CSCI1530 Computer Principles and Java Programming 2014-2015 Second Term
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Due date: 6 February 2015 (Fri) **Assignment 2** Full mark: 100
Expected normal time spent: 5 hours

Aim:
1. use of class **`Scanner`** to obtain keyboard input from console/ terminal.
2. use of class **`JOptionPane`** to create dialogues (Graphical User Interface/ GUI.)
3. practise declaring variables and performing calculations.
4. create and manage a NetBeans project with multiple Main classes.

Exercise Zero: Background Knowledge:

1. Use the Java Standard API Class **`JOptionPane`** to create dialogues in a Java Application. Here is an example **`SampleDialogue.java`** for your reference:

```
import java.util.Scanner;
import javax.swing.JOptionPane;

class SampleDialogue
{
    public static void main(String[] args)
    {
        String answer;
        answer = JOptionPane.showInputDialog("This is a Q&A window.");

        // parse/ convert the answer (which is a String) to a number (a double)
        double number;

        number = Double.parseDouble(answer);

        JOptionPane.showMessageDialog(null,
               "This is a pop-up window.  You typed " + number + ", isn't it?");

        // connect to the keyboard using class Scanner and System.in
        Scanner keyboard = new Scanner(System.in);

        // ask questions and get inputs via the console/ terminal
        System.out.print("Type a line and <Enter>: ");
        String aLineOfText = keyboard.nextLine();
        System.out.print("Type a number and <Enter>: ");
        int anIntegerNumber = keyboard.nextInt();

        System.out.println("Got " + anIntegerNumber + " after " + aLineOfText);
        // program terminates here after the last user interaction
    }
}
```

2. Explanations: both **`JOptionPane`** and **`Double`** are classes provided in the Java Application Programming Interface **應用程式介面** (API). Without discussing much about them, first of all, let's try using them!

3. Both classes **`JOptionPane`** and **`Double`** provide us some methods (actions!) that we can make use of. We are going to send messages **`JOptionPane.showInputDialog()`**, **`JOptionPane.showMessageDialog()`** and **`Double.parseDouble()`** in order to achieve our goal of implementing the following system.

4. Class **`Scanner`** and **`System.in`** help us getting text and number inputs from the console.

Exercise A) Using JOptionPane Tasks:

1. Create a N E W NetBeans Project **`JHealth`** with a package named **`tool`** and a new Java class **`BMI`**, i.e., **Create Main class** **`tool.BMI`**. Type your name, student ID and declaration statement clearly at the top comment block of the file **`BMI.java`**.

```
/**
 * CSCI1530 Assignment 2 Exercise A: BMI Dialogue
 * Aim: use of class JOptionPane to create dialogues
 *
 * Declaration: . . . <please refer to Assignment 1 for our requirements>
 * Student Name: xxx <fill in yourself>
 * Student ID  : xxx <fill in yourself>
 *
 */
```

2. The program asks for name, weight and height of a user using "Input" dialog boxes. We should get all the three answers from the user before proceeding to task number 3.

3. Because the user types some "text" (String) in respond to the weight and height questions, we have to "parse" (i.e., analyze) the textual inputs and convert the "text" to numbers using the example code demonstrated in the **`SampleDialogue`** example. In case the conversion fails, our Java program will be *terminated exceptionally*. This is an expected and accepted behaviour in this exercise.

4. It then calculates the Body Mass Index (BMI) of the user using the following formula:

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2}$$

where weight is in kilogram, height is in metre. We accept division-by-zero scenario.

5. The program displays the result in another "Message" dialog box. Then it terminates after the user clicking on the Ok button on the last dialog box.

6. **Please refer to our sample program for details such as the messages on the dialogues.**

7. We **need not validate** (check) the input of the user. The user is assumed to be rational and cooperative.

Exercise B Using Scanner Tasks:

1. In the SAME NetBeans Project **`JHealth`** and package **`tool`**, create another new class **`HeartRate`**, i.e., **create a new Java class** **`HeartRate`** under the existing package **`tool`** (pick from the drop-down list.) Type your name, student ID and declaration statement clearly at the top comment block of the file **`HeartRate.java`**.

```
/**
 * CSCI1530 Assignment 2 Exercise B: Heart Rate Training Zone Calculator
 * Aim: use of class Scanner to obtain user input
 *  ...
 */
```

2. The program asks for name (a line of text) and age (an integer) of a user using class **`Scanner`** and **`System.in`**. We should get all the answers from the user before proceeding.

3. It then calculates the Heart Rate Training Zone of the user using the following formula:

   a) Maximum heart rate (beat per minute) = 220 – age; where 220 is a magic number/ constant

   b) Target heart rate zone = 70-80% of the Maximum heart rate; again the range is fixed

   c) Lower target heart rate = Maximum heart rate x 0.7

   d) Upper target heart rate = Maximum heart rate x 0.8

   e) **Example usage session:**

```
Texts in blue are user inputs followed by <Enter>, pay attention to the spaces
12345678901234567890123456789012345678901234567890123456789012345678901234567890 Ruler for reference
Name: Peter PAN
Age: 25
Maximum heart rate = 220 - 25 = 195bpm
Lower target heart rate = 195 * 0.7 = 136.5bpm
Upper target heart rate = 195 * 0.8 = 156.0bpm
Target heart rate zone = 136.5 -> 156.0bpm
12345678901234567890123456789012345678901234567890123456789012345678901234567890 Ruer for reference
```

4. **There is NO sample program for this exercise. Take the above sample usage session as a guide for input/ output screen formatting.**

5. We **need not validate** (check) the input of the user. The user is assumed to be rational and cooperative. Let **`System.out.print()`** decide the number of decimal places in the printouts, i.e., we do not specify any format and we accept the printouts as is.

Running and Submission (for both Exercises A and B) :

1. Since there are two Main classes in the same project, we shall "Run File (Shift-F6)" to test run part A and part B respectively. Run Project will always run the default Main Class defined in the Project Properties → Run.

2. Zip and Submit the whole NetBeans project as **`JHealth.zip`**.

Marking Scheme and Notes:

1. The submitted program should be free of any typing mistakes, compilation errors and warnings.

2. Comment/remark, indentation, style are under assessment in every programming assignments unless specified otherwise. This program gives you an example of a well-formatted source file. Variable naming, proper indentation for code blocks and adequate comments are important.

3. Marker will test your programs vigorously with various inputs. You should test run your own programs properly with different user input scenarios.

4. Remember to do your submission before 6:00 p.m. of the due date. No late submission would be accepted.

5. If you submit multiple times, **ONLY** the content and time-stamp of the **latest** one would be counted. Be reminded to click "Submit"! We ONLY take into account the last submission.

University Guideline for Plagiarism

Attention is drawn to University policy and regulations on honesty in academic work, and to the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details may be found at http://www.cuhk.edu.hk/policy/academichonesty/. With each assignment, students will be required to submit a statement that they are aware of these policies, regulations, guidelines and procedures.

Faculty of Engineering Guidelines to Academic Honesty

MUST read: http://www.erg.cuhk.edu.hk/erg-intra/upload/documents/ENGG_Discipline.pdf
(VPN required if you are not in the CUHK network)
OR this link: http://www.cse.cuhk.edu.hk/csci1530/ENGG_Discipline.pdf