

CS 235 - Homework 9

Deadline:

Due by **11:59 pm** on **Wednesday, November 4, 2015**.

How to submit:

Submit your files using `~st10/235submit` on nrs-projects, with a homework number of 9, by the deadline shown above.

Purpose

To provide practice with some additional components and listeners.

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 in the way demonstrated in the in-class examples (as in `ButtonTest.java`)
 - ...but using appropriate additional helper methods in the `JPanel` subclass is fine, as is using additional classes. IF you use additional public classes, be sure to submit those as well!
- It is possible that some of your programs may be posted to the course Moodle site.

Problem 1

As a warm-up: after lab, I added a second menu item to the File menu in the Week 10 Lab example `ComponentPlay.java`. I also added helper methods to perhaps make this example more readable.

The second menu item, `Wake UP`, seeks to "wake up" the user by changing the sub-panel backgrounds to yellow. This is, ah, certainly eye-catching. It also gets old fast.

Modify the Week 10 Lab version of `ComponentPlay.java` into `ComponentPlayMore.java`, meeting the following requirements:

- add another `@author` line to its opening Javadoc comment, indicating that you have adapted this
- change the `@version` line to the date that you last modified this
- somehow visibly add your name to one of the sub-panels
- add a third menu item, `Calm down`, that changes the sub-panel backgrounds to a color of your choice that is less garish than `Wake UP`'s yellow

Optionally, you may make other changes (to fonts, colors, taco choices, add more menus and menu items, etc.) IF you would like.

Submit your resulting `ComponentPlayMore.java`.

Problem 2

Now that you are warmed up, let's combine more components into a single GUI. Consider a setting/theme of your choice (besides tacos... 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, 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, etc.)
- make a choice of exactly one from a somewhat larger number of options, for which a combo box/drop down box would be appropriate (choose one of many beverages, choose one of a large-ish number of states or cities or counties, 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.
- It should include a menu bar with at least one menu with at least two menu options of your choice (change something's color, change something's font, quit, etc.)
- 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 `ComponentPlay` -- 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`.

Problem 3

Recall `ColorPlay1.java` from Homework 5, Problem 3. 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` private class so that, whenever one of these scrollbars is adjusted:

- the "center" panel showing the 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 `PAGE_START`, `PAGE_END`, `LINE_START` or `LINE_END` of a `BorderLayout`.
- But - there are three scrollbars! There is probably something cool to be done here with `GridBagLayout`, or maybe even `BoxLayout` or `GridLayout`, but I used the multi-`BorderLayout`-panel approach instead -- one scrollbar in the `PAGE_START` of a panel, another in the `PAGE_START` of a second panel in the center of the first, and another in the `PAGE_START` of a third panel in the center of the second...!
- Since the scrollbar takes up the whole `PAGE_START` 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 I can note that a `Box` in the `LINE_START` of the outermost panel can work if you play with the `JLabels`' font size a bit... 8-)

I found the resulting application even more fun to play around with than the `ColorPlay1` was -- `JScrollBars` really are a good component for this task!

Submit your resulting `ScrollColorPlay.java`.