GENERAL INTRODUCTION TO TUTORIAL AND LAB WORK

There is one 1-hour tutorial and laboratory session each week. It is expected that you will attempt most of the questions in these exercises in the lab class, and you will finish the rest as part of your own personal study.

Assessment
Your attendance and your performance in class are assessable. As such it is important you complete all the work set in the exercises every week.

Important note
You may not have time to do every exercise in the lab class and you can practice on these exercises at home. Check your answers with your tutor.

INTRODUCTION AND JAVA BASICS

The key topic for this week is the basics of programming and the working environment.

TASK

1. Familiarize yourself with the working environment such as operating systems and Java
2. Use Windows/MS-DOS to make directories/folders and to copy files
3. Learn how to declare and initialize variables
4. Code, compile and execute simple Java programs through command-line instructions and *Eclipse.*

SECTION I LABORATORY EXERCISES

Note that **U drive** is your personal disk space on the network allowing you to store data/files. It can be accessed from any computers inside the laboratory. **It is important that you save your data/files inside your U drive to ensure you can retrieve your files again using other computers.**

C or D drive is local drive and is unique to the computer that you are currently working with. Thus, anything you stored inside **C or D drive** will not allow you to retrieve using other computers in the laboratory.

Build workspace folder in your U drive

1. To use the command-line instructions, start it up by **Start > Run,** type **cmd** and then press **OK** button.
2. Change to your **u drive** by typing **u:** after the command prompt and press **Enter** key
3. Build a workspace folder (e.g. comp9103) in your **u drive** to store all your projects and java files for your lab and homework. E.g., type u:>**md comp9103** to build the folder of U:\comp9103.

Introduction to Coding, Compiling and Executing

The process of programming requires 3 steps: coding, compiling and executing.

Step 1: Coding

The first thing you need to do in programming is to write your code into a file. This process is called *coding*. You need to write your code into a text file, that is, a file that contains only plain text. So you will need to use a program like or Notepad or TextPad on a Windows machine. Something like Microsoft Word will not work because it is not a plain text editor, and it adds other things like fonts, symbols, formatting, etc.

For instance:

1. Launch the Notepad editor from the **Start** menu by selecting **Programs > Accessories > Notepad**. In a new document, type your code as below:

```
/* This program simply prints out "Hello, World!" to the
screen */

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

2. Save it into a file with exactly the same name as the class ending with ".java". So in the above case, save the code in a file called *HelloWorld.java*

Note that H and W are capitals because they are capitals in the code; the class name must be followed exactly.

Step 2: Compiling
You have to translate the code you have written into "machine language" so that the computer can understand. This process of translation is called *compiling*.

1. Open "Command" window for typing commands. Such a window is called "command line interpreters". You can do this from the **Start** menu by choosing **Command Prompt** (Windows XP), or by choosing **Run...** and then entering **cmd**.
2. The prompt shows your current directory, change to your workspace folder (e.g. u:\comp9103). E.g., c:> is the current directory, and you need to change it to **u drive** by typing c:>**u:** and press **Enter**; and then change to your folder (e.g. u:\comp9103) by: u:>**cd comp9103** and press **Enter.**
3. At the prompt, type the command

   javac HelloWorld.java and press **Enter.**
4. The compiler has generated a **bytecode** file, HelloWorld.class. At the prompt, type **dir** to see the new file that was generated.

Step 3: Executing
If you have succeeded in compiling, you can finally run or execute your program. To do this, you type `java` followed by name of the class. So in the above case, you would type:
java HelloWorld

Note that when executing (unlike compiling), you do not include the ".java".

Using Eclipse IDE
Eclipse is an Integrated Development Environment (IDE) by IBM. It can be used to create diverse applications and projects. You will practice using IDE (e.g. Eclipse) to do coding, compiling, executing, and later debugging. Please refer to Supplement (2) for more information.

1. Run Eclipse SDK
2. Click FILE->"Switch Workspace…". In the box type **U:\comp9103**
3. Eclipse SDK will be started. [Optional]**Close the welcome window tab by clicking the "x".**
4. Build a new project named as **lab**:
   - **(a) File−>New−>Project**
   - **(b)** Select **"Java Project" −>Next**
   - **(c)** In the box of **"Project name",** type **lab**
   - **(d)** Leave the other options to default value and click **"Next"->"Finish"**
5. Create a new program in the current project named as **GoodMorningWorld.java**
   - **(a) File−>New−>Class**
   - **(b)** In the **"Name"** box, type **GoodMorninWorldl**
   - **(c)** Select the option **"public static void main(String [] args)"**
   - **(d)** Click **"Finish"** to generate the template for the source code of GoodMorningWorld.java
6. Type `System.out.println("Good Morning, World!");` in the main method
7. Compile the program:
   As default, your source code is dynamically compiled as you type.
   For example, if you forgot to type semicolon at the end of the statement, you will see the wriggly line in the editor pointing to the error.
8. Run the program:

(a) Right-click the class "**GoodMorningWorld.java**" in the project to display a context menu
(b) In the menu, select **"Run−>java application"**

Exercise 1: HelloSomeone.java

Write a new program that says "Hello, X!" where *X* is an argument specified by interactive input from command-line argument. An example of this in use would be:

```
> javac HelloSomeone.java
> java HelloSomeone Bob
Hello, Bob!
```

Exercise 2: HelloSomeone.java

Input the argument for this program using Eclipse. Please refer to Supplement 2 (Running a Program with an input argument).

SECTION II HOMEWORK EXERCISES

1. CODING

1) Write a program that asks the length and width of a rectangle from the argument input, and then prints:
   - The area and perimeter of the rectangle
   - The length of the diagonal
2) Remainder is very useful in programming. For instance, suppose today is Wednesday, you have a submission due in 10 days, what is the day of the due day? Write a program to calculate the day for you.

2. TUTORIAL EXERCISES

These tutorial exercises can be done on paper and do not require the use of a computer. Reading the lecture notes and the textbook will reveal the answers for most of these questions.

1) There are three stages in developing Java programs. First, the program is written in human-readable form during a process called *coding*. Then the code is *compiled* and finally it is *executed* by computer.
   a. What should the file name be for a public class named `HelloWorld`?
   b. What full command would you type to compile the code?
   c. How would you execute this program?

2) What is the difference between variable declaration and variable assignment? What is variable initialization?

3) Explain the errors in the following code segments.

```
a. final int X=0;
   int y = 2;
   X = X+y;

b. int x;
   int y = 1;
   x+1 = y+1;

c. int x;
   int y = x+2;
   z = x+y;
```

```
d. String s1="xyz";
   String s2="abc";
   System.out.println(S1,S2);

e. String 9tails="cat";
   System.out.println(9tails);

f. int  this=4, that=5;
   int x = (this+that);
   x = x+1;
```

4) Trace through the following code. Make sure to write down the values of variables when you have carried out each line.

```
int a=5+4;
int b=a*2;
int c=b/4;
int d=b-c;
int e=-d;
int f=e%4;
double g=18.4;
double h=g%4;
int i=3;
int j=i++;
int k=++i;
```

5) Assume `num1=5` and `num2=10`, trace the calculation of Boolean expression `((num1!=5)||(num2==10))&&!(num1==5)` step by step.

6) Calculate the results of the following Boolean operations

```
a. (3<5)&&(5==4+1)

b. (3<5)||(6==5)||(3!=3)

c. (5!=10)&&(3==2+1)||(4<2+5)

d. !(5==2+3)&&!(5+2!=7-5)
```

7) Trace the following code segments, giving the value of all variables after the segment is executed.

```
a. int x = 0;
   int y = 1;
   x = y;
   y = x;

b. int x = 10;
   x = x+10;

c. String title;
   String s = "Get ";
   String t = "Shorty";
   title = s + t;
```