Data Structures (CS 102), Professor Yap – Fall 2016

HOMEWORK

September 16, 2016

Homework 2: Due on Friday Sep 23, 2016

- This homework provides basic familiarity with tools of software development (edit-compile-test cycle) which are usually hidden in a complex IDE like Eclipse. You will need to install a Terminal program and "make" program. We provide some pointers in our wiki page on Programming, or google it or see youtube videos).
- This homework is worth 100+5 Points. In each homework, you get the extra 5 points by following all our instructions.
- READ CAREFULLY the homework instructions and policy in `http://cs.nyu.edu/~yap/wiki/class/index.php/DataStruc/Hw-page`. In particular, your zip file is named `hw2_Yap.zip` and it contains only one folder named `hw2_Yap` which has these four files: `README`, `Makefile`, `hw2_Yap.pdf`, `src` (a folder). The `Makefile` is provided by us.

PART A: WRITTEN ASSIGNMENT

Question A.1 (6 Points) Waterfall Program Development
This question is somewhat open ended. According to the text [p. 3], these[1] are the phases in the life cycle of a software:

(1)	Problem Analysis:	*goals and issues*
(2)	Requirement Specification	*what the software must do*
(3)	Architectural Design:	*structure of the program*
(4)	Implementation:	*coding in some language*
(5)	Testing and Verification:	*debugging*
(6)	Delivery:	*handing over to user*
(7)	Operation:	*actual use*
(8)	Maintenance:	*changes or additional functionalities*

Please instantiate each of these phases with YOUR experience in solving of the `ATM` program in hw1, Write one complete sentence for each phase. (OK, a long sentence if you must.)

NOTE: you probably use the "Spiral Model" in reality, but one could always (after the fact) re-cast the experience into the Waterfall model:)

Question A.2 (6 Points) Signatures

(a) The header for the `main` method of Java is fixed except for one token. Which token is this? Are there reasons to exploit this flexibility?

(b) Consider the following program:

[1] I skipped one.

```
class Hello {
    public static void main(String[] args){ }
    private int main(String[] myArray){ }
}
```

Will it compile? Explain (but first answer YES or NO.)

(c) What is the signature of this method?

```
protected int[ ] mystery( double[ ] dd, Account aa, int ii){...}
```

Question A.3 (24 Points) Terminals
Terminal programs are automatically available in Linux-type systems including MacOS. But in Windows, do not use the standard Command Window. Starting August 1, 2016, Windows 10 has officially (finally) supported `Ubuntu` (a Linux Operating System); but this `Ubuntu` is still lacking. So I recommend that you install `Cygwin` on Windows (see my Programming wiki page for hints).

(a) Every terminal is actually running a `shell` program that sits in an interactive loop (like our ATM program), ready to read your commands. Although there are different shells, the following basic commands should be available on any shell you happen to use:

`ls, cd, pwd, mkdir, rm, alias, echo, uname`.

Submit one or two screen shots in which you use all of these commands on your terminal. (Try to fit as many commands into one screen shot as possible.) Be sure that your screen shots show some information that proves that it is really you! E.g., echo ”My Name is Chee Yap”.

(b) Explain the significance of the $PATH variable for the shell program. Tell us how you can exploit it to ensure that you can access the following programs from the terminal:

`java, javac, eclipse, make`

Please provide a screen shot to show (1) the value of your $PATH variable, and (2) that you could access all the above programs from the terminal. The way to show that you have access to (say, javac) is to type the command “which javac”.

(c) You can personalize your shell by putting a file (usually called **.profile**, **.bashrc**, etc) in your home directory (˜). Some installation already provide some default file **.profile** or **.bashrc** for you. Our Programming Wiki Page (item Z) has some files you can download for this purpose. If necessary ask the tutors to help you. Give a screen shot to prove that you do have a personalization file (E.g., let the personalization file ”echo” some information unique to you).

Question A.4 (14 Points) Makefile
Modify the Makefile provided for hw2 (not hw1) so that it be used to achieve these actions:

```
make – this compiles all the java programs into bin
make run – this runs Cash with default argument 1
make run 0 – this runs Cash with argument 0
make help – print the variables PATH and HOME
make run ifile=src/Input.txt
```

where the last command runs Cash but taking its input from a file (say, src/Input.txt which we provided). For most students, the first 3 actions may just work without changing our Makefile! You must write the ”help” target.

NOTE: Your Java programs for hw2 can either be directly placed under `src` (this is called the **Default package** in Eclipse) or under `src/hw2` (i.e., in the **hw2 package**).

PART B: PROGRAMMING ASSIGNMENT (50 Points)

All needed files are found in `hw2_Yap.zip` from Piazza/Resources. They must be included in your `src` folder but unmodified.

Question B.1 (20 Points) The Cash ATM

In our lecture on Thursday, we saw that the ATM program in hw1 is really a shell program. In view of this, our next version of ATM will be called **Cash**, not **bash**. Do look at our hw1 solution `ATM.java` for ideas. You must use these provided files:

`Account.java, PlusAccount.java, SavingAccount.java, Client.java, CashAbstract.java, Input.txt`.

Your `Cash` class must extend the `CashAbstract` class. Note that `Account.java` is modified from hw1. You must not modify any of our provided files. A client can have either or even both kinds of accounts.

The detailed description of this project is found in the above files, but first read the Overview in `CashAbstract.java`.