

CS 110 Lab Sheet 1 - First Session

Lab cycle starting 2nd October 2019, Normal Deadline: one week after your lab in that cycle.

The point of the first lab sheet is to get you started with the computers in the lab, write some very simple programs, and explore the things that can go wrong and how to fix them.

If you have experience of programming already, and find the exercises too easy, there are exercises at the end of most chapters of notes.

You should be able to finish the tasks here in one session, regardless of your experience. But it's not a competition. Remember the assistants (and me) are here to help.

You are assessed on your completion of the lab tasks - one task from every lab sheet must be signed off by the postgraduates or me and this will count towards the assessment for the module. In this lab sheet, the final task (number 8) is assessed.

This lab task only has one week for you to complete it. *Normally, you will be given two weeks worth of sessions to complete a task and any time you choose to spend outside labs - remember, you only have four scheduled hours per week but you should be spending about 10 hours per week on this module. In exceptional circumstances, we will extend the deadline for another week - talk to us if you feel you need to do this for a particular lab sheet.*

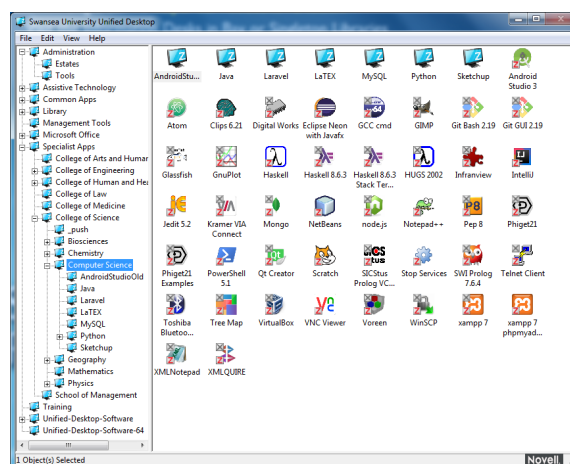
Use The Command Line Tools

For this exercise I want you to use the command line tools - later you can use tools like Netbeans, IntelliJ and Eclipse, but I want everyone to understand the basic process involved in writing Java.

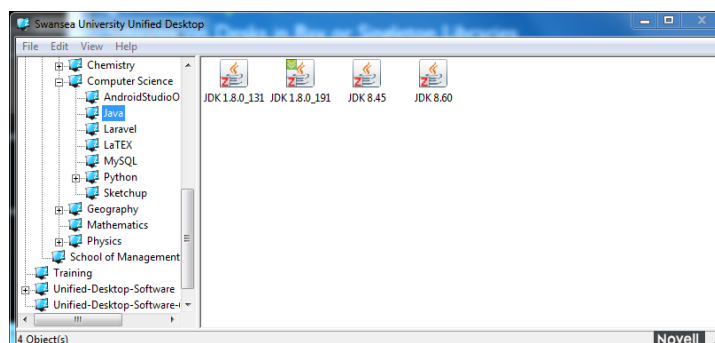
Stage 1: Getting Started and Some Possible Irritating Nonsense

Please remember that when 150 people log in for the first time and start installing software for the first time, things can be a bit slow.

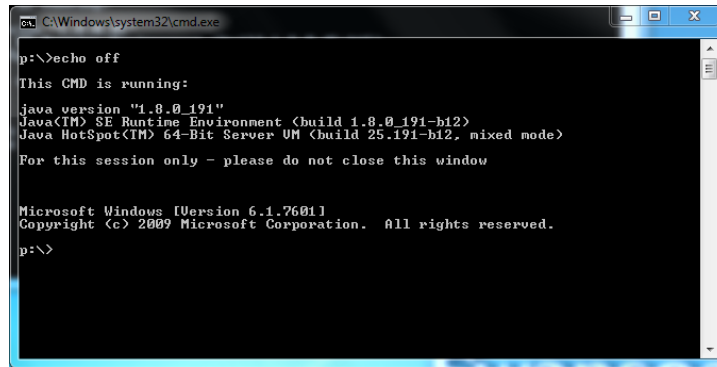
1. **Login** Your first task is to log in, using your University ID and password.
2. **Unified Desktop** Shortly after you log in, a folder should appear which says at the top “Swansea University Unified Desktop”. Down the side, select “Specialist Apps” then “College of Science”, and then “Computer Science”. It should look something like this:



3. **Open Java Folder.** Inside this folder you'll see one called 'Java' - open that and it should look something like:



4. **Versions.** There are four different versions of Java installed currently - and it basically does not matter which one you use. But you're probably better off with one of those with 'JDK 1.8' in its name. Double click one and you should see (possibly after some delay while it installs on the machine you are using) something like:



```
p:\>echo off
This CMD is running:
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)

For this session only - please do not close this window

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
p:\>
```

5. **Run Java.** You can check everything is ok by typing 'javac' (then hit return). If it says 'command not found' something is wrong. If it prints a long list of text, then it's fine. (Please note this works differently than from a version of Java you might install on your own machine - you *must* do it this way on the lab machines.)
6. **Open Notepad++** Open the editor *Notepad++* (which is in the Swansea University Unified Desktop -> Specialist Apps -> College of Science -> Computer Science folder - i.e. the one *above* the one with Java). It's icon looks like this:



You can find it the 4th row of icons in the first picture above - 3rd from the right.

7. **Create an Empty Java Program.** In Notepad++, save the (currently empty) editor file as *Hello.java* to your **P** drive - go to 'Computer' and then choose the option with your username in it. (The reason for saving an empty file is so that Notepad++ knows it's a Java file, and will start highlighting and colour coding your program when you start to enter it.) *It's important you get the name right - it should be Hello.java - not hello.java, or HELLO.java or any other variation.*

8. **Check** Go back to the Command Line and type ***dir*** (return). You should see your new (but still empty) file *Hello.java*.

Stage 2: Getting Hello World to Work

Type in the 'Hello World' program to the Hello.java file in Notepad++ and save it:

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

*You must type it **exactly** - case is significant.*

Now use the Java Compiler to *compile* your program - type this in the Java command window:

javac Hello.java

It's quite likely the first time you do this it will not work because you've made some kind of (minor) mistake. Look at the messages from the compiler that tell you what you have done wrong, and try to fix your program with Notepad++. There will be people in the room to help you - but have a go yourself first.

Once your program *compiles* you should run it:

java Hello

NOTE carefully the difference between the commands. To *compile* a program type ***javac*** followed by the name of the Java program (including the '.java' part). To *run* a program type ***java*** followed by the name of the program (without any '.' part).

Now type ***dir*** again. You should now see as well as *Hello.java* a new file *Hello.class*

The file *Hello.java* contains the *human-readable* program code, or *source code*. The computer cannot execute it in this form - hence the need to *compile* it to create the *executable* file *Hello.class*.

Stage 3: Compiler Errors

You probably made some mistakes when you wrote your program - if you didn't try to put some in deliberately to see what the compiler says in different cases - e.g. take out a ';', or misspell 'main'; or put in an extra '{' or '}'; or take one out. Look at the error messages the

compiler give you - the idea is to start to learn how the messages tell you what the problems are, where they are in your program, and how to fix them. For example, if you mistype the word 'println' in the example above as 'Println' - then the compiler says this:

Hello.java:3: error: cannot find symbol

```
                System.out.Println("Hello World!");
                ^
symbol:    method Println(String)
location: variable out of type PrintStream
1 error
```

I've put the most helpful parts of this message in red - the line number in the file (3); the actual error (cannot find symbol - it doesn't understand what some word means); and the ^ which marks where the error is (but be warned, it doesn't always get that bit right).

Stage 4: Making Changes to Hello World

Now we'll look at extending the program to do more things. First, think how you would get your program to print:

Hello World!

Hello Friends!

That is, two lines of text. Once you think you know what you need to do, try changing your program, and then compile and run it.

Stage 5: Calculating a Simple Expression

`System.out.println` can also output numbers. Try writing a program that contains the following:

```
System.out.println(7.5);
```

```
System.out.println(4+3.5);
```

```
System.out.println("4+3.5");
```

Make sure you understand why the last two are different.

Extend your program to print out the value of 17×1.75 - get Java to work out the answer; don't work it out yourself beforehand. The symbol for \times in Java is `*`.

Please make this a **new** program - don't just edit your "hello world" one. Always try to save programs separately, and keep them though out the module. **And give them sensible names** - remember Java program names should begin with a capital letter

Stage 6: Joining Strings

The + operator does not only add numbers, it also joins strings. Try this:

```
System.out.println("vanilla"+"icecream");
```

Can you explain why there is no space between the two words? Fix the problem.

An alternative way to do this is to use `System.out.print` instead of `System.out.println` (read the first one *carefully*). Try this:

```
System.out.print("vanilla");
```

```
System.out.println("icecream");
```

Now try joining more than two strings by using more than one + operator.

Stage 7: Joining Strings and Numbers

You can also use + to join strings and numbers - try:

```
System.out.println("Our Solar System has " + 8 + " planets.");
```

Assessed Stage 8: Combining What You Have Done So Far

You must get this task signed off by me or one one the Postgrads - when you are ready to do that, hold up a piece of paper.

You can convert distances in miles to kilometres (km) by multiplying by 1.609 (near enough). And the distance by bike between the two main campuses of Swansea University is 5.5 miles (according to Google). Write a program that prints out:

```
The distance between campuses is 8.8495km
```

Note there's a space before the 8 but not one after the 5. *Your program should do this using the techniques from stages 6 and 7; and **not** just print out the string directly.*