SYSC 2004
Fall 2018
Lab 0 – Because we have labs before our first lecture

Blue text – Vocabulary terms that you must know and use in all assignments and exams.

Red text – Lessons I want to pay attention to, appreciate; or warnings.

`Courier Font – names of Java classes, objects, variables and methods. Uppercase letters and no-spaces matter!`

Objectives

1. Compile and run an existing program to begin learning how to use a Java IDE
2. Complete a program template to practice some familiar programming tasks common to both object-oriented programming and structured programming

Objective 1

You are going to write the typical first program (**HelloWorld.java)** with the real objective of learning how to use a new software tool, a Java **Integrated Development Environment** (IDE).

- It is suggested that you use the **Netbeans IDE**, although you are welcome to also use Intelli Java. Please do not use BlueJ.

1. **Please read the PowerPoint presentation called "Getting Started with Netbeans" (Posted on CULearn)**.
2. In every IDE, you always start a new program by creating a project. You will need to create a new project for every part of this lab (and all labs, and all assignments).
   - Always start every new program with a New Project.
   - Create a "**JAVA APPLICATION**" project (this will work until we start writing GUIs).
   - Set the project name to be **helloWorld**. (Case of letters matters!)
   - Set the class name to **HelloWorld** (Case of letter matters!)
   - There will be some pre-generated code: Find the package, the class and the main() method. You must develop the expertise to know which code to leave and which code to replace.
3. Insert a simple println() statement within the main() method that prints a simple message to the console.
   ```
   System.out.println("Hello World");
   ```
4. **Run** the program.
5. Purposely create a compilation syntax error – remove a semi-colon or the double-quote at the end of the string.
   - Observe the red-flags that are put in the margins of the editor- window. Also, red-flags will appear (eventually) by the filename in the project-window.

- Try running the program with the compile error – How does the IDE react?
    - Rule of Thumb: Do NOT select Run-Anyway! Learn the difference between a compilation syntax error and a runtime bug.
    - Many students have sadly spent hours wondering why their program crashes by not observing these compilation red flags.
- Fix the error and run the program.
    - **TIP** - Sometimes, you have to save (CTRL-X) to trigger a re-compile.
    - **TIP** – Later, with many interdependent files, files get out of sync. If you are desperate, try selecting **RUN->CLEAN and BUILD project**. Find this option NOW!
    - In the Console Window, right-click and select CLEAR. Do this if you are overwhelmed by too many messages and you can't figure out which messages belong to which execution run.

Self-Assessment (Are you learning to the best of your ability?)

1. Where are your files stored? What files are stored? Use **Windows-Explorer** to search and find these files. Understand the relationship between packages-and-folders and between classnames-and-filenames.
2. Do the following start with uppercase or lowercase?
    - Package
    - Class
    - Method
3. Challenge Question: If you know the answers to the previous question, can you make any educated guess about the code to print messages?

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

`System` starts with an upper-case letter; therefore it is a ____________.

`println` starts with a lower-case letter; therefore it is a _____________.

I'll tell you about `out` later.

Objective 2

You are provided with a program template – a partial-program that you must complete (**Lab0_2.java**) Use the comments in the program template to guide your work. All of the tasks should be familiar from your previous programming courses – they are all about programming with arrays.

Lesson: Java and C have much in common. It is assumed that you are proficient in for-loops, while-loops, conditional statements and array manipulations.

1. Create a new project, called lab0_2.java.
2. Within that project, create a single class called Lab0_2.java
3. Open the posted file on CULearn and use that code (i.e. copy-paste) to over-write the auto-generated code for the main() method.
4. Compile the code, and even run it before making any changes. We call this "working incrementally" or "working in small steps".

Aside: This exercise will also be the first time that you add pre-written code to a new project. You may experience compile-issues. Carefully check the package name and the class name to make sure that they are exactly like the pre-written code.

- Tip: You can change the name of a package or class (or even the project) by "re-factoring".
  - Hover the mouse over the package or class
  - Right-click
  - Select Re-factor
  - Select Rename