# CS 235 - Homework 8

## Deadline:

**11:59 pm** on **Friday, November 19, 2021.**

## Purpose

To provide practice with some additional components and listeners.

## How to submit:

Submit your `.java` files for this homework to the course Canvas site.

## Important notes:

- Follow the Java coding standards mentioned in previous homework handouts and discussed in class.
- Note that Java applications with graphical user interfaces are expected to be structured as demonstrated in the in-class example `ButtonTest.java`
  - (that is, with an application class that creates and displays a `JFrame` subclass instance within the event dispatch thread,
  - and a `JFrame` subclass that creates and adds a `JPanel` subclass instance to itself in its constructor,
  - and a `JPanel` subclass whose constructor creates and adds appropriate components to itself)
- Using appropriate additional helper methods is fine, as is using additional classes. IF you use additional public classes, be sure to submit those as well!
- Because graphical user interfaces are involved in some of this homework's problems, **the CS50 IDE will NOT work for all of this homework's problems**. (Running in a browser, on the cloud, it cannot access your screen to display a `JFrame`.)

  If you have a Terminal or bash shell, you can compile and run Java as you do from the CS50 IDE Terminal.

  AND -- I have verified that Java works -- compiles and runs -- from the **Command Prompt** on `vlab.humboldt.edu` as it does from the CS50 IDE, also.

  That is:

  - Log into `vlab.humboldt.edu`
  - In the search bar on the lower left, search for "command prompt", and click on the "Command Prompt" app that comes up.
  - Even though this is a Windows Command Prompt window and not a bash shell, commands such as `mkdir` and `cd` and `ls` work here.
  - I found that if I saved a `.java` file on the vlab desktop, then from the Command Prompt I could do

the following:

```
C:\Users\st10> cd Desktop
C:\Users\st10\Desktop> javac MyGuiApp.java
C:\Users\st10\Desktop> java MyGuiApp
```

...and my application would compile and run!

– (But save your `.java` files to your Google Drive for safer, longer-term storage that can be more easily accessed than the vlab Desktop!)

# Problem 1 - `ScrollColorPlay.java`

## *`JScrollBar` basics*

Consider the Week 11 Lab's `JScrollBarDemo.java` (also posted along with this homework handout for your convenience).

Amongst its other uses, a `JScrollBar` can convenient for selecting an integer within a range. And it can be made sensitive to `AdjustmentListener` events (and it turns out that the `AdjustmentListener` interface is another functional interface! So you can create an instance of its using a 1-argument lambda expression.)

One of its constructors expects:

- the desired orientation for the new `JScrollBar`, either the named constant `JScrollBar.VERTICAL` or `JScrollBar.HORIZONTAL`
- the `int` initial value of the new `JScrollBar` (and where the left- or top- of its "bar", depending on its orientation, will initially be placed)
- an `int` so-called extent -- this affects how "wide" the bar on the `JScrollBar` appears, but it is also how far the value "jumps" when you "click" outside the "bar" on the `JScrollBar`
- an `int` minimum value for this `JScrollBar`
- an `int` maximum value for this `JScrollBar`

A `JScrollBar` can be made sensitive to `AdjustmentEvent` events by adding an `AdjustmentListener` to it, using its `addAdjustmentListener` method.

This interface `AdjustmentListener` happens to be a functional interface, so that you can create an instance of this interface with its required method by using a one-argument lambda expression. (Its one argument is the `AdjustmentEvent` event that triggered it.)

And, `JScrollBar`'s `getValue` method expects no arguments and returns the `JScrollBar`'s current value.

This should be enough to try out some `JScrollBar` objects in this problem.

## *Your task*

Recall `ColorPlay2.java` from Homework 6, Problem 1. It turns out that scrollbars are quite a

convenient choice for entering possible red, green, and blue values.

Create a Java application `ScrollColorPlay.java` that uses `JScrollBar` instances for entering the red, blue, and green values, meeting the following additional specifications:

- It must contain your name (visibly) within it, somewhere, in some pleasing form.
- Somehow label the scrollbars to indicate which is for the Red value, which is for the Blue value, and which is for the Green value (this can be tricky; see the tips below).
- Use an `AdjustmentListener` object or object such that, whenever one of these scrollbars is adjusted:
  - whatever panel you have showing the resulting color is updated accordingly, and
  - uneditable textfields containing the current red, green, and blue values are also updated accordingly (so if you find a combination you like, you can write down the red, green, and blue values for easier use elsewhere)

You choose the layout, but I will warn you that I found it a bit tricky - here are some tips from my experience playing with this:

- `JScrollBars` don't seem to work well using `FlowLayout` - in my playing around, they often came out very short and unusable.
  - I have had better luck using them in the `NORTH`, `SOUTH`, `EAST` or `WEST` of a `BorderLayout`,
  - or in a 3-row, 1-column grid on a `GridLayout` subpanel in the center of a `BorderLayout` panel or subpanel.
- But - there are three scrollbars! There is probably something cool to be done here with `GridLayout`, but I used the multi-`BorderLayout`-panel approach instead -- one scrollbar in the `NORTH` of a panel, another in the `NORTH` of a second panel in the center of the first, and another in the `NORTH` of a third panel in the center of the second...!
- Since the scrollbar takes up the whole `NORTH` if you use `BorderLayout` - how could one label them? The user needs to know which is red, which is blue, and which is green!
  - I hope one of you will come up with something slicker, but ...
  - This could be as straightforward as a label above or below the 3 scrollbars saying "set Red, Green, and Blue values as desired:" (and hope the user assumes they set Red using the top scrollbar, Green using the 2nd scrollbar, and Blue using the bottom scrollbar)
  - I *suspect* that a `GridLayout` subpanel in the `WEST` of the outermost panel can work if you play with the `JLabel` objects' font sizes a bit... 8-)

I found the resulting application even more fun to play around with than the `ColorPlay2` was -- `JScrollBars` really are a good component for this task!

Submit your resulting `ScrollColorPlay.java`.

## Problem 2

Let's combine more components into a single GUI. Consider a setting/theme of your choice (besides pizza... 8-) ). But, consider a setting where you'd like someone to:

- make a choice of exactly one option from not too large a number of options, for which radio buttons would be appropriate (choose one entree, choose one auto make, choose one hotel room type, choose one player category, choose one tree species, choose one concert, etc.)
- make a choice of zero or more options from not too large a number of options, for which checkboxes would be appropriate (choose zero or more condiments, choose zero or more auto options, choose from optional extra-cost hotel room amenities, choose player or tree characteristics, choose concert extras, etc.)
- make a choice of an integer quantity, for which a scrollbar would be appropriate (choose a desired spiciness level between 1 and 10, choose the sound-level of the engine between 50 and 80, choose the desired room quality index between 1 to 5, set player initial strength between 5 and 15, etc.)
- (you may add additional choices using additional components of your choice, if you wish)

Design and implement a pleasingly-designed application `PrefChooser.java`, which meets the following requirements:

- It must contain your name (visibly) within it, somewhere, in some pleasing form.
- It should include the components mentioned above to allow the user to make the choices you have determined above.
- The overall layout should look "nice".
- The user can indicate that they have finished making their choices by clicking a button that, when clicked, "sends" the chosen preferences by bringing up an appropriate `JOptionPane` whose contents include a summary of the current choices, based on what the user has selected at that point.
  - Notice that this is different from many of our earlier applications -- something isn't happening each time the user selects or unselects something. Instead, when a button is clicked, **then** the **current** selections are used.

Submit your resulting `PrefChooser.java`.