MACQUARIE University

Department of Computing

COMP125 Fundamentals of Computer Science
Workshop Week 1

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. As a first step, we'll see how can we *import* existing Java *projects* (same as *sketches* in Processing) and how Processing programs can be translated to Java programs. Following are this week's learning outcomes,

a. be ready to use the lab computers (setup accounts and iLearn)

b. revise COMP115 topics.

c. perform import of java projects.

d. perform *porting* of Processing sketches to Java projects.

e. write very simple Java project similar to a simple Processing sketch.

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.

Tutorial exercises

2. **Revision**

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

```
int result = 6 + 3 * 2;
```

Solution: 12

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

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

Solution: 2 (integer/integer = integer)

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

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

Solution: 2.0 (integer/integer = integer, and then that integer (2) is copied into a `float`)

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

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

Solution: 2.4 (float/integer = float (2.4), and then that float (2.4) is copied into a `float`)

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

```
int a = 7;
int result = 3;
if(a % 2 == 0) {
        result--;
```

```
}
else {
        result++;
}
```

Solution: 4 (expression driving the conditional code is `false` and so `result` increases by 1)

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

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

Solution: 16 (loop executes four times, for `i = 0, 1, 2, 3` and each time `result` doubles).

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

```
float foo(int a) {
        if(a > 20) {
                return a/2;
        }
        else {
                return a/4;
        }
}
```

```
float result = foo(16);
```

Solution: 4.0

h. Write a function that when passed two integers, returns the higher of the two.

Solution:

```
int higher(int a, int b) {
        if(a > b) {
                return a;
        }
        else {
                return b;
        }
}
```

i. Write a function that when passed two integers, returns the average of the two.

Solution:

```
float higher(int a, int b) {
        return (a+b)/2.0;
        // NOTE: 2.0 is important
        // if you write 2, integer/integer = integer --> WRONG
}
```

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

Solution:

```
boolean isEven(int n) {
        if(n % 2 == 0) {
                return true;
        }
        else {
                return false;
        }
}
```

k. Create two variables `x = 5` and `y = 8`, call the function `average` by passing `x` and `y`, and store the value returned in a variable `z`.

Solution:

```
int x = 5;
int y = 8;
float z = average(x, y);
```

l. Declare an array `arr` to store integers. Instantiate it to hold 400 integers. Initialize the items such that the first item is 1, second item is 2, third item is 3 and so on ...

Solution:

```
int[] arr; //declaration

arr = new int[400]; //instantiation

for(int i=0; i < arr.length; i++) { //populating the array
        arr[i] = i + 1;
}
```

m. **(advanced)** Declare an array `arr` to store characters. Instantiate it to hold all lowercase

letters of the English alphabet. Using a loop, initialize the items such that the first item is 'a', second item is 'b', and so on, till the twenty sixth item is 'z'.

Solution:

```
char[] arr; //declaration

arr = new char[26]; //instantiation

for(int i=0; i < arr.length; i++) { //lowercases
        arr[i] = (char)('a' + i);
}
```

n. 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;
        }
}
```

Solution: `{2.5, 0.2, -2.4, 1, 2.8}`
Note that first item is not changed since the loop started from `i = 1`.

3. Go to http://codingbat.com/java, create an account and solve as many warm-up exercises as you can in 10 minutes.

4. Structure of a (Runnable) Java program

   A java program that *executes* is wrapped in a class.

   ```
   public class HelloWorld
   ```

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`).

Inside the class, is the `main` method. This is the method that java looks for when executing.

```
public static void main(String[] args)
```

The program will compile correctly but not execute if you don't use the header above. So, in following cases, java will complain of `main` not being present.

```
public void main(String[] args)
public void main()
public static void main()
public static int main(String[] args)
static void main()
etc...
```

Inside the `main` method, you can write your code.

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

Putting it all together

```
//file name: HelloWorld.java
public class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello␣World");
        }
}
```

For next question onwards: You will work in pairs (it can be a different pair each week) with one person at a time in charge of the keyboard. The other person acts as a *critical observer*, watching carefully, suggesting things to try, making sure you've understood the problem correctly. You will swap roles regularly so that everyone has a chance to try each role.

5. Compiling and executing java programs from command prompt.

   Download the following files from iLearn,

   a. Program1.java

   b. Program2.java

   c. Program3.java

   Open Command Prompt/ PowerShell and run the following command to compile `Program1.java`.

   TIP:

   - Copy the following command
   - At command prompt, right click
   - (If not pasted already) choose "Paste"

```
javac Program1.java
```

   Run the following command to list the contents of the directory:

```
dir
```

   You will see a new file `Program1.class`. This is the assembly-level program called *java bytecode*. This is the code that is executed by Java Virtual Machine (JVM).

   Run the following command to execute `Program1.class`:

```
java Program1
```

   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`.

6. **Writing your first java program in Eclipse**

   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

7. **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 two projects - `workshop01demo, workshop01exercise`. Click on `Finish`.

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

   h. Double click on the `workshop01demo` 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 three java files -

      (a) `NumberClient`,

      (b) `PatternClient`, and,

      (c) `Brush`

   k. For each file, double-click the file and run it using the green play button in Eclipse. IMPORTANT! When you run `Brush.java`, choose **Run as Java Applet**. **Drag the mouse to draw.**

8. **(time-permitting) Explore Processing-Java relationship**

   In the workshop package, you'll find Processing source code for the brush program under the `Brush` folder as `Brush.pde`. Our java program was actually written in Processing and exported as a java program. This exercise is to illustrate that there is a close relationship between Processing and Java. **In pairs**, compare the files `Brush.pde` (on one person's computer) and `Brush.java` (on the other person's computer).

Take home exercises

9. Complete a Java program corresponding to a provided Processing program

   The second project from the archive file you imported earlier is `workshop01exercise`. Open the file `ClickCounter.java`. The code in `mousePressed` has been removed. Run the program (as an Applet) and you'll see if does nothing. Insert missing code by taking a look at the corresponding processing file `ClickCounter.pde`. Run it again, and it should behave the same way as the processing equivalent.

   Solution: Refer to `workshop01solutionCode.zip`

10. The purpose of this task is to re-acquaint you with arrays (that you studied in COMP115 or equivalent units). It's a series of methods that perform tasks of various levels of complexity. Working with arrays is a **CORE** skill required for problem solving and therefore COMP125.

    You are provided an incomplete `TaskToDo` in the project `workshop01exercise`. Complete the following methods in this class (you only need to write code inside the methods).

    a. `public static int sum(int[] a)`: returns the sum of all items in the array passed to the method.

    b. `public static int sumEven(int[] a)`: returns the sum of all **even** items (values that are divisible by 2) in the array passed to the method.

    c. `public static boolean contains(int[] a, int target)`: returns,

       - `true`, if array `a` contains item `target`.
       - `false`, if array `a` does not contain item `target`.

    d. (Challenging) `public static int countUnique(int[] a)`: returns the number of unique items in the array. That is, the count of items that occur exactly once in the array.

    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.

    Solution: Refer to `workshop01solutionCode.zip`

11. 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
```

Solution: Refer to `workshop01solutionCode.zip`