Carleton University
Department of Systems and Computer Engineering
SYSC 2004 - Object-Oriented Software Development - Winter 2015
Lab 11 - Graphical User Interfaces

Objective

In this lab, you'll gain more experience building simple graphical user interfaces (GUIs).

Attendance/Demo

To receive credit for this lab, you must demonstrate your work. **Also, you must submit your lab work to cuLearn**. (Instructions are provided in the *Wrap Up* section at the end of this handout.)

When you have finished all the exercises, call a TA, who will grade the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. **Any unfinished exercises should be treated as "homework"; complete these on your own time, before the final exam.**

Getting Started

If you haven't done so already, download the zip file containing the *TicTacToe* project that was presented in class last week. Extract the project and open it in BlueJ. Read the classes and run the program (execute the `main` method in class `TicTacToe`). If you understand this program, this lab should be very straightforward.

Step 1: Create a new folder named Lab 11.

Step 2: Download phone-GUI.zip to your Lab 11 folder.

Step 3: Extract the *phone-GUI* project from the zip file.

Step 4: Launch BlueJ and open the *phone-GUI* project.

Exercise 1 - Exploring the Project

This project contains a mock-up of a simple interface for a telephone.

Step 1: Compile the project and execute the `main` method. When you click a key in the GUI , a message is printed in the BlueJ Terminal window indicating which button you clicked. Close the program. You'll now use the BlueJ editor to examine the program's classes.

Step 2: Open class `PhoneGUI` and read the `main` method. We've seen similar code before. The user interface consists of a panel (an instance of `KeypadPanel`), which is placed in a `JFrame`.

Step 3: Open class `KeypadPanel`. This class is a subclass of `JPanel`. The constructor creates 12 buttons, adds them to the panel, then sets the preferred size of the panel. Two things are new, compared to last week's lab:

- In previous GUI programs, we've let the panel's default *layout manager* determine where to place the components in the panel. For this GUI, we want the buttons to be laid out in a rectangular grid. To do this, we have to change the panel's layout manager to a *grid layout* that has four rows and three columns. That's done by this statement in the `KeypadPanel` constructor:

  ```
  setLayout(new GridLayout(4, 3));
  ```

  When we add `JButton` objects to the panel, the first three buttons are placed, left-to-right, on the top row, the next three buttons are placed on the second row, etc.

  More information about class `GridLayout` can be found here:

  http://docs.oracle.com/javase/7/docs/api/java/awt/GridLayout.html

- In previous GUI-based programs, we've had a separate listener class for every button. Here, we create a single instance of `ButtonListener`, and register that object as the listener for all of the `JButton` objects in the keypad. If all buttons share the same listener, how can the listener's `actionPerformed` method determine which button was clicked? We'll answer that question in the next step.

Step 4: Open class `ButtonListener` and read the `actionPerformed` method. When a button is clicked, the `JButton` creates an instance of class `ActionEvent`, and passes a reference to that object to `actionPerformed`.

`ActionEvent` objects provide a method named `getSource`, which returns a reference to the object where the event occurred; that is, the button that was clicked. The return type of `getSource` is `Object`, so we have to cast the value returned by the method to `JButton`:

```
JButton button = (JButton) event.getSource();
```

`JButton` objects provide a method named `getText`, which returns a `String` containing the button's text label. Method `actionPerformed` calls this method on the `JButton` that was returned by `getSource`, to get the label of the button that was clicked. This is the information that is displayed in BlueJ's terminal window.

Exercise 2 - Displaying the Phone Number

We're going to modify the GUI so that the phone number is displayed in the GUI as keys in the keypad are clicked.

Step 1: Create a subclass of `JPanel` named `PhonePanel`. This panel will contain the `JLabel` where the phone number will be displayed, and an instance of the `KeypadPanel` class you explored in Exercise 1.

We'd like the `JLabel` positioned at the top of the panel, with the `KeypadPanel` below it. The easiest way to do this is for the `PhonePanel` constructor to call `setLayout` to change the panel's layout manager to a *border layout*, in the same way that the `KeypadPanel` constructor

calls `setLayout` to changes that panel's layout manager to a grid layout. More information about class `BorderLayout` can be found here:

http://docs.oracle.com/javase/7/docs/api/java/awt/BorderLayout.html

When you add components to a panel that uses a border layout, you must specify the region where the component will be placed. There are five regions, which are identified by five constants: `BorderLayout.NORTH`, `BorderLayout.SOUTH`, `BorderLayout.EAST`, `BorderLayout.WEST` and `BorderLayout.CENTER`. For example, this statement in the `PhonePanel` constructor will add a `JLabel` to the panel's west region:

```
add(new JLabel("Hello!"), BorderLayout.WEST);
```

In the `PhonePanel` constructor,

- change the panel's layout manager to border layout;
- create a new `JLabel` that is initialized with the string `" "` (one space between the quotation marks);
- place the `JLabel` in the north region of the panel;
- create an instance of `KeypadPanel` and place it in the south region of the `PhonePanel`.

Step 2: Open class `PhoneGUI`. Replace the code that adds a new `KeypadPanel` object to the `JFrame` with statements that add a `PhonePanel` object to the frame.

Compile the project and execute the `main` method. The blank label should be now displayed in the GUI, above the keypad, but no numbers will be displayed in the label when keys are clicked. We'll work on that in the next step.

Step 3: In class `PhonePanel`, define a method named `refreshUI` that is passed a `String`:

```
public void refreshUI(String text)
```

This method should update the `JLabel` by appending the character string in `text` to the string that is currently displayed in the `JLabel`. Class `JLabel` provides `setText` and `getText` methods. More information about these methods can be found here:

http://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html

Step 4: In class `ButtonListener`, modify `actionPerformed` so that is calls the `refreshUI` method, passing it the text from the button that was clicked.

To do this, the `ButtonListener` object must store a reference to the `PhonePanel` object. One way to do this is to pass the reference to the `PhonePanel` to the `ButtonListener` constructor. You will also need to modify the `KeypadPanel` constructor and the `PhonePanel` constructor.

Step 5: Compile the project and execute the `main` method. As you click keys in the keypad, the

phone number should appear in the label.

Exercise 3 - Adding a Clear Button

Now we'll add a button that clears the phone number from the GUI.

Step 1: Modify the `PhonePanel` constructor to create a button labelled "Clear" This button should be located in the GUI below the `JLabel` and above the `KeypadPanel`.

Step 2: In class `PhonePanel`, define a method that clears the phone number by setting the label's text string to `" "` (one space between the quotation marks).

Step 3: Define a new class called `ClearButtonListener`. Register an instance of this class as the `ActionListener` for the "Clear" button.

Step 4: In class `ClearButtonListener`, define an `actionPerformed` method that calls the method that clears the phone number.

Step 5: Compile the project and execute the `main` method. As you click keys in the keypad, the phone number should appear in the label. Verify the the "Clear "button clears the phone number.

Wrap-Up

1. With one of the TAs watching, demonstrate your *phone-GUI* project. The TA will review your solutions to the exercises, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/sign-out sheet.

2. The next thing you'll do is package the *phone-GUI* project in a *jar* (Java archive) file named phone-GUI.jar. To do this:

    2.1. From the menu bar, select Project > Create Jar File... A dialog box will appear. Click the Include source and Include BlueJ project files check boxes. A check-mark should appear in each box. Click the down-arrow in the Main class field and select PhoneGUI from the list of class names in the drop-down menu.

    2.2. Click Continue. A dialog box will appear, asking you to specify the name for the jar file. Type phone-GUI or select the BlueJ icon named phone-GUI in the list of files. **Do not use any other name for your jar file** (e.g., lab11, my_project, etc.).

    2.3. Click Create. BlueJ will create a file named phone-GUI that has extension .jar. (Note: you don't type this extension when you specify the filename in Step 2.2; instead, it's automatically appended when the jar file is created.) The jar file will contain copies of the Java source code and several other files associated with the project. (The original files in your phone-GUI folder will not be removed).

3. Before you leave the lab, log in to cuLearn and submit phone-GUI.jar. To do this:

    3.1. Click the Submit Lab 11 link. A page containing instructions and your

submission status will be displayed. After you've read the instructions, click the Add submission button. A page containing a File submissions box will appear. Drag phone-GUI.jar to the File submissions box. Do not submit another type of file (e.g., a .java file, a RAR file, a .txt file, etc.)

3.2. After the icon for the file appears in the box, click the Save changes button. At this point, the submission status of your file is "Draft (not submitted)". If you're ready to finish submitting the file, jump to Step 3.4. If you instead want to replace or delete your "draft" file submission, follow the instructions in Step 3.3.

3.3. You can replace or delete the file by clicking the Edit my submission button. The page containing the File submissions box will appear.

3.3.1. To overwrite a file you previously submitted with a file having the same name, drag another copy of the file to the File submissions box, then click the Overwrite button when you are told the file exists ("There is already a file called..."). After the icon for the file reappears in the box, click the Save changes button.

3.3.2. To delete a file you previously submitted, click its icon. A dialogue box will appear. Click the Delete button., then click the OK button when you are asked, "Are you sure you want to delete this file?" After the icon for the file disappears, click the Save changes button.

3.4. Once you're sure that you don't want to make any changes to the project you're submitting, click the Submit assignment button. A Submit assignment page will be displayed containing the message, "Are you sure you want to submit your work for grading? You will not be able to make any more changes." Click the Continue button to confirm that you are ready to submit your lab work. This will change the submission status to "Submitted for grading".

Extra Practice

Currently, this program has classes that implement a GUI, but there's no underlying model. To remedy this, define a new class named `Phone`. Instances of this class should store a contact list of phone numbers (use an `ArrayList`). Add a button to the GUI that lets you to scroll through the contact list, displaying the numbers in the list one at a time. (When you reach the end of the list, wrap around to the front.) Add a button to the GUI that adds the currently displayed phone number to the contact list, if it's not already there.