Summary

Today you will get practice in writing interactive programs using dialog boxes. Such programs are referred to as Java applications containing a graphical component, or Graphical User Interfaces (GUIs). Most of the programs we have developed so far display output in the command window. However, many applications use windows or dialog boxes to display output. Web browsers display web pages in their own windows. Typically, dialog boxes are windows in which programs display important messages to the users. Class JOptionPane provides pre-built dialog boxes that enable programs to display windows containing messages. In this practical assignment you will create a simple Graphical User Interface (GUI) program using dialog boxes. Your will also continue practicing with Strings and random numbers. Then, using the “git add”, “git commit”, and “git push” commands you will upload your program to your Git repository.

Review the Textbook

To learn more about designing dialog boxes, you need to forward to Section 6.6 of your book and read about designing dialog boxes using JOptionPane class. Additionally, you may read Chapters 3.9 and 4.7–4.9 to learn how to create frames, panels, text fields and buttons using JFrame, JLabel, JButton and JTextField classes. We may discuss frames and panels in November, if we have time. Also, you should read Section 3.8 to learn more about the wrapper classes.

Warm-Up Exercise: Study the given program

The example below shows a program that creates dialog boxes and displays messages using them. In the shared repository, under “practical4” directory, find a program called Practical4.java and copy it to your “practical4” folder in your own repository. Open the program and change the comment header with your own name and today’s date. Pay close attention to the lines highlighted in red bold-face type as they are described below.

Note that the first red line indicates that the program uses class JOptionPane from package javax.swing. This package contains many classes that help you create graphical user interfaces (GUIs). GUI components facilitate data entry by a program’s user and presentation of outputs to the user. The second red line calls JOptionPane method showMessageDialog to display dialog box containing a message. This method requires two arguments. The first argument helps the Java application determine where to position the dialog box. If the first argument is null, the dialog box is displayed at the center of your screen. The second argument is the String to display in the dialog box. The third red line uses JOptionPane method showInputDialog to display an input dialog containing a prompt and a field in which the user can enter text. Method showInputDialog’s
argument is the prompt that indicates what the user should enter. The user types the characters in the text field and clicks OK button to return the String to the program. Method showInputDialog returns a String containing the characters typed by the user. We store the String in the variable name. Finally, method Integer.parseInt returns an int data type corresponding to the value that gets stored as a String. Compile and execute this program. Integer is what is called a wrapper class, which represents a primitive int data type.

import javax.swing.JOptionPane;
public class Practical4
{
    public static void main ( String[] args )
    {
        int age = 0;
        int age1 = 20;
        Random random = new Random();
        // display a dialog with a message
        JOptionPane.showMessageDialog( null, "Let's get you a new identity!" );
        // prompt user to enter the first name
        String name = JOptionPane.showInputDialog(" What is your first name?" );
        //create a message with the modified first name
        String newName = "Your new first name is "+name+"ka";
        //display the message with the modified user's name
        JOptionPane.showMessageDialog(null, newName);
        // prompt user to enter the age
        age = Integer.parseInt(JOptionPane.showInputDialog("Enter your age"));
        // generate a random number from 0 to age and add it to age1
        age1 += random.nextInt(age);
        // modify the age
        String newAge = "Your age is: "+age1;
        // display a message with the new age
        JOptionPane.showMessageDialog(null, newAge);
    } //end main
} //end class Practical4

Assignment: Write your own program

Modify the Practical4 program that was provided for you by completing the following steps:

• Change the user’s interaction (questions/answers) to your own.
Following the given examples, add at least three more questions and modified answers that are your own.

After you have written and tested your program, do not forget to submit your program to your repository hosted by the Bitbucket.

General Guidelines for Practical Sessions

- **Experiment!** Practical sessions are for learning by doing without the pressure of grades or “right/wrong” answers. So try things! The best way to learn is by trying things out.

- **Submit Something.** Your grade for this assignment is a “checkmark” indicating whether you did or did not complete the work and submit something to the Bitbucket repository using the “git add”, “git commit”, and “git push” commands.

- **Practice Key Laboratory Skills.** As you are completing this assignment, practice using the `gvim` text editor and the Ubuntu terminal until you can easily use their most important features. Additionally, ask a teaching assistant or a course instructor to teach you some of the advanced features of `gvim` and the terminal, thereby helping you to work more effectively.

- **Try to Finish During the Class Session.** Practical exercises are not intended to be the equal of the laboratory assignments. If you are simply a slow typist, I’ve given you until the end of the day, but ideally you should upload a file, even a non-working one, by the end of the class period. You also should ensure that, for this assignment and all subsequent assignments, you can confidently upload files to your Git repository during the practical session.

- **Help One Another!** If your neighbor is struggling and you know what to do, offer your help. Don’t “do the work” for them, but advise them on what to type or how to handle things. If you are stuck on a part of this practical session and you could not find any insights in either your textbook or online sources, formulate an intelligent question to ask your neighbor, a teaching assistant, or a course instructor. Try to strike the right balance between asking for help when you cannot solve a problem and working independently to find a solution.

- **Update Your Repository Often!** You should add, commit, and push your updated files each time you work on them, always including descriptive messages about each code change.

- **Review the Honor Code Policy on the Syllabus.** Remember that while you may discuss programs with other students in the course, programs that are nearly identical to, or merely variations on, the work of others will be taken as evidence of violating the Honor Code.