

CST 8221 – JAP, Assignment #1, Part 1

Due Date: prior or on **21 February 2020**

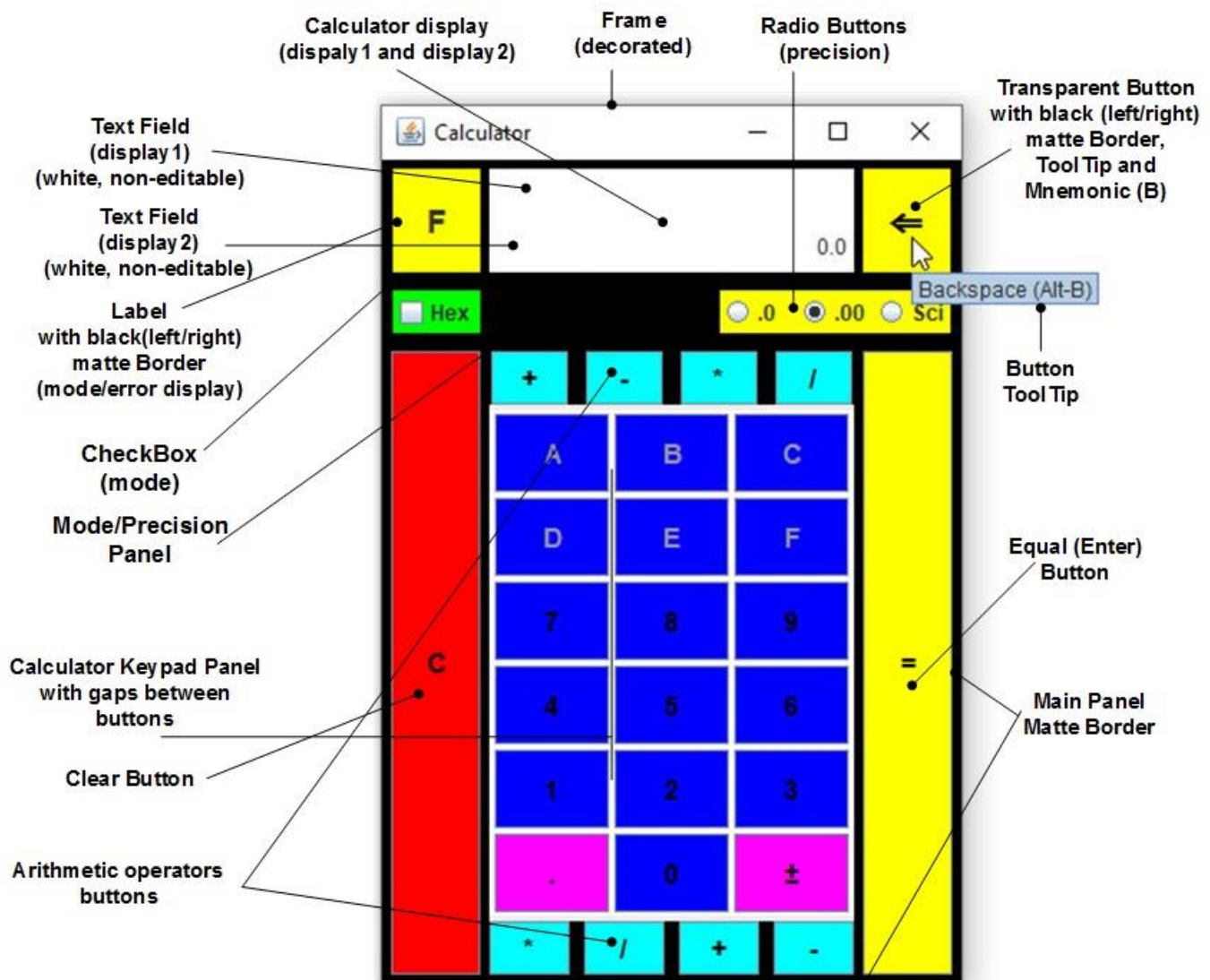
Earnings: 7% of your total course mark

Purpose: Building a pretty GUI using Swing

The purpose of this simple yet comprehensive assignment is to give you the enjoyment of exercising your programming skills and the delight of applying your knowledge of how to build a relatively simple GUI for a Calculator Application. The assignment is based on the material covered in lectures, lab exercises, and hybrid activities. You will also find Chapter 26 of your textbook to be very helpful.

Problem Specification:

In this part of Assignment #1 you are to build a relatively simple Swing GUI for a Calculator Application. Your GUI must have exactly the same appearance as the screen captures (**CalculatorSW_A1_W20_W10**) the posted on Brightspace. The image is captured with a screen resolution 1366X768 under Windows 10. The screen capture of the GUI with some implementation details is shown below:



Note: If you are running Windows 7 or 8, the frame borders and decoration may look differently but the rest of GUI should look the same as the one above. See the screen captures on BS LMS.

Requirements:

- ❖ The initial and the minimum size of the application frame must be (340, 500). The initial frame location on the screen must be set by the platform.
- ❖ The mode/error display label should have a preferred size of (55,55). It must be yellow at launch and must be opaque (not transparent). It must have a black matte border with thickness 0, 1, 0, 5. And must display the letter **F** in the middle of the label. Later the background color and the text will change depending on the mode of operation (floating-point or hexadecimal) or when an error occurs as a result of some calculation.
- ❖ The backspace button should have a preferred size of (55,55). It must be **transparent** with a black matte border with thickness 0, 5, 0, 1. You must use Unicode (Arrows chart (code page 21XX)) to display the text symbol \leftarrow on the face of the backspace button. The button must have a tool tip (see the image above). It must also react to the Alt-B key combination (mnemonic).
- ❖ All keypad buttons must have a black text. All calculator numeric keypad buttons must have a blue background. The sign(\pm) button and the . (dot) button must have a magenta background. You must use Unicode (Latin 1 supplement code page) to display the plus-minus sign symbol \pm . There must be a 3 pixels gap between the calculator keypad buttons. The alphabetical hexadecimal buttons must be disabled at launch. The panel that contains the keypad buttons must have a white background.
- ❖ The calculator arithmetic operator buttons must have a cyan background. The panels that contain the arithmetic operator buttons must have a black background. The gap between the buttons must be adjusted to the proper size so that it looks exactly the same as in the image above and the screen captures.
- ❖ The Clear button (C) must be red in color with a black text; the Equal (Enter) button (=) must be yellow in color with a black text. You should adjust the preferred size width of the Clear and Equal buttons so that the Clear is vertically aligned with the mode/error label and the Equal button is vertically aligned with the Backspace button.
- ❖ The calculator display consists of two text fields – display1 and display2. Both text fields must have 14 columns and height of 30. Both text fields must have a white background and must be non-editable. The displayed text must be right aligned. The display2 text field must display 0.0 when the GUI is made visible for the first time. The panel containing the calculator display, the mode/error label, and the backspace button must have a yellow background.
- ❖ The mode panel consist of a check box (mode) and three floating-point precision radio buttons. They checkbox and the radio buttons must be included in a *button group*. The radio buttons must be yellow. The **.00** radio button must be selected by default at launch. The checkbox must be green. The panel containing the container with the radio buttons group and the check box must be black. The radio buttons must be equal in size. You should adjust the preferred size width of the checkbox so that the check box is vertically aligned with the mode/error label. The mode panel should have an empty border
- ❖ To arrange the GUI components of the Calculator GUI you should use the following basic layout managers: BorderLayout and GridLayout. You may use other basic layouts but the appearance of the GUI must be the same as shown above.

- ❖ When the application frame is resized your GUI must have exactly the same behavior (appearance) as the screen capture **CalculatorSW_A1_W20_W10_RS** posted on Brightspace.
- ❖ The GUI may look a little bit different on your screens because you may have different operating system, screen resolution or different default “Look and Feel”, but relative locations, proportionality, alignments, and colors must same as on the screen captures.

Tasks:

The Object-Oriented analysis of the problem shows that, to solve the problem as stated in the problem specifications, you must design and implement several Java classes as outlined below. All classes must be placed in a package called **calculator**.

Class *CalculatorViewController*

The class *CalculatorViewController* is responsible for building and operating the calculator GUI. It must extend *JPanel* (not *JFrame*). The GUI should be built inside a constructor with no-parameters (the default constructor). The *CalculatorViewController* panel must be surrounded by a black matte border with the following insets (5, 5, 5, 5). It must have a *BorderLayout* with horizontal gap of 5p and vertical gap of 0p. The background of the panel must be black.

In this implementation the class must contain the following fields only:

```
private JTextField display1; // the calculator display1 field reference
private JTextField display2; // the calculator display2 field reference
private JLabel error; // the mode/error display label reference
private JButton dotButton; the decimal point (dot) button reference
private JButton [] hexButtons ; // reference to container for alphabetical hex buttons
```

If you like you can add some **final** fields to define constants like sizes, text, and colors used in the GUI.

The class must contain the following private method:

```
private JButton createButton (String text, String ac, Color fg, Color bg, boolean enable,
                             ActionListener handler)
```

The method is responsible for the creation of group of related buttons with the same basic properties (for example, the calculator numeric keypad buttons). The first parameter **text** is the button text label (for example, 8 or A). The second parameter **ac** represents the *action command* string for that button. The third parameter **fg** is the foreground color of the button. The fourth parameter **bg** is the background color of the button. The fifth parameter **enable** indicates if the button must be enabled or disabled at launch. The sixth parameter **handler** is a reference to instance of the event handler class (for example, object of type *Controller*). The method performs the following actions:

- Creates a new button with a specified **text** label;
- Sets the background and foreground colors of the button;
- Set the action command for the button. If *ac* parameter is *null*, the action command property of the button need not to be set;
- Set the size of the button font to 17 but does not change the button font name and style.
- If the **enable** parameter is **false** the button must be disabled.
- Registers the handler as an Action event listener for the button;
- Returns a reference to the created button.

Note: If the text of some of the created buttons displays as ..., Mac OS users may need to surround each button with an empty border and may need to set a button preferred size.

When creating the calculator keypad you **must call this method in a loop** in order to create all of the keypad buttons and add them to the calculator keypad panel (container). If the created button

is the **decimal point (.) button**, you must assign the reference returned by the method to the *dotButton* field before adding it to the panel. If the created button is an alphabetical hex button (A to F), you must assign the returned reference to one of the *hexButtons* array elements and then disable it (do not forget to create the array before running the loop – it must have 6 elements). You can use the same method to create other buttons (=, C, *, +, -, /), but not for the backspace button.

Class Controller

In this implementation (Part 1) this class is responsible for handling all the events generated by the GUI. In the Part 2 of the assignment implementation you will be allowed to add more controller classes. The class must implement the *ActionListener* interface. The class must be implemented as a **private inner class** of the *CalculatorViewController* class. In this implementation, the *actionPerformed()* method should implement the following only: if the check box or any button (including the radio buttons and the backspace button) is clicked the code of the method must get the action command string from the event and display it on the calculator display text field *display2*.

Class Calculator

This class is responsible for launching the application. The class must contain a main method only. Inside the main method you must first create a *CalculatorSplashScreen* object and call its *showSplashWindow()* method and display the splash screen for 5 s. Then the main method must call *EventQueue.invokeLater()* method with a *Runnable* instance the *run()* method of which creates a *JFrame* object; sets the frame title and minimum size; sets the default close operation for the frame; sets the content pane of the frame to a *CalculatorViewController* object; sets the application location at launch; and finally, makes the frame visible.

Class CalculatorSplashScreen

This class is responsible for displaying a splash screen before the launch of the application. The splash screen must display an image, and must contain your name and student number at the bottom of the screen window. The class must implement a method called *showSplashWindow()* that is responsible for building the splash screen and making it visible. It must contain a constructor which has as a parameter the duration time of the splash screen.

In Part 2 you will add a class *CalculatorModel* which will be responsible for the actual calculations.

What to Submit:

No paper submission is required for Part 1 of Assignment 1.

Code submission:

Compress in one **.zip** file all **.java** files, **.class** files, and **image** file(s). Upload the assignment **zip** file to Brightspace prior to or on the due date. The name of the zip file must have the following structure: Student's family name followed by the last three digits of the student ID number followed by **_JAP_A1P1**, and finally, followed by your lab section number (for example, s301). For example, *Ranev007_JAP_A1P1_s301.zip*.

The submission must follow the course submission standards. The **Java Assignment Submission Standard** and the **Java Assignment Marking Guide** are posted on Brightspace in the Course Information folder. Test Plan is not required for this assignment.

Enjoy the assignment. And do not forget that:

"A picture is worth a thousand words, but not all pictures are Swinging equally." Anonymous Swing Programmer
CST8221 –JAP, 21 January 2020, S^R