Macquarie University
COMP255
Assignment 1

September 13, 2016

Organisation

Marks Assignment 1 is 20% of overall unit marks.

Deadlines for submission

Sprint 1	Wednesday 21 September, **11:59am**
Sprint 2	Thursday 29 September, **11:59am**
Sprint 3	Sunday 9 October, **11:59pm**

No late submission.

Objectives The main objectives of this assignment are:

- to practise software project management (Agile);
- to gain experience in using version control systems (git);
- to gain experience in reading code documentation;
- to gain experience in writing code documentation;
- to gain experience in designing unit and integration tests.

Note This assignment specification aims to provide as complete a description of this assessment task as possible. However, as with any specification, there will always be things we should have said that we have left out and areas in which we could have done a better job of explanation. As a result, you are strongly encouraged to ask any questions of clarification you might have, either by raising them during a lecture or by posting them on the iLearn discussion forum devoted to this assignment.

The Specification

You have to develop a small library, in Java. The library should provide an interface to interact with other programs, typically shell scripts, not necessarily written in Java. More precisely, the objective is to be able to *programmatically interact* with a program that can read its input from the *standard input* and produce its results, as a string, on the *standard output*.

For this assignment, you may have to look up and learn by yourself, some concepts (e.g. *standard input, output*). The code to be written is fairly short and does not require intricate algorithms. The emphasis in this task is on:

- understanding the problem to be solved and write a formal specification;
- understanding what Java libraries offer and how to use them;
- documenting your code so that other developers can use it;
- writing tests for your code;
- developing the software as a Agile project;

An example of usage is as follows. In many operating systems, there is a command or script to list all the user's processes (e.g. `ps aux` on Unix) in the terminal. Assume you need to collect the set of running processes but programmatically from a Java program. In the Java program, you want to do the following:

```
1: // create a process manager to interact with `ps`
2: ProcessManager p = new ProcessManager("ps", "aux");
3: // spawn a process that runs `ps aux` and collect the result
4: String res = p.spanwAndCollect();
```

What and How to Submit?

For each subtask (sprint) you have to submit a *commit* via BBmarking. Your implementation will be based on a fork of the following project: `https://bitbucket.org/franck44/comp255-2016-assignment-1`.

The interfaces of the methods you have to implement are provided in the class `ProcessManager` in the file `src/main/java/ProcessManager.java`.

There are some example tests using Junit in `src/test/java/`. The two files `README.md` and `DESIGN.md` in the project root directory provide directions how to set up the environment variables, compile and run the tests. You have to update these files as part of the documentation process.

You have to install the bitbucket add-on *awesome graphs*. *awesome graphs* generate useful graphs to monitor your progress e.g dates of commits, punchcard. You have to make those graphs part of your commit when you submit (you can dump a PDF or a JPG ot the graphs) in the project root directory.

1 First sprint (6 marks)

In a first instance, we assume that the program to be run (e.g. `ps aux`), runs and terminates, and we do not need to collect the result of the output.

Useful Concept for this sprint: ProcessBuilder (Java)

Write the code for the `spawn` method in the Java class `ProcessManager`. Can you test that the process is actually created, and properly destroyed (resources released)?

Implement the method `destroy()`. Can you write tests (using Junit) to test that a process is created and destroyed?

You have to document your code using the JavaDoc[1] standard.

2 Second sprint (8 marks)

Useful Concept for this sprint: Pipes, InputReader/Writer, BufferedReader/Writer

You have to write the code for the `spanwAndCollect` method. To do this, you need to *connect* the input/output streams of the external process to the Java program (`ProcessManager`).

You have to write the documentation and provide some tests (JUnit). Notice that you may use other external processes to test your implementation (e.g. write your own scripts).

An example usage is as follows:

```
1: // create a process manager to interact with 'ps'
2: ProcessManager p = new ProcessManager("ps", "aux");
3:
4: // spawn a process and collect the result
5: String res = p.spanwAndCollect();
6:
7: // res should contain the list of processes
```

3 Third sprint (6 marks)

Add a timeout to the `spawnAndCollect` method. If the process does not answer before the timeout, an exception should be raised.

An example usage is as follows:

```
1: // create a process manager to interact with 'sleepAndEcho'
2: // where 'sleepAndEcho echoes "DONE" after 3 seconds
3: ProcessManager p = new ProcessManager("sleepAndEcho", "3");
4:
5: // spawn a process and collect the result.  Timeout 2 seconds
6: String res = p.spanwAndCollectWithTimeout(2000);
7:
8: // should raise a timeout exception
```

[1] `http://www.oracle.com/technetwork/articles/java/index-137868.html`