Setting Up the Programming Environment

(adapted from dr. Scott Chen's document)

Visual Studio Code (VSCode) is the recommended Java programming integrated development environment (IDE) for this course.

You may use the following link to the official VSCode resources on Java:

Microsoft VSCode Java Tutorial Page
https://code.visualstudio.com/docs/java/java-tutorial

Setting Up IDE

This section provides detailed instructions on how to set up Visual Studio Code, Java Plugins, and other required tools to support all 3SM4 activities.

The instructions cover the setup details for Windows, MacOS, and Ubuntu/Debian Linux. Please make sure you follow the instructions pertaining to your specific OS. Note though that Java is a platform-independent programming language; the procedure differences will only appear in environment setup. Once you've completed the IDE and environmental setup, all instructions will be identical to all platforms.

I. Preparing Your OS for Visual Studio Code

Windows
VSCode only works on Win 8/10/11. Make sure your OS is up-to-date before proceeding.

MacOS
VSCode only works on MacOS Monetery 10.11 and above. Make sure your OS is up-to-date before proceeding.

Ubuntu/Debian Linux
VSCode works on many Linux distros.
Call `sudo apt update` to make sure your OS is up-to-date before proceeding.

II. Obtaining VSCode

Visit the VSCode download page here: https://code.visualstudio.com/download

Windows

1) Download the Windows VSCode installer that matches your computer hardware.
2) Launch the installer, follow through the installation steps.

MacOS

1) Download the Universal (.zip) version. It works as a standalone application. No installation required.
2) Once downloaded, VSCode can be launched directly from the downloaded zip file. You may optionally keep VSCode on dock for convenience.

Ubuntu/Debian Linux

1) Download the Ubuntu/Debian installer that matches your computer hardware.
2) Launch the installer, follow through the installation steps.

III. Setting Up VSCode with Java Development Kit and Extension Packs

VSCode can be set up with additional plug-ins to help you execute and manage your Java project efficiently. While the installed contents for different OSes are more or less identical, please still read the matching instructions, as Microsoft provides express setup package for some of the OSes.

Windows and MacOS

1) Visit the VSCode Getting Started page here: https://code.visualstudio.com/docs/java/java-tutorial
2) Download and install the Java Coding Pack for your OS.

Linux

1) Install Java Development Kit from here (recommended):
   https://www.oracle.com/ca-en/java/technologies/downloads/#java23
2) Install VSCode Java Extension Pack from here:
   vscode:extension/vscjava.vscode-java-pack
   Or you can also search for the extension pack from within VSCode Extension tab.

IV. Getting Started with Java Programming on VSCode

While this is not the most complete starter guide, it is enough to get you started on coding and learning Java. All the in-class activities assume you have completed the activity in this guide.

1) Launch VSCode.

2) Upon startup, click on *Explorer* icon (top left corner), then click on *Open Folder*.

3) Create a new folder in your favourite *COMPENG workspace root folder*, name it **3SM4_Hello_World**, and open the ***Newly Create Folder***. This folder will be the basis upon which VSCode builds a Java project for you.

   In future labs, you are expected to open the lab-specific repo folders to carry out the activities.

4) Click *Add a New File* icon, and name it "HelloWorld.java". The text editor will immediately show the contents of hello.java, which is now empty.

   Furthermore, a Java Project sub-tab will appear at the bottom of the left-hand side panel, under which you will find your ***2SM4_Hello_World*** project and the single ***HelloWorld.java*** file under it. This indicates that VSCode has constructed a proper Java package for you.

   All files created within the folder will now be part of the project, and are visible to each other.

5) Type the following code into the ***HelloWorld.java*** text content. Note that the class name must match the java file name as per the Java syntax requirement.

```
public class HelloWorld
{
      public static void main(String[] args)
      {
```

```
            System.out.println("Hello World!");
        }
    }
```

6) Save the file.
7) Pay attention to the prompt line immediately above `public static void main(String[] args)` You can easily choose to ***Run*** the main method of this Java project, or ***Debug*** your java program through the VSCode IDE debugger interface.

8) Click Run. Observe **Hello World!** Is printed on the console screen to confirm that Java is set up correctly on your VSCode.

V. Getting Started with Java Debugging in VSCode

We further want to confirm the debugger functionalities for supporting Java Debugging in VSCode.

1) Using the same code from last section, modify the contents in the main method as:

```
public static void main(String[] args)
{
    int a = 3, b = 4;
    int result = a + 2 * b;
    System.out.printf("%d + %d = %d\n", a, b, result);
}
```

2) Set up a breakpoint on the first line of the main method (`int a = 3, b = 4;`) by clicking on the left side of the respective line number. A red dot will be toggled on, indicating the breakpoint is inserted.

3) Go to the floating prompt above the main method, and click on ***Debug***. The VSCode IDE debugger will launch.

4) Upon Debugger launching, your program will immediately stop at the breakpoint. Use the debugger control buttons to step through the program and observe the variable changes on the Variable tab on the left-hand side panel.

(listed from left to right)

Continue	Execute the program at normal speed up to the next breakpoint encounter / end of the program
Step Over	Execute the current statement as a plain statement. If the statement involves a function call, it will be executed in one-shot as part of the plain statement.
Step In	If the current statement involves a function call, the debugger will go into the function, and execute the function implementation statement-by-statement.
Step Out	If the debugger is currently running in stepping mode inside a function, use this button to finish the remaining implementation of the function in one-shot, and return back up the call stack. Upon return, the debugger will resume the stepping mode.
Restart	Restart the program and the entire debugging session.
Stop	Stop the debugger and return to the editor mode.

5) Once confirmed the debugger functionalities, you may end the debugging session.

Important Note: The usage of `System.out.printf()` method is almost identical to `printf()` in C. Most of the output string formatting knowledge you learned in COE2SH4 are fully transferrable.