Programming Assessed Exercise 1

General feedback

There were 72 submissions. Overall the quality of work on this exercise was excellent. The average mark was A3, and the highest mark of A1 was obtained by 24 students. If you feel these marks sound high, please note that this is quite typical for this first exercise.

Some general comments which apply to several students are given below, under the headings of “functionality”, “design” and “documentation”. In some cases the shortcomings noted were not penalised in this first exercise, but they may be in subsequent exercises.

Functionality

The functionality was generally very well done, though common errors included not catching null pointer exceptions when either of the input dialog boxes (corresponding to the customer’s name or initial balance) were cancelled. Sometimes the transaction amount did not include the service charge in the case of a return transaction.

Most people managed to arrive at a JFrame whose components were well laid out and their positions did not jump around when the frame was resized. Where the components did suffer from this problem, people had tended to use FlowLayout which is not robust enough as a layout manager for *all* of the required components.

In some cases, the transaction amount and/or current balance was not always given to two decimal places. This occurred either throughout the program’s execution or for just part of it (typically when the initial balance was first shown in the JFrame).

Some students incorrectly displayed the transaction amount in the case of a return (i.e., the service charge had not been removed). There was also a problem in some cases with mixing up plus and minus when modifying the customer balance in the case of a sale or return. Also in some cases “CR” was not displayed correctly for credit balances.

Design

In some cases, actionPerformed was rather long. It could be split up by calling a helper method to acquire the details about the Wine object, and then once this has returned, actionPerformed can test whether a sale or return should be processed, calling the appropriate method in each case.

Some students had a great deal of repetition of code between the parts of actionPerformed that process a sale or return in LWMGUI. The Wine object was created in the parts that process a sale or return, but was not used to proper effect. By following the approach in the previous paragraph, the object would be used to better effect in each case.

In some cases the indentation of the code had gone off track in a few places, making it hard to read. This can be corrected automatically by Eclipse (under “Source”, choose “Correct indentation”).

Some people declared some of the instance variables of CustomerAccount and Wine to be public, which should not be done (refer to the lecture on "data encapsulation").

In a method, the name of an instance variable and the name of a method parameter should not be the same (even if "this" is used to disambiguate).

Some classes had extraneous instance variables. For example some people's CustomerAccount had a transaction amount instance variable, which is unnecessary. Similarly, the same information (e.g., the balance of a customer) should not appear as an instance variable of more than one object.

A few students seem to have missed the conventions we use for naming variables and methods. In particular, some variables were given the same name as the class. In addition, some methods are starting with uppercase letters as are some variables. It can be confusing.

Documentation

There was very little commenting of the code in some cases. More complicated methods such as those for processing a sale or return in CustomerAccount should ideally have comments that describe the underlying algorithm.

Some of the identifiers in some solutions are really not very meaningful – e.g., button1 and button2 in LWMGUI – it would be better to call these saleButton and returnButton, for example.

Though reports were in general pretty good, the main area these fell down was testing. Although only a limited selection of screenshots was actually requested, your own testing should have been much more extensive. Lack of this led to some people stating that their programs fully met the specification, when in fact they did not.

Some students provided screenshots in their status report without any commentary beside them to indicate what they are meant to be conveying.

A lot of people missed out assumptions, and a few missed the requirement to have just one line detailing whether the program met the specification or not (with further information on known deficiencies). The line where you detail whether your program meets the specification should ideally be placed right at the start of the report.