COMP 3021 Programming Assignment 1

A Simple Point-of-Sale (POS) System

Due Date: 17 March 2012 11:59pm

1. OBJECTIVE

The objective of assignment 1 is to get you familiar with Java basics, which include the syntax, Java input/output, classes, objects, inheritance and interfaces. After performing this assignment, you should be able to master these basic Java concepts. You need to use Eclipse for this assignment.

2. TASK

In this assignment, the overall task is to develop an application that records sales and handles payments in a retail store.

Your POS application can be executed in two modes: interactive and batch. In the interactive mode, the POS accepts textual commands submitted by cashiers at their consoles to handle payments in a retail store. In the batch mode, the POS accepts textual commands from a file. If a program variable (the batch file is specified) is appended (i.e., run as "`java POS input.txt`"), POS runs in the batch mode. Otherwise POS runs in the interactive mode (i.e. run as "`java POS`"). Each time POS reads a line from the console or from the batch file. Upon correct inputs, there should be no differences in application behavior if the inputs from the console and the batch file are the same. Therefore, in the following, we mainly illustrate our requirements using the interactive mode.

```
Welcome to the Point-Of-Sale Registration System
Please enter your user name: chunlin
Please enter your password: chunlin
<LOG> Electronic-Sales Counter is started successfully by user chunlin!
```

In the interactive mode, a cashier will be asked to log in the system using his/her user id and password. (There are no limits on the maximum trials of incorrect passwords or users). The system displays a successful message if the cashier inputs the right user id and password. If the user id or password is wrong, the cashier will be asked to re-enter as follows. In the batch mode, the POS exits with an error log upon incorrect user id or password.

```
Welcome to the Point-Of-Sale Registration System
Please enter your user name: chunli
<LOG> Non-existent user chunli, please enter again!
Please enter your user name:
```

```
Please enter your user name: chunlin
Please enter your password: chun
<LOG> Wrong password chun for user chunlin, please enter again!
Please enter your password:
```

After the successful login, the system will act as an electronic-counter, asking the cashier to enter a sales record or to press ‘2’ to exit as follows.

```
---------------------------------------------------------
Welcome to the Electronic-Sales Counter!
Please enter '1' to record sales or '2' to exit:
```

If a number other than ‘1’ and ‘2’ is entered, the system responds with an error log message. It then prompts the cashier for the next command. If the cashier selects to enter ‘2’, then the Electronic-counter will be quitted. In the batch mode, the POS exits with an error log upon an invalid command.

```
Please enter '1' to record sales or '2' to exit: 3
<LOG> Invalid command
Please enter '1' to continue or '2' to exit:
```

```
Please enter '1' to continue or '2' to exit: 2
<LOG> User chunlin has successfully logged off!
```

A sales record consists of several purchasing products. After ‘1’ is selected, the cashier is required to enter a purchasing product one by one, each consisting of the product ID and the number of purchased products (the system will assign a SalesID (formatted as SalesIDXX, initially SalesID00, then SalesID01, etc.) each time after ‘1’ is selected). After the cashier inputs the product ID, the product name will be displayed, and the system asks for the number of purchased products as shown in the following. Remember that if the product ID is wrong, or the purchased number of items is not a digit, the cashier is required to re-enter in the interactive mode, or the system exits with an error log (`Incorrect product ID!` or `Incorrect Number!`) ***in the batch mode***. If the cashier enters ‘c’, it means that all products in the current sale transaction have been entered, the total price will be displayed (retain 1 digit of the decimal part).

```
Please enter a list of purchasing-product ID and number
--------------------------------------------------------
Please enter product ID or press 'c' to end the list: ID001
Product name is pencil, purchased number: 1
Please enter product ID or press 'c' to end the list: ID002
Product name is basketball, purchased number: 1
Please enter product ID or press 'c' to end the list: c
--------------------------------------------------------
Purchasing-product list:
pencil * 1  = $1.0
basketball * 1    = $120.0

The total price is $121.0
Please pay with cash, received cash (or cancel by entering '0')$:
```

During the payment process, the entered cash amount should be larger or equal to the total price, otherwise the cashier will be required to re-enter the cash amount, or the system exits with an error log (`Cash not enough!`) ***in the batch mode***. The cashier can also select to enter ‘0’ to cancel this sale transaction. After the successful payment or sale abandon, the system returns back to the electronic-counter.

```
--------------------------------------------------------
The total price is $: 60.0
Please pay with cash, received cash (or cancel by entering '0')$: 40
Sorry, cash not enough, please enter cash amount again: 70
Change $: 10.0
Please enter '1' to continue or '2' to exit:
```

```
--------------------------------------------------------
The total price is $: 1.0
Please pay with cash, received cash (or cancel by entering '0')$: 0
<LOG> SalesID01 cancelled!
Please enter '1' to continue or '2' to exit:
```

All the data of the valid users and products are kept separately in two files named “userPasswordFile.txt” and “productListFile.txt”. The “userPasswordFile.txt” contains lines of the user id and password pair (separated by a space, there are no limits on the user or password formats). The system will check the password according to the information in the file and reject the login with the wrong user id and password, detailed information can be found by running the demo.

```
chunlin chunlin
yangjun yangjun
zhulin zhulin
```

The “productListFile.txt” file contains the information about all the products in the store such as product id (formatted as IDXXX, where XXX are digits), product name, unit price, stock number. An example is shown below. Each line contains the information of one product, including product id, product name, unit price and stock (for simplicity, assume that the stock is always enough, so you do not need to handle the stock-not-enough exception) separated by spaces.

```
ID001 pencil 1.0 1000
ID002 basketball 120.0 1000
ID003 notebook 2.2 1000
ID004 tissue 10.0 1000
ID005 doll 33.0 1000
```

The last function of the system is to automatically record the sales information and log information (those lines in the previous figures marked as <LOG>). Every success sale and other log info will be recorded in the file "logAndSales.txt" in the way shown below. Each line contains a sales record or a piece of log info.

```
Electronic-Sales Counter is started successfully by user chunlin!
SalesID00       total $7.6; pencil 1; notebook 3
User chunlin has successfully logged off!
```

Please make sure your program can handle invalid inputs properly and no data loss (data in the files changed by success payments) if the program quit abnormally (Such as Ctrl+C).

Finally, please also include your test suite and test cases for unit testing when you submit your program. You are required to obtain 100% statement coverage to all public methods and the main routine of the POS program. Statement coverage of exception handling blocks is not required.
Your test classes and test methods should strictly follow their naming regulations as introduced in the lab session. That is:
Test class should be named as xxxTest such that xxx is the name of the tested class.
E. g: Class name: DateHelper,
Test class name: DateHelperTest.
Test method should be named as testXxx in which Xxx is the name of the tested method. You can use a sequential number as the identifier of different test cases for the same method.
E. g: Method name: MatchDate(),
Test method name: testMatchDate01().
It is strongly recommended that you organize all your test cases in a package separated from your program, so that our TAs may check the code of your test cases quickly.
When you submit your assignment, make sure that you have included a test suite which executes all of your tests.

3. GRADING

Grading will be based on functionalities. The breakdown of grades is as follows:

- The program runs and does not crash(Hint: even given invalid inputs): **20**
    - We will use some invalid inputs to check whether your program can handle them properly without crash.
- The correctness of the output result: **50**

    - The correctness of your program will be checked using five groups of inputs which cover all the functions of the program.
    - For each group of input files, if the output of your tool is the same as the correct output, you will be given 10 points for this group. The output includes not only the messages shown in the console but also the corresponding files.
    - Any deviation with the correct output will be assessed as “minor”, “medium”, and “severe”. For these three cases, you are given 8, 5, and 2 point respectively.
- Code quality (design and documentation): **30**
    - Good design (e.g. overall design, reusable classes, clear classes design... etc.): 15
    - Good documentation (Brief but meaningful comments for each class and function, appropriate use of javadoc tags): 10
    - Good coding style (e.g. appropriate indentation, meaningful variable and method names, explanatory comments for difficult parts ... etc.): 5

- Test Quality : **10**
    - Good naming regulations (Strictly follow the naming regulation introduced above. If you fail to follow test naming regulation, your score of test quality will be under 2): 3
    - Test suite availability (You need to include a test suite containing all tests you’ve written, and include a test runner implementation so that all tests are executed with a single execution of that runner): 2
    - Test adequacy (You are required to obtain 100% statement coverage for all public method and the main method of the whole program. Coverage of exception handling codes is not required. If you fail to achieve this requirement, your score will be based on the actual coverage level: 1 for coverage level 0%~50%, 3 for coverage level between 51%~99%. The coverage level will be evaluated by EclEmma report of your tests): 5

Note:

- If your program could not be compiled, we may give you partial credits (normally under 20 points).
- This is an individual assignment. Please acknowledge the resources (books, web sites and so on) that you have consulted and the people whom you have discussed with. Please be reminded that plagiarism is a serious offence. Students who are involved in plagiarism could fail the course assignments.

4. SUBMISSION

- The assignment is due at 11:59pm on the due date and you may use Course Assignment Submission System (CASS) to submit your assignment. To use CASS, you may first go to https://course.cs.ust.hk/cass/submit.html. Then choose Comp3021 for the course and select the files you would like to submit. Please submit the file named assignment1.zip.
- Please ensure your submitted program is self-completed, that is, can be compiled and run by itself. For fairness, we will not supply more files to compile and run

your program during grading. And also provide us a readme file to tell us how to run your program.

- Please submit your assignments by the due date. **Late submission will not be accepted.**
- You are recommended to use JDK 6.0 to compile and run your program.