CS 273 Laboratory 12: Java Libraries

This laboratory introduces you to Java's standard libraries and the use of API (Application Programming Interface) documentation to learn how to use a few of the classes associated with graphics user interfaces (GUIs) in Java. This lab has a total of 120 possible points.

Java Libraries

Java has many predefined classes. We've seen a few of these already, such as the `String`, `Color`, `Graphics` and `Math` classes. These classes are collected into "packages" of related classes which you can `import` into your program.

Your textbook covers only a few of the many class libraries in Java, however; for learning how to use a new class, the Sun's Java API web pages can be very helpful: **http://docs.oracle.com/javase/6/docs/api/**.

The page is made up of three frames: two small ones on the left of the page and a larger one on the right. The two frames on the left allow you to browse a list of Java packages (the top left frame) or browse a list of all individual classes in alphabetical order (the bottom left frame). When a package is selected, the lower left frame will show only the classes belonging to the selected package; to get back the list of all classes in all packages, select the "All Classes" link at the top of upper left frame.

To see detailed information about a particular class, select it in the lower left frame. The detailed information will be displayed in the large frame on the right. The class information is typically composed of the following sections:

- a short description of the class
- a "Field Summary" which describes the public and protected attributes of the class
- a "Constructor Summary" which is a list of all the available constructors for the class
- a "Method Summary" which lists the public and protected methods of the class

Clicking on any of the hyperlinks in the summaries will bring you to the part of the page with more specific details about that item.

Pro Tip: Often you can find the SDK page for the class you want quickly by searching for it using Google. Include "java 6" in your search query. For example, if you search for "java 6 jframe" the first hit will likely be the SDK documentation for the JFrame class.

Laboratory

Create a directory named `lab12` on the `cs273` your network drive. Run BlueJ and create a new BlueJ project in your `lab12` folder. Create a new class named `JavaAPILabMain` in your BlueJ project. Delete the example code and start afresh with a new class definition that contains a `main` method. To confirm everything is ready to go, add a `System.out.println` call to your main method and have it print out the name of your favorite movie. Then compile and run the program to verify that it works.

Part 1: Explore the Java API

Using the two left-hand frames in the API browser-window, find the `String` class and display its details in the large frame.

Notice that right above the words `Class String` are the words `java.lang` in a smaller font. `java.lang` is the package in which class `String` may be found. You already know that often you need to add an `import` statement to your program to make certain classes available; this is how you find out whether you need an `import` statement and, if so, which one. As it happens, `java.lang` is Java's default package: you never need to add an `import` statement for `java.lang`, because these classes are always available. If a different package name should appear there, however, you will know that you need to add an `import` statement to your program if you wish to use that class. *You will need to use this fact later in this lab.*

Answer the questions below:

1. Jill has two strings s1 and s2 which each contain different email addresses (e.g., `bill@microsoft.com`). She wants to know if both addresses are from academic institutions. The easiest way to do this is to see if both addresses end with "`edu`". Find the best method in the String class for doing this and complete the code Jill should use below:

   ```
   boolean bothEduAddresses = ________________________________________;
   ```

2. Paul is writing software to store information about all the card carrying members of his secret club. Each membership card is stamped with a unique ID that contains three sections separated by hyphens. These sections are always four digits, four

characters and then eight digits respectively. For example, one ID is "1994-NEPR-48573923".

Find the best method in the String class for comparing two ID strings to see if their second sections match. *No other methods in the `String` class should be necessary.* Write a code snippet using that method to compare the 4th and 14th entry in the members array to see if their second sections match.

checkpoint 1 (10 points): Show your lab instructor or assistant that you have the correct answers to the above questions.

Part 2: Using the JFrame Class

In this section, you will add some code to your main method that creates an empty window on the screen. Locate the `JFrame` class in the Java API documentation. Then follow these steps to use that class in your program.

1. Add the required `import` statement to the top of your `JavaAPILabMain` class.
2. In your `main()` method, create a variable of type `JFrame` object. Then, assign a new `JFrame` object to that variable. Use the constructor that will allow you to set the window title to `"Fresh Green Beans"`?

3. Next, you need to instruct the `JFrame` that it is to be visible on the screen.
    a. The method for setting the size of the `JFrame` has been inherited from another class. Find this method and use it in your code to make the `JFrame` visible.
    b. Add a comment above that line of code to indicate which class the method was inherited from.

4. Now we need to set the size (width and height) of the window.
    a. Locate the required method. (It has also been inherited.)
    b. To call this method, you will need an instance of the `Dimension` class.
        i. Add a new import statement to your code so that you have access to the `Dimension` class
        ii. Then create a variable and assign it a new `Dimension` that is 300x300
    c. Use the `Dimension` object you created to set the size of your window. Be sure to import any Java packages that you need to utilize this object.

5. Finally, we want the program to exit when the user clicks the red close button on the top-left corner of the window frame. `JFrame` has a particular method for specifying this behavior. Locate and use this method on your window to get the desired behavior. **Hint:** You will need to pass in a particular value that has been declared in the `JFrame` class. Access the value via that class like so, `JFrame.`THE_VALUE.

checkpoint 2 (20 points): Show your lab instructor or assistant that your program creates a 300x300 window with the title "Fresh Green Beans" and demonstrate that that the program exits when you hit the close button.

Part 3: Understanding Components

Answer the questions below:

1. Locate the `Component` class and read its documentation. What are some examples of components?
2. Return to the `JFrame` class documentation. Under "Methods inherited", one may note that the `JFrame` class has inherited a particular method for adding a new `Component` to a `JFrame` object. In fact, there are several overloaded versions of this method. What is the name of this method? How many versions of it exist?
3. Which of the following classes is not a `Component`: `Choice, JTextArea, FlowLayout, Canvas`?

checkpoint 3 (10 points): Show your lab instructor or assistant your answers.

Part 4: Refactoring Interlude

Follow these steps:

1. Create a constructor for your `JavaAPILabMain` class. Move the code from your `main()` method into that that constructor.
2. Have the `main()` method create an instance of your `JavaAPILabMain` class (thus calling the constructor).
3. Declare a method named `layout` in your code and then call it in your constructor (passing it your `JFrame` object):

   ```
   private void layout(JFrame frame)
   ```

   The method should do nothing for now.
4. Verify that your program still has the same behavior as before.

checkpoint 4 (5 points): Show your lab instructor your working program.

Note: In the remainder of this lab, many of the details about how to use a class are not given so that you will gain experience using the API documentation.

Part 5: Using Components

The `layout()` method you created in the previous checkpoint will be used to layout the content of the window.

First, you must set the layout manager for the window. The layout manager determines how the components will be arranged inside the window. The simplest layout manager is `FlowLayout`. Add code to your `layout()` method that creates a `FlowLayout` object and then sets it as the layout manager for your `JFrame` object. You will likely need to add a new import statement to the top of your file to support this.

Now add code to `layout()` that creates the following components and adds them to your window:

1. A `JButton` labeled "Clear"
2. A `JTextArea` that has 5 rows and 25 columns. Insert some initial text into this text area that reads: `"I love to eat fresh green beans!"`
3. A `JButton` labeled "Edit"

You will likely need to add some new import statements to the top of your file to support this.

checkpoint 5 (25 points): Show your lab instructor or assistant that your program now has two button and a text area.

If you have completed all above checkpoints, you have a grade of C- (70) for this lab.

Part 6: Learning about Action Events

When the user presses a button, it creates an event. Right now, your program does nothing on a button press. If you want to have your program respond, your class needs to be able to "listen" for action events from the buttons.

Use the Java API to read about the `ActionListener` interface. Answer these questions:

1. What Java reserved word is used to have a class implement an interface?
2. How many methods are declared in the `ActionListener` interface?
3. How is the `ActionEvent` class related to this interface?
4. What method on a `JButton` object would you call to add an action listener to it?

checkpoint 6 (10 points): Show your answers to your lab instructor or assistant.

If you have completed all above checkpoints, you have a grade of B- (80) for this lab.

Part 7: Implement the Clear Button

In this part of the lab, you will implement the 'Clear' button so that when the users presses it, all the text in the text area will be removed. Follow these steps:

1. In your `layout()` method, you created a `JTextArea` object. To make the object accessible to other methods in the class, declare an instance variable in the `JavaAPILabMain` class and have the `layout()` method assign the `JTextArea` object to this variable. Should your new instance variable be public or private?
2. Your program should import the `ActionListener` interface and also the `ActionEvent` class.
3. Implement `ActionListener` with your `JavaAPILabMain`.
4. To fulfill its obligation as an implementor of the `ActionListener` interface, create an `actionPerformed()` method in the `JavaAPILabMain` class. This method must have exactly the definition required by the `ActionListener` interface. When called, the method should clear the content of the `JTextArea` by setting it to an empty string. **Hint:** The best method for setting the text in the `JTextArea` is a method that `JTextArea` inherited from the `JTextComponent` class.
5. In your `layout()` method, the `JavaAPILabMain` object should add itself as the action listener for the "Clear" `JButton`.

checkpoint 7 (20 points): Show your working Clear button to your lab instructor or assistant.

If you have completed all above checkpoints, you have a grade of A (100) for this lab.

Part 8: Implement the Edit Button

Implement the Edit button so that when the user presses it, `" Yum!"` is appended to the existing text in the text area.

To do this, your `actionPerformed` method will need to distinguish between events that it received from the Clear button vs. the Edit button. You can accomplish this by examining the `ActionEvent` object that is passed to the `actionPerformed` method.

Don't forget to tell the Edit button that you want to listen to it.

checkpoint 8 (20 points): Show your lab instructor or assistant that both your Edit button and your Clear button behave correctly.

If you have completed all above checkpoints, you have a grade of A (120) for this lab.

Part 9: Finish Up

Close all windows, and log off. You're done.