CS 235 - Week 5 Lab Exercise - 2021-09-24

Deadline

Due by the end of lab on 2021-09-24.

How to submit

Submit your `.java` files for this lab on https://canvas.humboldt.edu

Purpose

To practice a bit with some of the Java AWT and Swing classes discussed this week.

Important notes

- IF you are attending the lab via Zoom, you are expected to pair program in a breakout room (possibly trio-program if necessary based on class members' Internet and the number of class members attending via Zoom).
  - In this case, **be sure to TYPE BOTH (all) OF YOUR NAMES** in the beginning comment of EACH of your `.java` files

  But, because of the delta variant surge, if you are attending lab in person in BSS 317, you will each work on a separate computer, although discussion amongst those attending will be encouraged!
- Because graphical user interfaces are involved here, **the CS50 IDE will NOT work here**. (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!)

Lab Exercise set-up

- FIRST: in your directory/folder for today's lab exercise, create a local copy of:
  - `GameDie.java` (this can be either the Week 1 Lab version, or your version from the Week 4 Lab Exercise)
  - `PreLabButtonTest.java`, included along with this lab exercise handout

Problem 1

We ran some examples of `JOptionPane` during the Week 5 Lecture in `jshell` -- a little application including these, `SimpleOptionPaneTest.java` is also now posted with Week 5 Lecture examples.

Write a small Java application, `OptionPaneDiePlay.java`, that does at least the following:

- Creates at least one `GameDie` instance.
  - (Options: you can let its size be entered as a command-line argument, or you can simply create it with a certain number of sides always.)
- Rolls the die, and displays what it rolls in a message dialog (for example, like the `simpleDialog` does in `SimpleOptionPaneTest.java`).

Some **optional** additions:

- Have your application use an input dialog (using `JOptionPane`'s static method `showInputDialog` as in the example `SimpleOptionPaneTest.java`) to ask the user if they would like to roll the die again, and use this to control a loop that repeatedly rolls the die and displays the result in a message dialog until they answer negatively.
- Try the above, but experiment using the `showConfirmDialog` static method of `JOptionPane` instead of an input dialog (see https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/javax/swing/JOptionPane.html)

Problem 2

Fun fact #1! You can put HTML in labels...!

In the `String` given as the argument to `JLabel`'s constructor, you can surround the `String`'s contents with `"<html> ... </html>"`.

For example:

```
JLabel greeting = new JLabel("<html><h1>Hi! I am a JLabel!</h1></html>");
```

But note: Horstmann warns, on p. 646 of the course text, that "We don't recommend HTML in *buttons* -- it interferes with the look-and-feel."

Fun fact #2! Java has a `Font` class!

Aren't you getting tired of me showing you GUI examples with tiny little font sizes?

...Another way to make those more readable, that ALSO works for JButton objects, is to specify a different font for `JLabel` and/or `JButton` objects.

- The AWT, package `java.awt`, provides a `Font` class,

  and many components either inherit or have a `setFont` method that expects a `Font` instance as an argument).
- One of `Font`'s constructors expects the following arguments:
  - a `String`, giving the **name** of the font desired
    - it can be logical or physical
    - we'll stick with logical for now, because these 5 must be supported by any Java runtime environment:

      `"Dialog"`, `"DialogInput"`, `"Monospaced"`, `"Serif"`, `"SanSerif"`
      - (These logical fonts are not actual font libraries -- they are mapped to physical fonts by the Java Runtime Environment (JRE). The look and feel may vary, but they will exist!)
  - an `int`, a **constant** provided from the `Font` class giving the **style** of the font desired.
    - For example: `Font.PLAIN`, `Font.ITALIC`, `Font.BOLD`, `Font.BOLD|Font.ITALIC`
  - an `int`, a size in **points** (I believe there are 72 points in an inch.)
- SO, for example:

```
Font myAppFont = new Font("SanSerif", Font.PLAIN, 20);
JLabel myLabel = new JLabel("Howdy!");
myLabel.setFont(myAppFont);
this.add(myLabel);
```

Fun fact #3! You can change a `JLabel`'s text's color!

`JLabel` (and many components) have a `setForeground` method. In the case of a `JLabel`, `setForeground` allows you to set the color of the text of that label (that's considered to be the foreground of a label).

- This can take `Color` class constants (such as `Color.RED`) as its argument.
- The `Color` class also includes a constructor that takes 3 arguments, `int` values in [0, 255], that let you specify the red, green, and blue (RGB) values for a color you want.

NOTE: whether you are ALLOWED to set some graphical components' background colors or not (using `setBackground`) is Operating System-dependent...! And sometimes there are just more steps for doing so...

BUT -- for example:

```
Font myAppFont = new Font("SanSerif", Font.PLAIN, 20);
```

```
JLabel myLabel = new JLabel("Howdy!");
myLabel.setFont(myAppFont);
myLabel.setForeground(new Color(0, 102, 0));
this.add(myLabel);
```

Your task for Problem 2

Copy the source code file `PreLabButtonTest.java`, and modify it, meeting **at least** the following requirements:

- (You may carefully change the name of this class if you like, but be sure it still includes `ButtonTest` somewhere in its name if you do.)
- Include both of the names from your pair in a third `@author` line in its opening comment
- Change the `@version` comment appropriately
- Add a `JLabel` to this containing your names.
  - Change the foreground color of this label to a noticeable-and-also-readable color of your choice.
  - (Where it appears relative to the other components is your choice.)
- Add a fourth active `JButton` (and appropriate additional code as needed) for a **fourth** color of your choice.
- Noticeably change the **fonts** from the default for all 4 buttons and for the new label.
  - **optional** addition: add another `JLabel` whose text uses HTML. (Where it appears relative to the other components is your choice.)
- Change the **size** of the `ButtonFrame` as desired.
- You may make other additions and changes if you like, but you need to keep the original three buttons, and they should still work to change the background's color.

- When you are done, or before you leave lab, use Gmail to
  - MAIL a copy of your `.java` files to BOTH/ALL of you, and
  - EACH of you should SUBMIT the required files on Canvas