Faculty of Science and Engineering

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 enivronment. 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?

```
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)

e. 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).

f. 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

g. Write a function (henceforth called a *method* in Java) 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;
        }
}
```

h. Write a method 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
}
```

i. Declare an array `arr` to store integers. Instantiate it to hold 400 integers. Initialize the items such that the first item is 0, second item is 1, third item is 2 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;
}
```

j. Declare an array `arr` to store characters. Instantiate it to hold all letters of the English alphabet - upper, and lower cases. Initialize the items such that the first item is 'a', second item is 'b', and so on, and then twenty seventh item is 'A', twenty eighth item is 'B', and so on.

Solution:

```
char[] arr; //declaration

arr = new char[52]; //instantiation

for(int i=0; i < 26; i++) { //lowercases
        arr[i] = (char)('a' + i);
}

for(int i=26; i < arr.length; i++) { //uppercases
        arr[i] = (char)('A' + i - 26); //since 26 was the initial offset
}
```

For next question onwards: This semester in COMP125 we will practice *pair programming*. 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.

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

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

a. Unzip `tut01code.zip` provided on iLearn under `Workshop Week 1 files`. It contains a folder `templateCode` containing all archive files required from this task onwards.

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.

e. Select "Select archive file" option and browse for the archive file `processingToJavaCompleted.zip`, that is inside the `templateCode` folder and choose `Open`.

f. It should show a project `processingToJavaProjectCompleted` in the list of projects. Click on `Finish`.

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

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

i. Double click on the `src` to reveal `comp125`. `"comp125"` is the package name which we will, and you should, mostly use.

j. Double click on the `comp125` to reveal three java files (`ProcessingToJava1, ProcessingToJava3, ProcessingToJava5`).

k. For each file, double-click the file and run it using the green play button in Eclipse. When prompted, choose *Run as Java Applet*. **While the program is running, click and move the mouse if the program is not doing anything.**

4. **Explore Processing-Java relationship**

In the workshop package, you'll find Processing source codes for the three Java programs you ran in the previous question. Examine processing codes and their corresponding java codes to find how Processing in fact hides some details from the programmer. Notice some subtle differences like when the data type `float` is used, Java requires you to put an 'f' after the numerical value, that Processing does for you.

Solution:

a. *Some header information* gets added at the top of the Processing code when translated to Java code.

```
1  package comp125;
2
3  import processing.core.*;
4  import processing.data.*;
5  import processing.event.*;
6  import processing.opengl.*;
7
8  import java.util.HashMap;
9  import java.util.ArrayList;
10 import java.io.File;
11 import java.io.BufferedReader;
12 import java.io.PrintWriter;
13 import java.io.InputStream;
14 import java.io.OutputStream;
15 import java.io.IOException;
```

b. We have added the following statement ourselves to ignore warnings. Warnings are not the same as errors. The program still executes successfully with warnings. We shall discuss more on this at an appropriate time during the semester.

```
SuppressWarnings({ "unused", "serial" })
```

c. The entire Processing code is placed in a `class`, the name of which is the same as the name of your Processing program.

```
public class ProcessingToJava1 extends PApplet {
```

d. The keyword `public` is added in front of `void setup()` and `void draw()`.

e. In absence of `void draw()`, a `noLoop();` statement is added at the end of the `setup()` method

f. A `main` method is added in the class. The only difference in the `main` methods of all Java files is the String value in the double quotes.

```
static public void main(String[] passedArgs) {
    String[] appletArgs = new String[] { "ProcessingToJava1" };
    if (passedArgs != null) {
      PApplet.main(concat(appletArgs, passedArgs));
    } else {
      PApplet.main(appletArgs);
    }
}
```

5. **Write a simple Java program corresponding to a provided Processing program**

Import the Java project from `processingToJavaToBeCompleted.zip`. Complete `ProcessingToJava2.java` so as to get the same output as in the Processing sketch `ProcessingToJava2`.

Repeat the same process for `ProcessingToJava4`.

Solution: Please refer to project in archive file `processingToJavaToBeCompletedSolution.zip`.

Programming exercise

6. 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 `AssessedTask.java` in the project `processingToJavaToBeCompleted`. 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. (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`) to help you with array basics has been uploaded on iLearn.

Supplementary task (for self-study)

7. Add a piece of code in the `main` method of class `AssessedTask.java` 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:

50 42 98 67 56 6 62 61 74 74 8 45 37 74 74 89 14 95 9 99
Average age: 56.7
7 people under 50 years of age
Two consecutive people over 80: false
Items occuring exactly once:
50 42 98 67 56 6 62 61 8 45 37 74 89 14 95 9 99

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

Sample output 2:

57 13 75 83 12 4 54 65 47 84 48 37 70 26 16 99 29 68 1 78
Average age: 48.3
10 people under 50 years of age
Two consecutive people over 80: false
Items occuring exactly once:
57 13 75 83 12 4 54 65 47 84 48 37 70 26 16 99 29 68 1 78
```