Programming IT
Assessed Exercise 1

This exercise is worth 10% of the course grade.

Specifications

Lilybank wine merchants sell a variety of wines on a sale and return basis. You are to write a program that processes a series of sales / returns.

1. When your program starts it should take the name and balance of the customer. Positive balances mean that the customer owes money to the merchant, negative means the merchant owes money to the customer.
2. The program should then loop indefinitely. Within each loop, the user should be prompted for a wine name, quantity, and price.
3. The transaction should be processed and the customer balance updated.
4. If they buy wine (quantity is positive) the balance should increase. If they return, it should decrease.
5. When a wine is returned, the customer gets back 80% of the price they paid for it.
6. If the user leaves the wine name blank, the loop should terminate.
7. After each transaction, a summary of the transaction and the new balance should be printed.
8. A file called statement.txt should be produced with the following format:
    a. First line: Customer name and initial balance.
    b. Subsequent lines: Wine name, cost, quantity, transaction cost, new balance

We will give a demonstration of a working system in class.

Approach

- You should adopt an object oriented approach. We suggest (as a minimum) the following objects:
    - CustomerAccount
    - Wine
    - The rest of the code can go in a main method.
- Exception handling should be done properly for the file writer (i.e. exceptions caught individually) and the file should be closed at the end of the program.
- Although users should enter amounts in pounds and pence (e.g. 1.23 for £1.23) we suggest storing money in pence (as an int) and then converting this into pounds and pence for display. Exactly two decimal places should be used when displaying amounts (e.g. 1.2 should be displayed as £1.20).

What you should submit

- A set of .java files containing all classes used in your solution. The main method should be in a file called AssEx1.java.
- Code should be commented.
- A status report (details below)

How to submit

- Compress your .java files and report (ideally as a .pdf, but .docx fine too) into a single zip file, and submit this through Moodle.
    - The lab machines have 7-zip installed. To zip, select all of the files you would like to include, then right click and choose “Add to archive”.
- Late submissions will be penalised by 2 bands per day.
- You can submit as many times as you like. Only the last one will be assessed (feel free to do a test submission to make sure you know how submission works)
- Make sure you tick the “declaration of originality” box
- Deadline: 6th November 2018, 5:30pm

Status report

Should include:

- Your name and student number at the top
- Short summary of the final state of your program (should indicate how closely your program matches the specification)
- Any assumptions you have made that have affected your implementation
- Any known deficiencies and how you might correct them
- Details of the test data used and the results expected and obtained should be provided (with corresponding screen dumps to give evidence of functionality)
- At most one page, excluding screen dumps

Mark scheme

Element	**Weighting**	**Contents**
Functionality	40%	Proportion of the specified functionality implemented, and program correctness
Design	34%	Choice of methods and instance variables within classes, design of methods
Documentation and Testing	26%	Readability of code (incl. comments), quality of status report, selection of test data

Plagiarism

- Ok to discuss the exercise with friends but, submitted code must be your own
- What is collusion? Students A and B are said to be colluding on an exercise if:
    - A completes the exercise (or part of the exercise) and provides his/her code to B
    - B makes only trivial changes (see later) and submits the resulting code as his/her own
- Provider and recipient are penalised equally
- What is reasonable discussion?
    - students A and B discuss on paper at a higher level than Java code as to how to solve the problem
    - each implements the solution separately at their own machine
- See plagiarism policy and guidelines for more information – https://webapps.dcs.gla.ac.uk/LTC/policies/ PLAGIARISM_Policy_2011.pdf
- This exercise can be completed using your own classes and Java classes that have been covered in course. You do not need to use other third party classes and methods.
- Usage of code that is not your own:
    - proportionate penalty for acknowledged use of code that is not your own (within status report) e.g., snippets of code from internet
    - higher penalty for unattributed use of other code
    - blocks of code from lecture notes or demo programs do not need to be acknowledged