Department of Computing

COMP125 Fundamentals of Computer Science
Workshop - Introduction

Learning outcomes

Most of you have done COMP115 at Macquarie with Processing as the programming language and environment. We will revise some of the basics of programming learnt in COMP115. Also, in COMP125, Java is the programming language and Eclipse is the environment. Following are this week's learning outcomes,

a. 15 minutes: introductions

b. 10 minutes: be ready to use the lab computers (setup accounts and iLearn)

c. 50 minutes: revise COMP115/WCOM115 topics

d. 20 minutes: create a simple java program in Eclipse and under

e. 15 minutes: import a java project in Eclipse

1. **Access your account**

   To log on to the lab machines, first make sure the machine is powered on. Enter your username (OneID) and password in the appropriate entry boxes. Please note that if you have already accessed the labs, your username and password are the same as in the previous semester, and if are accessing the labs for the first time, you should have the required information from enrolment/ orientation session. If you do not have this information, please ask the tutor to assist you.

   Once you login to the computer, make sure you have access to COMP125 homepage in iLearn. If not, please ask the tutor to assist you.

COMP115/ WCOM115 Revision

2. What is the value of `result` when the following code is executed?

```
int result = 12/5;
```

3. What is the value of `result` when the following code is executed?

```
float result = 12/5;
```

4. What is the value of `result` when the following code is executed?

```
float result = 12.0/5;
```

5. Consider the following code:

```
size(300, 200);
background(255);
int x = (int)random(width);
int y = (int)random(height);
if(x < width/2) {
        if(y < height/2) {
                line(0, 0, x, y);
        }
        else {
                ellipse(x, y, 20, 20);
        }
}
else {
        if(y < height/2) {
                rect(x, y, 20, 20);
        }
        else {
                triangle(x, 0, x, y, 0, y);
        }
}
```

   Give 4 samples of `(x, y)` pairs, such that for each pair a different shape is drawn. Also state which shape is drawn for which `(x, y)` pair.

6. What is the value of `result` when the following code is executed?

```
int result = 1;
for(int i=0; i<4; i++) {
        result*=2;
}
```

7. What is the value of `result` when the following code is executed? Can you state the value of `result` in terms of `n`?

```
int n = 64;
int result = 0;
for(int i=1; i*i<=n; i++) {
        result++;
}
```

8. What is the value of `result` when the following code is executed? Draw a memory diagram to illustrate the memory transactions that occur during the function call. Refer to the document `functions.pdf` released in the revision package to help you with the diagram.

```
int foo(int a, int b) {
        if(a < b)
                return a;
        else
                return b;
}
```

```
int p = 6;
int q = 3;
int result = foo(p - 2, q + 2);
```

9. Write a function that when passed an integer, `true` if it is even (divisible by 2), and `false` otherwise.

10. Write a function that when passed an integer and a boolean, returns:
    - if the boolean value is `true`, then the square of the integer
    - otherwise
        - if the integer is positive, then the integer itself
        - otherwise, negative of the integer

11. Draw the memory diagram that illustrates the arrays stored in the memory for the following code:

```
int[] a = {10, 70, 20, 90, 30};
int[] b = {a[2], a[1], a[0]};
int[] c = b;
```

12. a. Declare an array `arr` to store integers.
    b. Instantiate it to hold 400 integers.
    c. Using a loop, initialize the items such that the first item is 10, second item is 15, third item is 20 and so on ...

13. What is the state of the array `a` when the following code is executed?

```
float[] a = {2.5, 2.2, -3.4, 0, 6.8};
for(int i=1; i < a.length; i++) {
        if(a[i] > 0) {
                a[i] = a[i] - 3;
        }
        if(a[i] < 0) {
                a[i] = a[i] - 1;
        }
        else {
                a[i] = a[i] + 1;
        }
}
```

14. Writing, compiling and executing a simple Java program

    The structure of a basic java program is as follows:

```
//filename: HelloWorld.java
public class HelloWorld {
        public static void main(String[] args) {
                //FIXME
        }
}
```

    The meaning of each of the components will be covered in the lecture. Write now, you can focus on replacing the comment FIXME with your code.

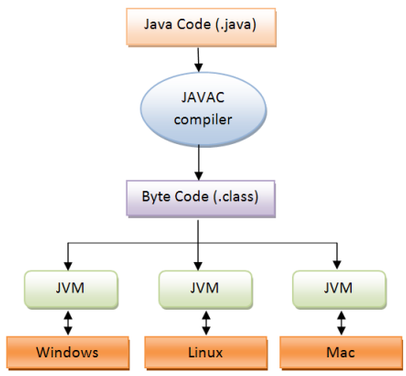
    a. Copy the code above
    b. Open text editor "Atom" installed in lab computers
    c. Paste the copied code
    d. Fix the indentation
    e. Save it as "HelloWorld.java" under a convenient directory (Say "Documents"). If the class name is `HelloWorld`, it **must** be in a file named `HelloWorld.java` (and **not** `helloWorld.java` or `helloworld.java` or `Helloworld.java`).
    f. Replace the comment FIXME with the statement,

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

    g. Save
    h. Exit.

    Next, to *compile* and *execute* the program.

    The following diagram describes the process of compiling a java source code (.java) into java bytecode (.class) and then executing the bytecode on the Java Virtual Machine (JVM).



    Open Command Prompt/ PowerShell and go to the folder in which you saved the java file. Run the following command (or `ls` on a Mac/Linux) to list the contents of the directory:

```
dir
```

    You should see the file (amongst others)

```
HelloWorld.java
```

Type the following command to compile `HelloWorld.java`.

```
javac HelloWorld.java
```

If everything is ok, you won't see any error message and if you run the `dir` command again, you will see the additional file

```
HelloWorld.class
```

If you *do* see an error message, take a look at the error message and fix the corresponding line. Ask your peers or tutor to help you if required.

Once the errors (if any) are fixed, type the following command to run/execute `HelloWorld.class`.

```
java HelloWorld
```

IMPORTANT: NO ".class" in the execution command.

Now, repeat the process with `Program1.java`. After compiling and running, you should see the following output:

```
number: 147
number of digits: 3
```

Now, repeat the process with `Program2.java`. You will see that the compilation doesn't succeed and that there are errors.

Open the file `Program2.java` in either Atom text editor (preferred) or WordPad (not Notepad).

Figure out the problems in the program, fix and compile until you succeed.

Once the compilation is finished, run the program. You will see the following output (which is incorrect):

```
n = 6
sum of integers from 1 to 6: 15
```

Go back to the program, fix the logical error, and run the program again.

You will notice that the output is unchanged (WHAT!?!?!?!?)

Every time we update the source code (.java), we must recompile the program before executing it.

So, compile the updated program again, and execute it. Now you should get the correct output:

```
n = 6
sum of integers from 1 to 6: 21
```

Repeat the process with `Program3.java`.

15. **Writing your first java program in Eclipse (pair-programming)**

    a. Follow the instructions in video uploaded here to create your first program in Eclipse that displays the following message in the console:

    `Bow before <person 1 name> and <person 2 name>`

So if your names are Bender Rodriguez and Philip J. Fry (since this exercise is being done in pairs, and if you didn't find a partner, choose your favourite character's name), the message should be

```
Bow before Bender Rodriguez and Philip J. Fry
```

b. Make sure there are no red-crosses on any line of your code (red crosses imply syntactical error and are demons hiding in your computer).

c. Compile and Run your program

16. **Importing Java project from archive file**

Follow the following instructions to import Java project contained in `workshop01demoProgram.zip` archive file.

a. Download `workshop01code.zip`. **DO NOT UNZIP IT** .

b. Open `Eclipse` IDE. A shortcut for it should be located on your desktop.

c. If prompted, set your `workspace` (a location where all your projects will be saved). We suggest you use a dedicated folder in your network drive as the workspace.

d. Click on File –> Import –> General –> Existing Projects into workspace (and NOT "Archive file") .

e. On the next window, choose "Select archive file" option and browse for the archive file `workshop01code.zip` and choose `Open`.

f. It should show projects `workshop01exercise` and `arrayHelp`. Click on `Finish`.

g. You should see a project in Eclipse in the left panel (`Package Explorer`).

h. Double click on the `workshop01exercise` to reveal `src`. `"src"` is short for `"source"` (that is, source code).

i. Double click on the `src` to reveal package `"(default package)"`.

j. Double click on the `(default package)` to reveal source file `TaskToDo`.

k. Your job is to complete the following methods (description in java file) in this class (you only need to write code inside the methods). Note the method headers are written in form `methodName(<parameterTypes>): <returnType>`.

   (a) `sum(int[]): int`

   (b) `sumEven(int[]): int`

   (c) `count(int[], int): int`

   (d) `contains(int[], int): boolean`

   (e) (Challenging) `countUnique(int[]): int`

   We have supplied a sample array in the `main` method and called the above methods in it by passing this array. Expected values are provided as comments next to these invocations. This is just a **sample data** and we will test your program by passing other arrays to the methods.

   A separate java project (`week1arrayHelp`) is provided with the workshop package and can be imported in the same way as the other archive files.

Advanced questions (D/HD)

1. Add a piece of code in the `main` method of class `TaskToDo` that performs the following tasks,

   a. creates an array to hold age of 20 people.

   b. assigns each value in the array to a random value between 1 and 100 (including 1 and 100). For this, a random number generator `rand` has been created for you and you can generate a random value between 1 and 100 using `rand.nextInt(100) + 1`. Display all values separated by a space on the console using `System.out.print(ages[i]+" ")` assuming `ages` is the name of the array.

   c. displays the average age

   d. computes the number of people under 50 years of age. Display this on the console using `System.out.println(var+" people under 50 years of age")` assuming you are storing the result in variable `var`.

   e. displays if there are two consecutive people aged over 80 years of age.

   f. **(Advanced)** displays all unique ages. That is values that occur exactly once in the array.

```
Sample output 1:

72 9 43 65 45 34 33 27 13 81 45 31 45 88 38 31 66 38 1 21
Average age: 41.3
15 people under 50 years of age
Two consecutive people over 80: false
Items occurring exactly once:
72 9 43 65 34 33 27 13 81 88 66 1 21

----------------------------

Sample output 2:

75 49 59 50 49 50 4 70 90 82 93 10 92 23 28 33 32 68 66 31
Average age: 52.7
9 people under 50 years of age
Two consecutive people over 80: true
Items occurring exactly once:
75 59 4 70 90 82 93 10 92 23 28 33 32 68 66 31
```

2. Write a method that when passed two integer arrays, returns `true` if they are identical (exact same items in the same order), `false` otherwise.

3. Write a method that when passed an integer array, returns an array containing the positive values in the array passed.

4. Write a method that when passed an array, returns the sub-array starting at the first index beyond which the array is in ascending order. For example, if the array passed is `{40, 10, 80, 20, 70, 70, 90}`, the method should return the array `{20, 70, 70, 90}`.