**Evaluation**

**Introduction**

On behalf of BMC Consulting, I was tasked with developing a computerized system for Gas Company, to update their existing software. I was to design, implement and test an object oriented solution for a Gas Company. This was to be a simple program with a non-graphical user interface.

**The Requirements & Specifications**

The specification was to create a Java class called Gas Account, with sub class Business Account which extends Gas Account. A Main menu was to be created in the program class to manipulate the methods. Within the menu program one business customer to be created by input and not with constructor.

**Main specifications** – create business account with – customer reference number, customer name, customer address, customer units and discount. The menu should begin with the request to fill in these values.

Program should;

* Create New Customer
* Make deposit to pay off account balance.
* Allow for entry of the units used by the customer to increase the balance with business discount.
* Display the full customer details.
* Amend the discount of the business customer.
* Amend the unit cost.
* Onscreen help – with explanation.
* Exit.

**Test Plans**

A unit test plan would be created for Gas Account, and an integration test would be created for the Business Account. The Menu programme would be fully tested as well to make sure the system was functioning according to the requirements.

**Review**

Having taken note of requirements and specifications for the contract, I set about the tasks of UML diagrams for the GasAccount and BusinessAccount objects declaring all the properties and behaviours (methods) for each of these classes, including the private and public elements. I drew up an inheritance diagram to show the relationship between both the Gas Account and Business Accounts. This assisted me in knowing how the program would work with the inheritance, reusing the methods from the super class in the sub class and redefining the use of method *recordUnits which added and extra calculation for taking off discount and then using overriding to use this in the Business Account .*

I next proceeded to write the pseudo code and flowcharts for object classes GasAccount, BusinessAccount and the GasMenu. This gave me a better feel for the program and how it would operate. *I could see at this point the advantage of not just creating a single program but breaking up the program into Java class objects as it can mean that less redundancy and reusing and redefining of the methods in the superclass.*

I created Data Dictionaries for each class, *this assisted me* in the creation of the menu program where I had a better sense of types of input.

During the implementation of the tasks of writing the code for GasAccount, BusinessAccount and the coding the GasMenu program class, *I thought there could be other additions made to the account for instance, ‘ Customer Forename, Customer Surname, Telephone No and Email Address’, this would have given the company more details on the customer and been of benefit for contacting the customer. This could have been viewed easily with option 5, ‘display full account details’.*

I created a test plan to test the GasAccount, BusinessAccount and GasMenu program. This tested the integration of the entire program by unit testing. I tested each of the constructors, methods and manipulated them with different data, both valid and invalid. *During the test 20 of the GasMenu program I found that my validation was not correct. When I entered a zero into the input it changed the balance to zero which failed the validation test. This was because the if statement in the method was set to >= 0.0 rather that >0.0, this was changed and after this it would not accept zero.*

The object classes (Business Account & Gas Account) and programme classes (GasMenu, TestGasMenu & TestBusinessAccount) were planned using flowcharts and pseudo coding which helped to set out what the main variables, methods and instances would be.

I commented all the coding, classes and methods with Javadoc for future repair and maintenance of the program and included this in my report with the ‘dist’ file. This will assist any future changes to the objects or program in the future. *During the running of the Javadoc report several errors were reported and due to my limited knowledge of these reports I found this quite taxing, most the errors were due to there being more information needed due to ‘ unknown tags and parameters’. I will need to get assistance to repair these outstanding errors but they do not affect the workings of the main program.*

I designed and created a user guide that covered the main elements of how to use the program step by step. I later had to update this as I added a new option (see improvements)

**Peer Review**

I had my software peer reviewed by Gerard Houston (fellow student). He reported several issues with my program that require changes (see Appendix A);

User Guide is too basic and requires more detail, initially I thought that simple written explanation would suffice for my user guide but I can see Gerard’s point that more screen shots are needed with valid and invalid inputs and outputs. It makes it easier for the first-time user. I have redone the user guide with these features.

Help Menu is too elongated, it needs to be shorter and more compacted together to give the user less scrolling. This I have changed taking out the spaces between each option explanation.

The outputs for the balance has too many decimal places, I initially didn’t notice this due to the values I had entered were most likely rounding the numbers down more neatly but when 9.99 was used for deposit it revealed this inadequacy. This was changed by adding coding, %1$.2f to the outputs, reducing the balances to two decimal places.

The outputs for errors were too basic, when option 2-7 are used without first creating a customer an error was raised that said, ‘ Error – Create New Customer First – Option 1’. This did not cover when invalid alphabetical data was inputted so this has been changed to, Error – Enter Numeric Value or Create New Customer First – Option 1’. This way I can deal with both issues at the same time, whether the user fails to create a new customer or the user enters alphabetical input instead of numeric.

**Improvements & Future Improvements**

This non-graphical interface is very simple to use but it could be more attractive to the eye and even easier to use with a *graphical user interface*. Each option could be highlighted in colour when you move the cursor over it allowing quicker access to the program. Another option could be added including the user guide which could connect with a word document .file or video of the program working. This could be further extended to by verbal communication, like the banking systems information where options are provided offering option 1, option 2 or 3 etc.

I thought it would be a good idea to add the ability to send the business customer an email when the account was overdue for payment. So I have added an extra option and a new parameter in all of the constructors. I have called this option 9, ‘Send Email BALANCE OVERDUE’. This will open *Outlook Express* with customer email address and warning all ready to be sent. This will enable the company to let their customer know when the balance hasn’t been paid and will also inform them of suspension of services. I have updated the user guide and the option 8 ‘Help For User’.

**Overall Summary**

Java coding has given a solution to the problems encountered by this company. I have learned from the experience the importance of the life cycle developing process, understanding customer specifications properly, planning, implementing, testing and feedback.

Appendix A.

**Interview**

**Gerard Houston 22.5.17**

**Q. Do you think the program menu meets the requirement?**

A. Yes, it mostly does, there could be some improvement on the user guide, it seems too basic. More screen shots to assist the first-time user. These should show valid and invalid inputs/outputs as well.

**Q. Is the program easy to use and understand?**

A. Generally it is okay but the Help For User (option 8) is quite long, you could shorten it and make it easier to scroll up and down to read it. When I checked some of the outputs, I could see that the balance for the account was appearing with too many decimal places, this is unnecessary. Also, the outputs for erroneous data needs revised as it doesn’t explain what the user should do next properly.

**Q. Do the main options work and calculate properly?**

A. All the options do what they should, right calculations are doing what they should and giving the user the correct outputs.

**Q. Would you say the program is easy or hard to operate?**

A. Very simple, anyone could work it with ease.