CP2406 Programming-II: Practical-9

This document has been prepared by Dr. Dmitry Konovalov for James Cook University. Updated 3 August 2017.

Instructions for on-campus version:

- **WHEN**: Teaching week **#9** at JCU; Teaching week **#8** at JCUS/JCUB (scheduled after lectures)
- **DURATION**: two hours
- **ATTENDANCE**: compulsory (students must attend). You (student) ***must sign/initial the attendance sheet*** provided by your instructor.
- **MARKING [1 mark]**: Complete the tasks from this practical and show the completed tasks to your instructor. Each completed practical is awarded **ONE participation mark** towards the participation assessment component of this subject.
- **EARLY SUBMISSIONS**: You are encouraged to attempt (and complete) some or all of the following tasks ***before*** attending the practical session.
- **LATE SUBMISSIONS**: You may finish the following tasks in your own time and then show your completed tasks during the following week practical. **The main intent here is to encourage you as much as possible to complete all practicals**. ***If you are late by more than one week***, you will need a valid reason for your instructor to be awarded the marks.

Instructions for off-campus/online version

- WHEN: Teaching week #2 at JCU; Teaching week #1 at JCUS/JCUB (scheduled after lectures)
- DURATION: two hours
- ATTENDANCE: compulsory (students must attend). You (student) ***must submit your work to the appropriate CP2408 Practical assessment dropbox on LearnJCU***.
- MARKING: Complete the tasks from this practical and your instructor will provide feedback via assessment rubrics.

TASK-3: Chapter-15 Advanced GUI Topics: Debugging Exercises [10-20 min]

Debugging Exercises

1. Each of the following files in the Chapter15 folder of your downloadable student files has syntax and/or logic errors. In each case, determine the problem and fix the program. After you correct the errors, save each file using the same filename preceded with *Fix*. For example, DebugFifteen1.java will become **FixDebugFifteen1.java**.

   a. DebugFifteen1.java
   b. DebugFifteen2.java
   c. DebugFifteen3.java
   d. DebugFifteen4.java

- Fork https://github.com/CP2406Programming2/cp2406_farrell8_ch15 and then create the corresponding IntelliJ project. Work your way through all of them until all compiling errors are fixed. Run each class [that contains the **main**-function] to see what it does. Commit and push your solution to your github account.

TASK-4: Chapter-15 Advanced GUI Topics: Programming Exercises [10-20 min]

- Complete any ***four*** exercises from the following list, ***or as directed by your instructor.***
- **Help**: See textbook, and/or https://github.com/CP2406Programming2/cp2406_farrell8_ch15 (or your own fork where you fixed all compiling errors).
- **MORE HELP:** https://github.com/CP2406Programming2/cp2306_farrell8_prac_solutions/tree/master/Chapter15/ProgrammingExercises .

Programming Exercises

1. Create a `JFrame` and set the layout to `BorderLayout`. In each region, place a `JButton` that displays the name of a classic movie that has the region name in its title. For example, the east button might indicate the movie *East of Eden*. When the user clicks the button, display the year of the movie's release and the name of one of its stars. Save the file as **JMovieFrame.java**.

2. Create an educational program for children that distinguishes between vowels and consonants as the user clicks buttons. Create 26 `JButtons`, each labeled with a different letter of the alphabet. Create a `JFrame` to hold three `JPanels` in a two-by-two grid. Randomly select eight of the 26 `JButtons` and place four in each of the first two `JPanels`. Add a `JLabel` to the third `JPanel`. When the user clicks a `JButton`, the text of the `JLabel` identifies the button's letter as a vowel or consonant, and then a new randomly selected letter replaces the letter on the `JButton`. Save the file as **JVowelConsonant.java**.

3. Create a `JFrame` that holds five buttons with the names of five different fonts. Include a sixth button that the user can click to make a font larger or smaller. Display a demonstration `JLabel` using the font and size that the user selects. Save the file as **JFontSelector.java**.

4. Create a `JFrame` that uses `BorderLayout`. Place a `JButton` in the center region. Each time the user clicks the `JButton`, change the background color in one of the other regions. Save the file as **JColorFrame.java**.

5. Create a `JFrame` with `JPanels`, a `JButton`, and a `JLabel`. When the user clicks the `JButton`, reposition the `JLabel` to a new location in a different `JPanel`. Save the file as **JMovingFrame.java**.

6. Create a class that extends `JPanel` and whose constructor accepts two colors, a `Font`, and a `String`. Use the colors for the background and foreground of the panel and display the string using the font parameter. Create an application named `JPanelDemo`. Use `GridLayout` to display four sample panels. Save the files as **JFlexiblePanel.java** and **JPanelDemo.java**.

7. Write an application that lets you determine the integer value returned by the `InputEvent` method `getModifiers()` when you click your left, right, or—if you have one—middle mouse button on a `JFrame`. Save the file as **JLeftOrRight.java**.

8. a. Search the Java Web site for information on how to use a `JTextArea`. Write an application for the WebBuy Company that allows a user to compose the three parts of a complete e-mail message: the "To:", "Subject:", and "Message:" text. The "To:" and "Subject:" text areas should provide a single line for data entry. The "Message:" area should allow multiple lines of input and be able to scroll if necessary to accommodate a long message. The user clicks a button to send the e-mail message. When the message is complete and the Send button is clicked, the application should display "Mail has been sent!" on a new line in the message area. Save the file as **JEMail.java**.

   b. Modify the `JEMail` application to include a Clear button that the user can click at any time to clear the "To:", "Subject:", and "Message:" fields. Save the file as **JEMail2.java**.

9. a. Create an application that uses a graphic interface to capture employee data and writes that data to a random access output file. The data required for each employee includes an employee ID number from 1 through 99 inclusive, the first and last names of the employee, and the employee's hourly pay rate. Allow the

user to enter data one record at a time and to click a button to save each record. Save the class as **CreateRandomEmployeeFile.java**.

b. Create an application that allows the user to enter an employee ID number. When the user clicks a button, display all the stored data for the employee. Save the file as **ReadRandomEmployeeFile.java**.

10. Create a `JFrame` for Java Junior College. Use menus to allow the user to access information about different campuses, major fields of study offered, and activities. Include at least two options in each menu. Save the file as **JavaJuniorCollege.java**.

=== END OF THIS PRACTICAL ☺ ===