CP2406 Programming-II: Practical-2

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 **#3** at JCU; Teaching week **#2** 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-1: Chapter-2 Debugging Exercises [15-30 min]

Debugging Exercises

1. Each of the following files in the Chapter02 folder of your downloadable student files has syntax and/or logic errors. In each case, determine the problem and fix the application. After you correct the errors, save each file using the same filename preceded with *Fix*. For example, DebugTwo1.java will become **FixDebugTwo1.java**.

   a. DebugTwo1.java
   b. DebugTwo2.java
   c. DebugTwo3.java
   d. DebugTwo4.java

- The above description of the debugging exercises is from this subject textbook.
- Clone https://github.com/CP2406Programming2/cp2406_farrell8_ch02 and then create the corresponding IntelliJ project.
- **Productivity tip**: go the https://github.com/CP2406Programming2 and clone all the chapters into your github account, so you could use them with the rest of the pracs.
- Try to build this project. IntelliJ will display ***compiling*** errors and highlight the offending sections of the java code.
- 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-2: Chapter-2 Programming Exercises [10-20 min]

- Complete any ***two*** exercises from the following list, ***or as directed by your instructor.***
- **Help**: https://github.com/CP2406Programming2/cp2406_farrell8_ch02 (or your own fork).
- **MORE HELP:** https://github.com/CP2406Programming2/cp2306_farrell8_prac_solutions/tree/master/Chapter02/ProgrammingExercises .

4. a. Write a Java class that declares a named constant to hold the number of quarts in a gallon (4). Also declare a variable to represent the number of quarts needed for a painting job, and assign an appropriate value—for example, 18. Compute and display the number of gallons and quarts needed for the job. Display explanatory text with the values—for example, `A job that needs 18 quarts requires 4 gallons plus 2 quarts`. Save the class as **QuartsToGallons.java**.

   b. Convert the `QuartsToGallons` class to an interactive application. Instead of assigning a value to the number of quarts, accept the value from the user as input. Save the revised class as **QuartsToGallonsInteractive.java**.

5. a. Write a Java class that declares named constants to represent the number of kilometers (1.852) and the number of miles (1.150779) in a nautical mile. Also declare a variable to represent a number of nautical miles and assign a value to it. Compute and display, with explanatory text, the value in kilometers and in miles. Save the class as **NauticalMiles.java**.

   b. Convert the `NauticalMiles` class to an interactive application. Instead of assigning a value to the nautical miles variable, accept it from the user as input. Save the revised class as **NauticalMilesInteractive.java**.

6. a. Write a class that declares a variable named `inches`, which holds a length in inches, and assign a value. Display the value in feet and inches; for example, 86 inches becomes 7 feet and 2 inches. Be sure to use a named constant where appropriate. Save the class as **InchesToFeet.java**.

   b. Write an interactive version of the `InchesToFeet` class that accepts the inches value from a user. Save the class as **InchesToFeetInteractive.java**.

7. Write a class that declares variables to hold your three initials. Display the three initials with a period following each one, as in J.M.F. Save the class as **Initials.java**.

8. Meadowdale Dairy Farm sells organic brown eggs to local customers. They charge $3.25 for a dozen eggs, or 45 cents for individual eggs that are not part of a dozen. Write a class that prompts a user for the number of eggs in the order and then display the amount owed with a full explanation. For example, typical output might be, "You ordered 27 eggs. That's 2 dozen at $3.25 per dozen and 3 loose eggs at 45 cents each for a total of $7.85." Save the class as **Eggs.java**.

9. a. The Huntington Boys and Girls Club is conducting a fundraiser by selling chili dinners to go. The price is $7 for an adult meal and $4 for a child's meal. Write a class that accepts the number of each type of meal ordered and display the total money collected for adult meals, children's meals, and all meals. Save the class as **ChiliToGo.java**.

   b. In the previous example, the costs to produce an adult meal and a child's meal are $4.35 and $3.10, respectively. Modify the `ChiliToGo` program to display the total profit for each type of meal as well as the grand total profit. Save the class as **ChiliToGoProfit.java**.

10. Write a class that calculates and displays the conversion of an entered number of dollars into currency denominations—*20*s, *10*s, *5*s, and *1*s. Save the class as **Dollars.java**.

11. Write a program that accepts a number of minutes and converts it both to hours and days. For example, 6000 minutes equals 100 hours and equals 4.167 days. Save the class as **MinutesConversion.java**.

12. Travel Tickets Company sells tickets for airlines, tours, and other travel-related services. Because ticket agents frequently mistype long ticket numbers, Travel Tickets has asked you to write an application that indicates invalid ticket number entries. The class prompts a ticket agent to enter a six-digit ticket number. Ticket numbers are designed so that if you drop the last digit of the number, then divide the number by 7, the remainder of the division will be identical to the last dropped digit. This process is illustrated in the following example:

Step 1	Enter the ticket number; for example, 123454.
Step 2	Remove the last digit, leaving 12345.
Step 3	Determine the remainder when the ticket number is divided by 7. In this case, 12345 divided by 7 leaves a remainder of 4.
Step 4	Assign the Boolean value of the comparison between the remainder and the digit dropped from the ticket number.
Step 5	Display the result—`true` or `false`—in a message box.

Accept the ticket number from the agent and verify whether it is a valid number. Test the application with the following ticket numbers:

- 123454; the comparison should evaluate to `true`.
- 147103; the comparison should evaluate to `true`.
- 154123; the comparison should evaluate to `false`.

Save the program as **TicketNumber.java**.

=== END OF THIS PRACTICAL ☺ ===