CPS 240 Homework 2

CPS 240 Due:

*Introduction*

One of the most commonly used patterns in computer science is the **model-view-controller** (MVC). True to its name, the MVC is typically made up of three segments: a *model*, a *view*, and a *controller*. The **model** is the main portion of the application and works directly with the data and logic of the application. The **view** is the output that the user sees, e. g. a window with buttons and text. The **controller** interacts with the user and accepts input. Chances are, you have run into an MVC pattern. For instance, you click (controller) a button (view) to update your social media info (model). As with most patterns, the specifications can blur at the details and can be implemented in different ways with different languages (patterns are language agnostic, their methodology is the importance).

*Assignment*

Create four classes – CalculatorMain, CalculatorModel, CalculatorView, and CalculatorController – which will be used to create a calculator application. CalculatorMain should be the entry point and should instantiate each of the following:

CalculatorModel.java

Holds all the data logic; holds **only** the following:

1. Should contain two private doubles, **a** and **b** with getters and setters.
2. Should also contain four public methods: **sum()**, **sub()**, **mul()**, and **div()**, which should return the added, subtracted, multiplied, and divided results of a and b respectively.

CalculatorView.java

Sets up the graphical aspect of the MVC. Use awt or swing. See the code in **Figure 1** for an example of how to set up the interface using awt (**you will have to have your class extend Frame to have this example code work properly**). Should contain:

1) Two textboxes for operand 1 and operand 2

2) Four buttons for added, subtracting, multiplying and dividing

3) A label to display the results of the operation (.setText( String s ))

CalculatorController.java

To access the model and the view data members (variables and methods), this class will need a reference to both. Pass references to both the model and view through the constructor’s parameters and store them in variables. Check for button clicks. A simple way is to add ActionListeners for each button as shown in **Figure 2** to call the appropriate method in the CalculatorModel. Update the result label. Also check for divide-by-zero.

Hint: Use the .getText() method (returns String) for TextFields to get their input data

**Figure 1:** Beginning of possible setup for CalculatorView.java

**private** **TextField** inputTF1; // you will have to check the values of these text field

**private** **TextField** inputTF2; // and use them with the model

**private** **Label** resultLabel; // update this to update the result shown

**private** **Button** addBtn;

// add the other buttons

// you should comment what this code does

**public** **CalculatorView**() {

// Setup Frame

setSize(300,200); // size of window

setLayout(**null**); // no layout is required for this

setVisible(**true**); // a visible window

// Text Field Labels

**Label** **opLabel1** = **new** Label("Op 1:"); // a label, which just present text

opLabel1.setBounds(25,50,50,25); // the location and size of the label (x, y, size\_x, size\_y)

**Label** **opLabel2** = **new** Label("Op 2:");

opLabel2.setBounds(25,75,50,25);

**Label** **resultTextLabel** = **new** Label("Result:");

resultTextLabel.setBounds(25,125,50,25);

resultLabel = **new** Label("Press the operation buttons to calculate.");

resultLabel.setBounds(25,150,250,25);

// Text Field (a TextField takes in user input)

inputTF1 = **new** TextField(); // operand 1

inputTF1.setBounds(75, 50, 100, 25);

inputTF2 = **new** TextField(); // operand 2

inputTF2.setBounds(75, 75, 100, 25);

// Buttons

addBtn = **new** Button("+"); // add

addBtn.setBounds(200,50,50,25);

// Add to Frame (if you do not add them to the frame, they will not appear)

add(opLabel1);

add(opLabel2);

add(resultTextLabel);

add(resultLabel);

add(inputTF1);

add(inputTF2);

add(addBtn);

}

**Figure 2**: Example of how to create multiple listeners for different buttons

button1.addActionListener( **new** ActionListener() {

**public** **void** **actionPerformed**( **ActionEvent** e) {

// do stuff for button 1

}

} );

button2.addActionListener( **new** ActionListener() {

**public** **void** **actionPerformed**( **ActionEvent** e) {

// do stuff for button 2

}

} );