CS 235 - Week 6 Lab Exercise - 2021-10-01

Deadline

Due by the end of lab on 2021-10-01.

How to submit

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

Purpose

To practice more with a Java GUI application using the Java Swing and AWT classes, including practice with `JTextField` and with Swing borders, and some more practice catching exceptions and using `Font`.

Also includes an introduction to one means of getting a formatted `String` version of a `double` value, and specifying the horizontal alignment of text in a `JTextField`.

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:
  - `Mult2.java`, included along with this lab exercise handout

Problem 1

Answer the following questions in a plain text file named `lab6-1.txt`, which starts with both of your names.

Problem 1 - part a

Did you notice that the value in the `JTextField` object in Week 6 Lecture's example `TextField1.java` was left-justified?

That's just the default justification, though. You can change this default justification using `JTextField`'s `setHorizontalAlignment` method. You call this method with the appropriate named constant for the horizontal alignment you want.

If you Google something like "`Java 16 JTextField class`", one of the top matches should be for the Oracle Help Center's documentation for the `JTextField` class.

And if you click on the name of the `setHorizontalAlignment` method within that documentation, that's a link to additional documentation for this method.

List the **five named constants** that are valid for use as arguments for `setHorizontalAlignment`.

Problem 1 - part b

A method's documentation in the Java API also includes the types of any exceptions it might throw.

Looking at the documentation for the `JTextField` class's `setHorizontalAlignment` method, give the **name of the specific exception type** that `setHorizontalAlignment` might throw.

Problem 1 - part c

Aside - a quick intro to/demo of formatting output in Java!

Our course text, in Chapter 3, Section 3.7.2, on pp. 78-83, describes two means for formatting output in Java. (There are more...!) But the two described here are based on the C library's `printf` function, so if you have ever used that, you'll recognize the format specifiers used in these methods described in Section 3.7.2: Java's `printf` method and the `String` class' static method `format`.

For today's lab, we are interested in the `String` class' static method `format`, since it returns a `String` suitable for use as an argument to a `JTextField` object's `setText` method!

Here is a short example to get you started (and you can then read up more on these later).

The `String` class' static method `format` expects:

- a format string, with desired text also containing format specifier(s), that each start with `%`, for each value you want formatted and inserted within that text,
- and then it also expects an expression for each format specifier in that format string.

Then it returns a `String` with those expressions' values inserted within that format string's text, formatted as specified.

For example:

- `"%s"` specifies to format and insert a string
- `"%7.2f"` specifies to format and insert a floating point value in a field of size 7, with a precision of 2 fractional places (and the numeric value will be right-justified in that field).

So,

```
String.format("Hello, %s! Here is your $%7.2f.",
              "Sakura", 433.6666);
```

returns the `String` object:

```
"Hello, Sakura! Here is your $ 433.67."
```

(And you can verify this by trying it in `jshell`.)

As a second example:

```
double weight = 3.312436;
String wtString = String.format("%5.1f", weight);
```

sets `wtString` to `"  3.3"`.

(And you can verify this, also, by trying it in `jshell`.)

Your task for Problem 1 - part c

Using `jshell` to "debug" your statement, write a statement that declares a `String` variable `partC` and uses the `String` class's static `format` method to format the `double` value 57.38 to exactly 4 fractional places, with one space to the left of the 5 in the resulting `String`.

Copy this statement into your `lab6-1.txt` as your first answer for this part, and on the next line put the resulting value of variable `partC`.

Problem 1 - part d

Using `jshell` to "debug" your statement, write a statement that declares a `String` variable `partD` and uses the `String` class's static `format` method to format the `double` value 57.38 to exactly 1 fractional place, with no spaces to the left of the 5 in the resulting `String`.

Copy this statement into your `lab6-1.txt` as your first answer for this part, and on the next line put the resulting value of variable `partD`.

Problem 2

Read over the provided Java source code file `Mult2.java`; most of it should be understandable to you based on the Week 6 lecture.

Copy the source code file `Mult2.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 `Mult2` somewhere in its name if you do.)
- Include both of the names from your pair in a second `@author` line in its opening comment.
- Change the `@version` comment appropriately.
- Add a `JLabel` to this containing your names.
  - The color and font of this `JLabel` are your choice -- you can use the default color and font, or you may set them as you wish, as long as the result is reasonably visible!
  - (Where this appears relative to the other components is also your choice.)
- Modify this so that **at least two components** use a font or fonts visibly and obviously different than the default font.
- Give **at least two components** a noticeable, visible border of your choice.
- Make all three of the existing `JTextField` objects right-aligned. (Numbers look better when they are right-justified!)
- Modify this so that, if the user enters a value that cannot be parsed as a `double`, a `JOptionPane` will be displayed letting the user know that a number must be entered, and all 3 textfields will be set to contain `0`.
  - Hint: you need to add one or more appropriate `try-catch` block(s) to do this.
- Modify this so that the product displayed appears in a **non-editable** `JTextField`.
- Modify this so that the product displayed is formatted to precisely 3 fractional places.
- Change the **size** of the `Mult2Frame` as desired.
- You may make other additions and changes if you like, but you need to keep the existing functionality (users should still be able to enter two numbers into `JTextField` objects and click a `JButton` to see their product displayed in another `JTextField` object.).

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