in Swing and JavaFX
- Learn GUI controls in Swing and JavaFX
- Learn about event handling and listener Interfaces

Programs:

1. **Program 1 School Management System**

   - react/Reactive.java

```java
package react;

import java.awt.Component;

public class Reactive<CompType extends Component, HookType> {
    public CompType comp;
    public HookType hook;
    Renderable<CompType, HookType> renderer;
    Clearable<CompType> clearer;


    public Reactive(CompType comp, HookType hook, Renderable<CompType, HookType> ren
            Clearable<CompType> clearable) {
        this.comp = comp;
        this.renderer = renderable;
        this.hook = hook;
        this.clearer = clearable;
        renderable.renderer(comp, hook);
    }
    public Reactive(Reactive<CompType,HookType> Rcomp, HookType hook, Renderable<Com
            Clearable<CompType> clearable) {
        this.comp = Rcomp.comp;
        this.renderer = renderable;
        this.hook = hook;
        this.clearer = clearable;
        renderable.renderer(comp, hook);
    }
    public void setRenderable(Renderable<CompType, HookType> renderable){
        this.renderer = renderable;
        renderable.renderer(comp, hook);
    }
    public void setClearer(Clearable<CompType> clearable) {
        this.clearer = clearable;
        renderer.renderer(comp, hook);
    }
    public void setHook(HookType newHook) {
        hook = newHook;
        clearer.clearer(comp);
        renderer.renderer(comp, hook);
        comp.repaint();
        comp.revalidate();
    }
```

```
}
```

- react/Clearable.java

```
package react;

import java.awt.Component;

public interface Clearable<CompType extends Component> {
    public void clearer(CompType comp);
}
```

- react/Renderable.java

```
package react;

import java.awt.Component;

public interface Renderable<CompType extends Component, HookType> {

    void renderer(CompType comp, HookType hook);
}
```

- RegistrationForm.java

```
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JTextField;
import javax.swing.JList;
import javax.swing.JOptionPane;
import react.Reactive;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Container;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;

import java.util.ArrayList;
import java.util.List;

import java.awt.FlowLayout;

class Arrangement {
    static public void allCenter(Container parent, Component child) {
        JPanel centerer = new JPanel(new GridBagLayout());
        centerer.add(child, new GridBagConstraints());
        parent.add(centerer);
    }

    static public void left(Container parent, Component child) {
        JPanel lefter = new JPanel(new FlowLayout(FlowLayout.LEFT));
```

```
        lefter.add(child);
        parent.add(lefter);
    }

    static public void right(Container parent, Component child) {
        JPanel righter = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        righter.add(child);
        parent.add(righter);
    }

    static public void center(Container parent, Component child) {
        JPanel centerer = new JPanel(new FlowLayout(FlowLayout.CENTER));
        centerer.add(child);
        parent.add(centerer);
    }

    static public void top(Container parent, Component child) {
        JPanel topper = new JPanel(new BorderLayout());
        topper.add(child, BorderLayout.NORTH);
        parent.add(topper);
    }

    static public void bottom(Container parent, Component child) {
        JPanel bottomer = new JPanel(new BorderLayout());
        bottomer.add(child, BorderLayout.SOUTH);
        parent.add(bottomer);
    }

    static public void fill(Container parent, Component child) {
        JPanel filler = new JPanel(new BorderLayout());
        filler.add(child, BorderLayout.CENTER);
        parent.add(filler);
    }

    static public void addEmpty(Container parent) {
        parent.add(new JPanel());
    }
}

class FormInput extends JPanel {
    public FormInput(String labelString, Component comp) {
        super(new GridLayout(1, 2, 20, 0));
        Arrangement.right(this, new JLabel(labelString));
        Arrangement.top(this, comp);
    }
}

class Student {
    public String id;
    public String name;
    public String gender;
    public List<String> courses;
    public String department;

    public Student(String id, String name, String gender, List<String> courses, Str
```

```
            this.id = id;
            this.name = name;
            this.gender = gender;
            this.courses = courses;
            this.department = department;
        }
    }

    class Register extends JFrame {

        JPanel formPanel;

        public Register(JFrame prevFrame, List<Student> students) {
            super("Register Frame");
            formPanel = new JPanel(new GridLayout(6, 3));
            JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
            JButton backButton = new JButton("Back");
            backButton.addActionListener(e -> {
                this.setVisible(false);
                prevFrame.setVisible(true);
            });
            topPanel.add(backButton);
            JPanel genderPanel = new JPanel(new GridLayout(2, 1));
            ButtonGroup genderGroup = new ButtonGroup();

            JRadioButton maleRadioButton = new JRadioButton("Male");
            maleRadioButton.setActionCommand("Male");
            genderGroup.add(maleRadioButton);

            JRadioButton femaleRadioButton = new JRadioButton("Female");
            femaleRadioButton.setActionCommand("Female");
            genderGroup.add(femaleRadioButton);

            genderPanel.add(maleRadioButton);
            genderPanel.add(femaleRadioButton);

            JPanel coursePanel = new JPanel(new GridLayout(3, 1));

            JCheckBox physicsCheckBox = new JCheckBox("Physics");
            JCheckBox dsaCheckBox = new JCheckBox("Economics");
            JCheckBox javaCheckBox = new JCheckBox("Java");

            coursePanel.add(physicsCheckBox);
            coursePanel.add(dsaCheckBox);
            coursePanel.add(javaCheckBox);

            JComboBox<String> departmentComboBox = new JComboBox<String>();

            departmentComboBox.addItem("Computer Science");
            departmentComboBox.addItem("Phyics");
            departmentComboBox.addItem("Ecconomics");

            JTextField idInput = new JTextField();

            JTextField studentInput = new JTextField();
```

```
addFormRow(new FormInput("Student Id", idInput));
addFormRow(new FormInput("Student Name", studentInput));
addFormRow(new FormInput("Gender", genderPanel));
addFormRow(new FormInput("Courses", coursePanel));
addFormRow(new FormInput("Department", departmentComboBox));

JButton submitBtn = new JButton("Submit");
submitBtn.addActionListener(e -> {
    // check if all inputs are empty
    if (idInput.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Error: Student Id is required",
                JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (studentInput.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Error: Student Name is required
                JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (genderGroup.getSelection() == null) {
        JOptionPane.showMessageDialog(this, "Error: Gender is required", "Er
        return;
    }
    if (!physicsCheckBox.isSelected() && !dsaCheckBox.isSelected() && !java
        JOptionPane.showMessageDialog(this, "Error: At least one course is r
                JOptionPane.ERROR_MESSAGE);
        return;
    }
    String name = studentInput.getText();
    String id = idInput.getText();
    String gender = genderGroup.getSelection().getActionCommand();
    List<String> courses = new ArrayList<String>();
    String department = (String) departmentComboBox.getSelectedItem();

    if (physicsCheckBox.isSelected()) {
        courses.add("Physics");
    }
    if (dsaCheckBox.isSelected()) {
        courses.add("DSA");
    }
    if (javaCheckBox.isSelected()) {
        courses.add("Java");
    }
    // print all info
    System.out.println("Name: " + name);
    System.out.println("Id: " + id);
    System.out.println("Gender " + gender);
    System.out.println("Courses: " + courses);
    System.out.println("Department: " + department);
    students.add(new Student(id, name, gender, courses, department));
    JOptionPane.showMessageDialog(this, "Student Registered Successfully", "
            JOptionPane.INFORMATION_MESSAGE);
});
```

```
            addFormRow(submitBtn);

            this.add(topPanel, BorderLayout.NORTH);
            Arrangement.top(this, formPanel);
        }

        void addFormRow(Component comp) {
            Arrangement.addEmpty(formPanel);
            Arrangement.fill(formPanel, comp);
            Arrangement.addEmpty(formPanel);
        }

    }

    class View extends JFrame {
        private int studentIdx;

        public View(JFrame prevFrame, List<Student> students) {
            super("View Frame");
            studentIdx = 0;
            Reactive<JPanel, Integer> studentSelector = new Reactive<JPanel, Integer>(ne
                    (comp, hook) -> {
                    }, (comp) -> {
                        comp.removeAll();
                    });

            Reactive<JPanel, Student> studentRenderer = new Reactive<JPanel, Student>(ne
                    students.size() > 0 ? students.get(0) : null, (comp, hook) -> {
                    }, (comp) -> {
                        comp.removeAll();
                    });
            studentRenderer.setRenderable((comp, hook) -> {
                if (hook != null) {
                    comp.setLayout(new GridLayout(6, 1, 10, 20));
                    comp.add(new JLabel("Id: " + hook.id));
                    Arrangement.fill(comp, new JLabel("Name: " + hook.name));
                    comp.add(new JLabel("Gender: " + hook.gender));
                    comp.add(new JList<String>(hook.courses.toArray(new String[hook.cour
                    comp.add(new JLabel("Department: " + hook.department));

                    JButton deleteBtn = new JButton("Delete");
                    deleteBtn.addActionListener(e -> {

                        int option = JOptionPane.showConfirmDialog(comp, "Delete?", "Del
                                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE)
                        if (option == JOptionPane.NO_OPTION) {
                            return;
                        }
                        if (option == JOptionPane.YES_OPTION) {
                            if (students.size() == 0) {
                                return;
                            }
                            students.remove(hook);
                            if (studentIdx > 0) {
                                studentIdx--;
```

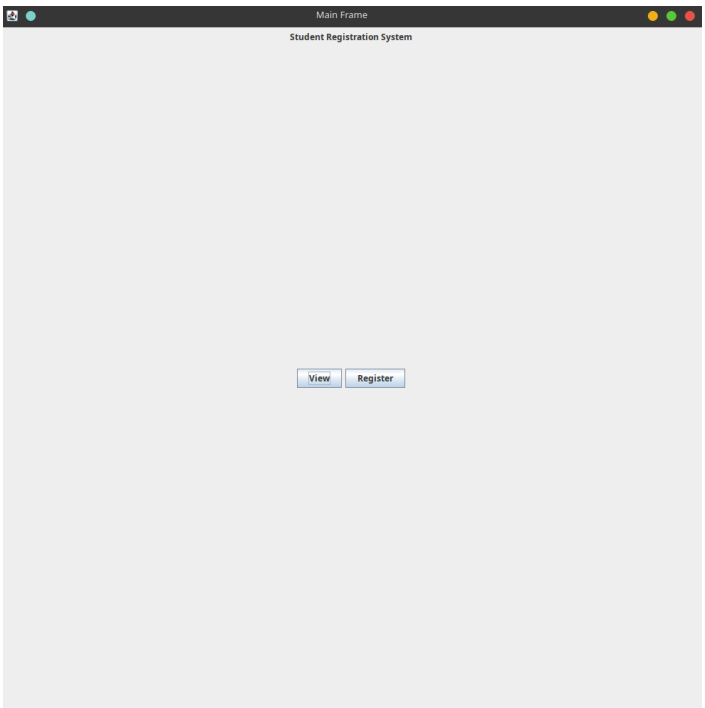
```
                    studentRenderer.setHook(students.get(studentIdx));
                    studentSelector.setHook(studentIdx);
                } else if (students.size() == 0) {
                    studentRenderer.setHook(null);
                    studentSelector.setHook(studentIdx);
                } else {
                    studentRenderer.setHook(students.get(studentIdx));
                    studentSelector.setHook(studentIdx);
                }
            }

        });
        Arrangement.center(comp, deleteBtn);

    } else {
        comp.add(new JLabel("No Students"));
    }
});

studentSelector.setRenderable((comp, hook) -> {
    comp.setLayout(new GridLayout(4, 1));
    Arrangement.bottom(comp, new JLabel("Select Student"));
    JButton prevButton = new JButton("Prev");
    prevButton.addActionListener(e -> {
        if (studentIdx > 0) {
            studentIdx--;
            studentRenderer.setHook(students.get(studentIdx));
            studentSelector.setHook(studentIdx);
        }
    });
    Arrangement.bottom(comp, prevButton);
    if (students.size() == 0) {
        comp.add(new JLabel("No Students"));
    } else {
        String[] ids = new String[students.size()];
        for (int i = 0; i < students.size(); i++) {
            ids[i] = students.get(i).id;
        }
        JList<String> idList = new JList<String>(ids);
        idList.setSelectedIndex(hook);
        idList.addListSelectionListener(e -> {
            studentIdx = idList.getSelectedIndex();
            studentRenderer.setHook(students.get(studentIdx));
        });
        comp.add(idList);
    }
    JButton nextButton = new JButton("Next");
    nextButton.addActionListener(e -> {
        if (studentIdx < students.size() - 1) {
            studentIdx++;
            studentRenderer.setHook(students.get(studentIdx));
            studentSelector.setHook(studentIdx);
        }
    });
    Arrangement.top(comp, nextButton);
```

```
        });

        JButton backButton = new JButton("Back");
        backButton.addActionListener(e -> {
            this.setVisible(false);
            prevFrame.setVisible(true);
        });

        JButton refresh = new JButton("Refresh");
        refresh.addActionListener(e -> {
            studentRenderer.setHook(students.get(studentIdx));
            studentSelector.setHook(studentIdx);
        });

        JPanel backContainerPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        backContainerPanel.add(backButton);

        JPanel topPanel = new JPanel(new GridLayout(2, 1));
        topPanel.add(backContainerPanel);
        this.add(topPanel, BorderLayout.NORTH);

        JPanel leftPanel = new JPanel(new GridLayout(1, 1));
        Arrangement.allCenter(leftPanel, studentSelector.comp);
        JPanel centerPanel = new JPanel(new GridLayout(1, 1));
        Arrangement.allCenter(centerPanel, studentRenderer.comp);
        this.add(centerPanel, BorderLayout.CENTER);
        this.add(leftPanel, BorderLayout.WEST);
        this.add(refresh, BorderLayout.SOUTH);
    }
}

public class RegistrationForm {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<Student>();
        JFrame mainFrame = new JFrame("Main Frame");
        JFrame registerFrame = new Register(mainFrame, students);
        JFrame viewFrame = new View(mainFrame, students);
        mainFrame.setSize(1000, 1000);
        registerFrame.setSize(1000, 1000);
        viewFrame.setSize(1000, 1000);

        JButton registerBtn = new JButton("Register");
        registerBtn.addActionListener(e -> {
            registerFrame.setVisible(true);
            mainFrame.setVisible(false);
        });

        JPanel buttonPanel = new JPanel();
        JButton viewBtn = new JButton("View");
        viewBtn.addActionListener(e -> {
            viewFrame.setVisible(true);
            mainFrame.setVisible(false);
        });

        buttonPanel.add(viewBtn);
```

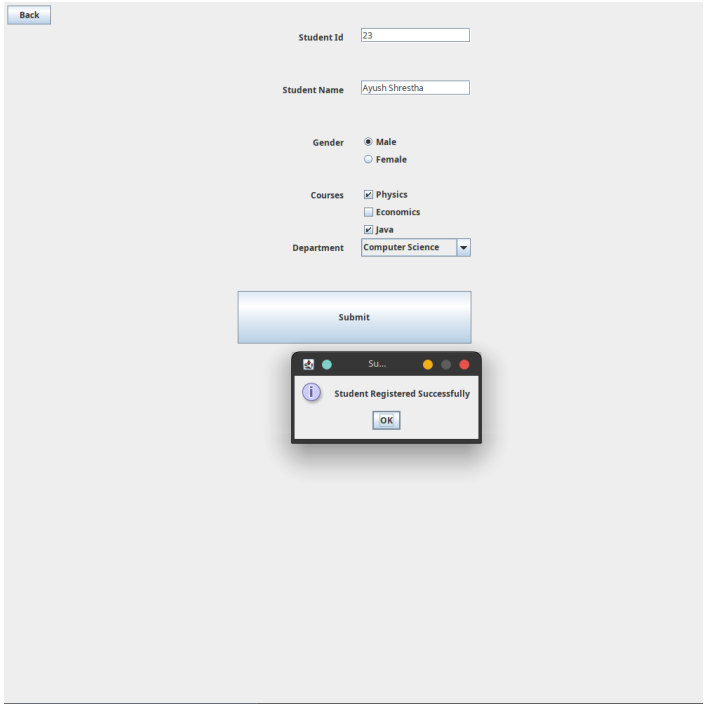
```
        buttonPanel.add(registerBtn);

        JPanel topBar = new JPanel(new FlowLayout(FlowLayout.CENTER));
        topBar.add(new JLabel("Student Registration System"));

        mainFrame.add(topBar, BorderLayout.NORTH);

        JPanel centerPanel = new JPanel(new GridLayout(1, 1));
        Arrangement.allCenter(centerPanel, buttonPanel);
        mainFrame.add(centerPanel, BorderLayout.CENTER);

        viewFrame.setUndecorated(true);
        registerFrame.setUndecorated(true);

        mainFrame.setVisible(true);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

Output:

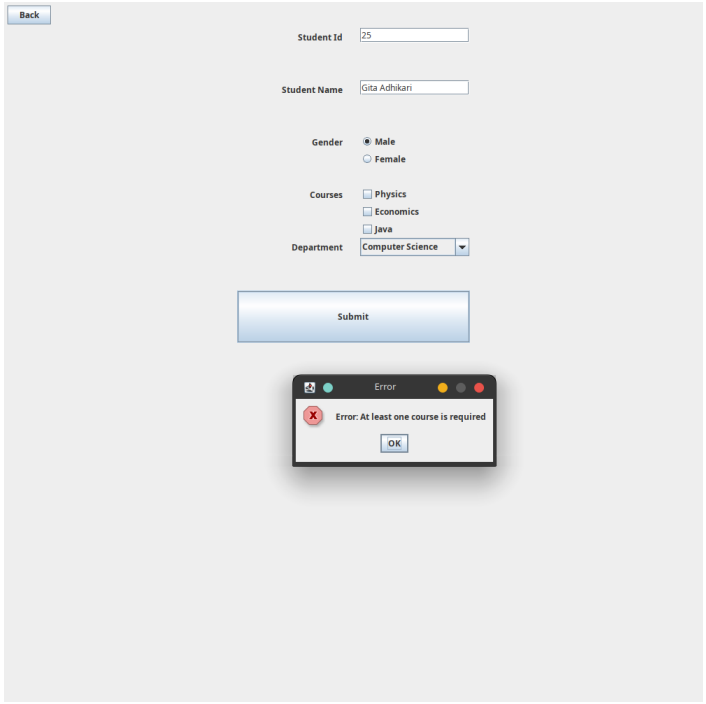
- **Main Frame**



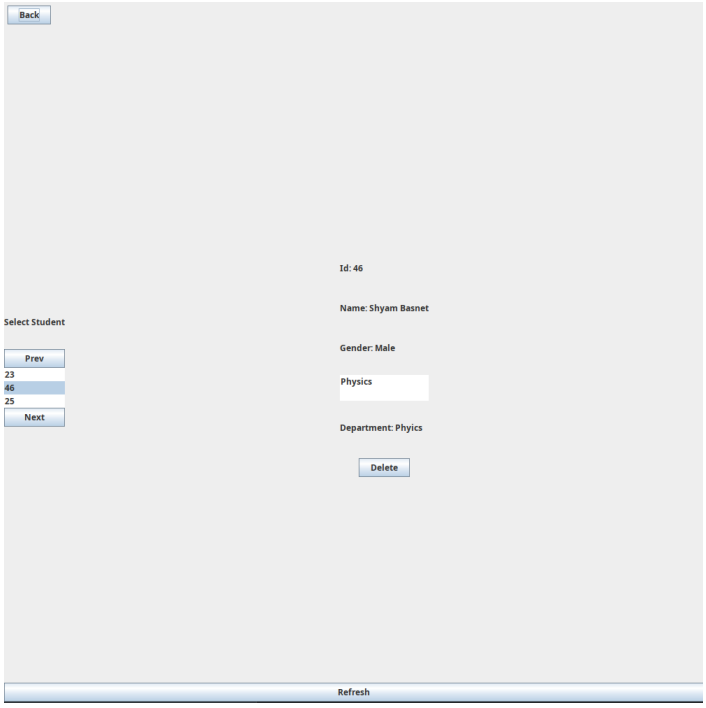
- **Registration Frame**



– **Registration Error**



– **View Frame**



2. **Program 2: JavaFX program**

```
package org.mdhe.jfx;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Tab;
import javafx.scene.control.TabPane;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.text.Text;
import javafx.stage.Stage;



public class HelloApplication extends Application {
    @Override
    public void start(Stage stage){
        stage.setTitle("WOW");

        Scene newScene = new Scene(createContent(), 300, 300);

        stage.setScene(newScene);

        stage.show();
    }

    private Parent createContent() {
        Tab BMItab = new Tab("BMI", createBMITab());
```

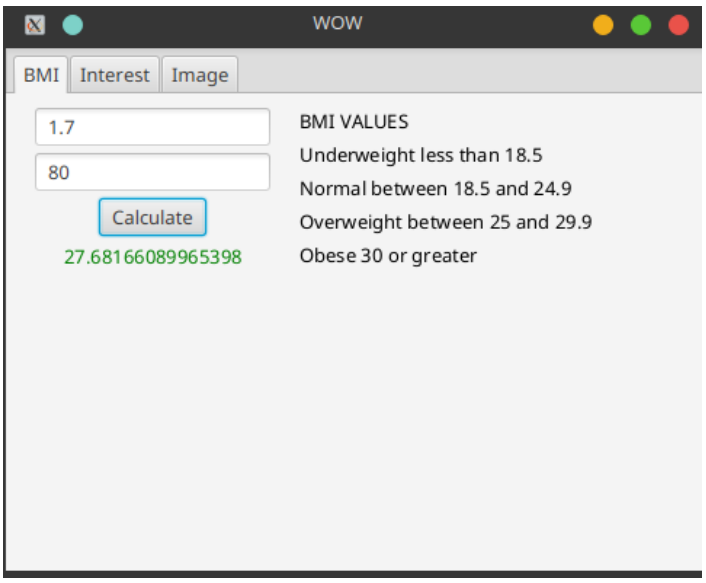
```java
        BMItab.setClosable(false);
        Tab InterestTab = new Tab("Interest", createInterestTab());
        InterestTab.setClosable(false);
        Tab ImageTab = new Tab("Image",createImageTab());
        ImageTab.setClosable(false);

        TabPane tabPane = new TabPane(BMItab, InterestTab, ImageTab);

        return tabPane;

    }

    private Node createBMITab() {


        TextField height = new TextField();
        height.setPromptText("Height in meters");
        TextField weight = new TextField();
        weight.setPromptText("Weight in KG");
        Button calculate = new Button("Calculate");

        Text Result = new Text();
        Result.setFill(Color.valueOf("green"));
        calculate.setOnAction(e -> {
            double h = Double.parseDouble(height.getText());
            double w = Double.parseDouble(weight.getText());
            double bmi = w / (h * h);
            Result.setText(String.valueOf(bmi));
        });

        VBox v = new VBox(
                height,
                weight,
                calculate,
                Result
        );
        v.setAlignment(Pos.TOP_CENTER);
        v.setSpacing(5);


        VBox details = new VBox(
                new Text("BMI VALUES"),
                new Text("Underweight less than 18.5"),
                new Text("Normal between 18.5 and 24.9"),
                new Text("Overweight between 25 and 29.9"),
                new Text("Obese 30 or greater")
        );
        details.setSpacing(5);

        HBox h = new HBox(v,
                details
        );
        h.setSpacing(20);
        h.setPadding(new Insets(10, 20, 0, 20));
```

```
        return h;
    }

    private Node createInterestTab() {


        GridPane gridPane = new GridPane(10, 5);

        gridPane.setPadding(new Insets(10, 20, 0, 20));

        TextField principal = new TextField();
        principal.setPromptText("Enter principal");

        TextField time = new TextField();
        time.setPromptText("Enter Time");


        TextField rate = new TextField();
        rate.setPromptText("Rate");

        TextField result = new TextField();
        result.setEditable(false);
        result.setFocusTraversable(false);

        Button calculate = new Button("Calculate");
        calculate.setOnAction(e -> {
            double p = Double.parseDouble(principal.getText());
            double t = Double.parseDouble(time.getText());
            double r = Double.parseDouble(rate.getText());
            result.setText(String.valueOf((p * t * r) / 100));
        });


        gridPane.add(principal, 0, 0);
        gridPane.add(rate, 0, 1);
        gridPane.add(time,0,2);
        gridPane.add(calculate, 0, 3);
        gridPane.add(result, 1, 0, 1, 4);


        return gridPane;
    }

    private Node createImageTab(){
        GridPane grid = new GridPane(10,10);
        grid.setPadding(new Insets(10,20,0,20));


        Image img = new Image("file:hojlund.jpg",0,0,true,true);
        ImageView imgView = new ImageView(img);
        imgView.setPreserveRatio(true);
        imgView.setFitWidth(200);
        grid.add(imgView,0,0);
```

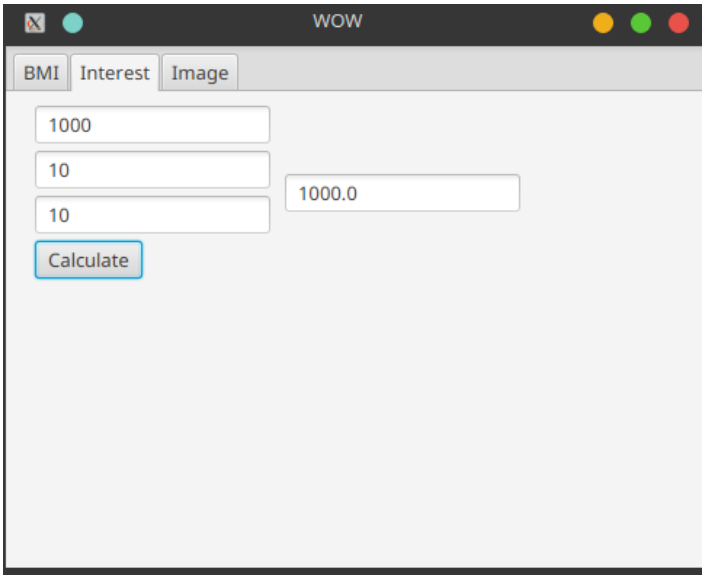
```
        Text hojlundInfo = new Text("Picture of football player Rasmus Hojl
        grid.add(hojlundInfo,1,0);

        Image java = new Image("https://logos-world.net/wp-content/uploads/
        ImageView javaImgView = new ImageView(java);
        javaImgView.setPreserveRatio(true);
        javaImgView.setFitWidth(200);
        grid.add(javaImgView,1,1);

        Text javaInfo = new Text("Java Logo");
        grid.add(javaInfo,0,1);


        return grid;
    }

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

Output:

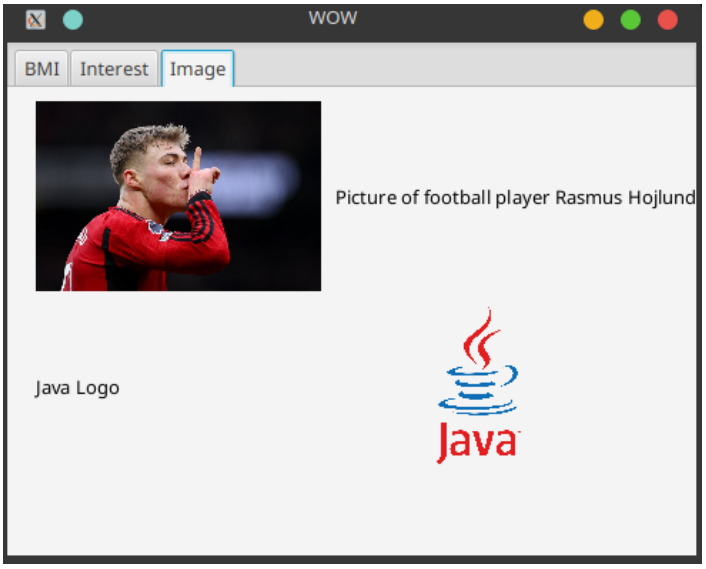
- **BMI Tab**



- **Interest Tab**



- **Image Tab**



Conclusion:

- We learned about different layouts in Swing and JavaFX
- We learned about GUI controls in Swing and JavaFX
- We learned about event handling and listener Interfaces