**Assignment 1 – ITC 322**

**Discussion:**

**Task1** Writing BaseAccount wasn’t too hard as thankfully an outline of the code was already given from interact. This includes some aspects such as the passing as object, which is incredibly easy but I didn’t understand from abstract and descriptive point of view. It was only after the “for loops” and such, that was laid out, that I understood.

It was nice how the unique account numbers and the existing methods were programmed into the file beforehand, only implementations needed. The comments and explaining took a bit of research though. After commenting I did the Javadoc’s, so that was a bit different and rearranging where certain explanations go. Not sure if we supposed to use custom Javadoc annotation but I used the default ones. As for actual code, there weren’t many lines needed for this task, just using the built-in java String’s methods of equals and compare and some casting. I did make the mistake of manually programming a compareTo but then I found out how easy was to use the built-in one. I still commented the expected results of it though.

**Task2** Cheque account was smaller than expected. The programming of the many “test of positive” if statements were annoying. The factoring in the transaction charge proved somewhat challenging though, simply because it was to get balance of fewer lines and dealing with + and – in the different methods getting everything done at once, it required some thought. I’m not sure of what the point was of having credit limit when it wasn’t measured against at all. Both Withdrawal and Deposit were transactions so they could not leave the account overdrawn, so what could? Nonetheless it programmed to work as if such feature was to be used in the future.

I used tabs for table output as it seemed nice and neat. The String.format also proved useful for rounding which I didn’t know it could, until I tested it. I set the transaction charge as a variable dedicated to the class and put it as final. I knew it wouldn’t change so it made sense. Account Type was something I added after finishing the program as didn’t feel it the program outputted as much detail as I’d like. So whilst working in the confines the outline, I added it. I don’t know though if one could make some sort of “abstract” variables where it must be defined in the subclasses however. I will look it up next time because it would make sense to have account type variable compulsory defined.

**Task3** SavingsAccount was similar difficulty, relatively easy. Using “super” for creating itself with base account was simple. Due to there being no extra variables associated with the actual savings account itself and the withdrawal and deposit methods having no extra checks, they very easy. The “must be positive” was still annoying. I still used the account type of the output of “toString” with tabs and it worked perfectly, no issues. Although I could have changed the input interest rate but I took it to mean it’s whole amount of “8.0”. So a little calculation was needed but that was fine as there were other rules to follow and this may have been one as well.

**Task4** Test Accounts. Surprisingly, this took the most strength to do. I do contribute some of it to the table and the little extra type and credit limit added on but was still a pain. The check and casting took a bit to do, especially for the withdrawing code. This was preliminarily due to the subject outline laws imposed as such that the “not all classes are expected to have a withdrawal method”. As such, I had to create two if statements with one nested so that a wrong result was not printed on the screen.

A point about the pre-written code, the initialiser of the accounts was just “Accounts” not “BaseAccount”, I didn’t see how this would have worked earlier but I’ve open to suggestions, a rename was added. A had to also create temp object on the fly so that I could access some methods specific to some accounts. I have a feeling that there I another way but alas, I’m out of time. I figure out funny enough that just as Strings are objects, so are accounts (I don’t why I didn’t found out earlier). So then I didn’t have to change variables back, they were already changed as I using references.

**How to run:**

1. Open **Eclipse** IDE for Java EE Developers
2. Open the “**file**” menu
3. Click “**Import**”
4. Expand “**General**”
5. Double click “**Existing projects into workspace**”
6. Browse for the **root directory** (the one with .project files etc.)
7. Import “**ITC322\_assign1**”
8. **Expand** and **browse** to the tasks
9. **Double click**
10. Click **Run** (top toolbar) > click R**un**
11. **Enjoy.**

**Java Documents:**

The java docs were written into the code and generated using the Java Development Toolkit’s JavaDoc.exe. The generated html files reside in “**ITC322\_assign1\doc**” and can be opened by accessing “**index.html**” with your preferred web browser.

A screenshot of the output is located at the bottom of this document.

References

How to run retrieved from previous assignment:

Graham, J (2013), ITC 313 in *Assignment 2*

Assignment class files obtained from:

http://interact.csu.edu.au/portal/site/ITC322\_201430\_B\_I/page/aa0e4576-d62c-4ea8-801a-2fc80147f0e4

