CMT219 : JAVA EXERCISES

Exercise Set: 1

Learning Outcomes

After completing these exercises, you should be able to ...

- manipulate primitive types in Java
- import a class from a package in the Core API
- read user input
- use if-then-else, switch statements and loops in an application
- use the Java API Documentation to find appropriate classes and methods
- carry out some simple debugging

... and understand what is meant by ...

- a nested loop

Primitive Types

1. Download the source file **CastTest.java**. This application declares an **int** variable with the value 150 and outputs the result of **casting** it to a few other **primitive types**. After running the application to check that its output is as you expect, edit it to output the result of casting it to the primitive types **boolean**, **byte**, **char** and **short**.

   Which of these casts is Legal in Java?

2. Write an application **FloatingPointTest** in which you:

   (a) Assign a **float** variable the value **1.36f** and a **double** variable the value **1.36**. Output the result of performing the equivalence operator on the two values.

   (b) Assign a **float** variable the value **0.1f** and add he value **0.6f** to it. Output the result.

   (c) Store the value **1 / 49** in a **float** variable and output the result of multiplying by 49. Repeat using a **double** variable.

   Are the results what you expected?

User Input, If-Then-Else and Switch

3. Download the source file **RewriteNumber.java**. This application asks the user to enter an integer between 0 and 9 and then prints that number to standard output. Edit this application so that:

(a) the correct package is imported to fix the compilation error.

(b) instead of repeating the numerical value (e.g. 0, 1, ..., 9) it prints out the English word corresponding to that value (e.g., 'one', 'two',..., 'nine').

Try answering part (b) in each of the following ways:

(a) by using a chain of **if** and **if else** statements.

(b) by using a **switch** statement.

Which was the most convenient approach for this application?

Loops

4. Write an application that contains a sequence of loops to:

   (a) Produce the output:

   1
   2
   3
   4
   5

   (b) Produce the output:

   $$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,$$

   (c) Produce the output:

   $$15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,$$

   (d) Produce the output:

   $$5, 10, 15, 20, 25, 30, 35, 40, 45,$$

5. Write an application that outputs the square roots of the integers 1 up to 20.
   HINT: Use an appropriate method from the **Math** class.

6. Write an application **PrintStarSquare** that outputs a 10×10 square of asterisks (i.e. $*$) to standard output.

7. A **nested loop** is a loop within another loop. A nested loop can be written in Java by enclosing the inner loop in the curly braces {} of the outer loop. In each iteration of the outer loop, the inner loop will be executed once. Write an application **PrintStarRectangle** that prompts the user to enter the dimensions of a rectangle (i.e., it requests the height h and then the width w) and uses a nested loop to print an h by w rectangle of asterisks to standard output.

Java API Documentation

8. The Java API Documentation documents the classes and packages available in the Java Core API. By using the Java API Documentation to identify relevant fields and/or methods in the class **java.lang.Math**, write an application that outputs:

   (a) The sine of 15.

   (b) $63^{2.5}$.

   (c) The square root of 2498.

   (d) A random number greater or equal to 0 and less than 1.

   (e) π.

Debugging

9. Open SublimeText and enter the following code:

```
public class HelloWorldError
    public static void main( String[] args )
    {
        System.out.println( Hello World );
    }
}
```

   and save it in a file named **HelloWorldError.java**. Before executing the application it needs to be compiled. Open the command line (**cmd.exe** on Windows) and navigate to the directory where you saved **HelloWorldError.java** by using the command **cd** (**c**hange **d**irectory). Attempt to compile the application by typing:

```
> javac HelloWorldError.java
```

   The application is supposed to output the text "Hello World" to **standard output**. However, there are three syntax errors that prevent it from compiling. **javac** will print a list of **compilation errors** to the command line to help you identify where the problems are. Fix the errors to allow the application to be successfully compiled.

10. Once you manage to successfully compile the application, a **HelloWorldError.class** file will be created in the same directory as **HelloWorldError.java**. This **Java class file** contains the **Java bytecode** for the application. You can now execute the application by typing:

```
> java HelloWorldError
```