COMP201 Assignment 1

Asessor: T. Carroll

1 Details

Module COMP201

Weighting 20% of module grade

Deadline 12 noon, Friday 8th November 2019

Purpose of Assignment To assess student ability to analyse, generate, and document user requirements; to assess student ability to create a software product that is compliant with user requirements.

Learning Outcomes Assessed: The following learning outcomes are assessed, as per the module specification:

LO 2: Understand the need to design systems that fully meet the requirements of the intended users including functional and non functional elements;

LO 4: Be fully aware of the principles and practice of an O-O approach to the design and development of computer systems;

LO 5: Be able to apply these principles in practice;

LO 6: Produce O-O requirements and design documentation in UML which demonstrates the features of good design such as loose coupling and high cohesion;

2 Brief

A Credit Union operates in a similar way to a commercial bank, though with a community-based outlook, holding members' savings, and loaning money to other members. Membership is normally restricted to residents of a particular community, or members of a particular profession, and capital is normally held for loans to members, as opposed to being invested for profit.
Consider the following scenario:
The Rutherford Village Credit Union was established in 1932, and to this day continues to use a paper-based system for book keeping tasks and account management. One day each week, from a function room in the village community center, the clerk facilitates the operation of the credit union. It has comissioned you to create a *prototype* system that models a back-end server, which will facilitate the actions of *Persons* upon the *Accounts*. It is hoped that this will pave the way for automation of some tasks in the future, thus helping the credit union to expand.
The Credit Union has two types of *Person*: a *Clerk* and a *Customer*. The Credit Union has two types of *Account*: a *Saving* account, and a *Loan* account, each of which can be *opened* (with a non-zero balance) and *closed*.
A saving account can be opened with a non-zero positive amount of money. A loan can be opened with a non-zero negative amount of money, which represents the balance as the customer's debt to the credit union.
The requirements for the *Saving* account are:

- Accept a *Payment*, which increases the *balance* by the relevant amount
- Allow a *Withdrawal*, which decreases the *balance* by the relevant amount

- Withdraw all funds upon closure
- Only allow a balance which is 0 or greater.

The requirements for a *Loan* account are:

- Accept a *Payment*, which increases the *balance* by the relevant amount
- Only allow a balance which is 0 or lower
- Allow an interest rate to be applied to the balance, which decreases the balance by the relevant amount
- Only allow closure when the balance is 0.

A *Clerk* shall be able to do the following:

- open a saving account on behalf of a customer, taking the cash opening funds from the customer
- close a saving account, giving the customer their balance in cash.
- open a loan on behalf of a customer, provided the amount is no more than £5000, no more than 5% of total capital held by the credit union, and the credit union has enough capital to loan the money. They give the customer the cash.
- Apply a weekly interest rate of 0.01% on all outstanding loan balances in the credit union.

A *Customer* shall be able to do the following:

- Withdraw money from their savings account
- Pay money into their savings account
- Pay money towards the outstanding balance on their loan account

2.1 Assumptions

You are make the following assumptions about the scenario:

- The Credit Union is small. On days where the Credit Union clerk is in the office, they will run the system on a single laptop computer.
- This is a prototype system; the full implementation would have a user-friendly front-end, and would also be linked to a database to hold all information with persistence.

Task 1 (35%) Produce **UML Use-Case Diagrams and Use-Case descriptions** for the described system. You shall consider only *human actors* for this task. You can split your answer into multiple diagrams, if needed.
For drawing the UML diagrams, you can use software or hand draw them **neatly** on plain paper, and scan them in. Please be aware that if they can not be read due to bad handwriting, bad quality scan, etc.... then the mark will be 0 for that diagram.

Task 2 (5%) Identify **5 functional requirements** of the described system.

Task 3 (10%) Identify **5 non-functional requirements** of the described system, using the description of the scenario. You shall propose a mechanism and criteria that make these non-functional requirements *verifiable*, i.e. Describe a technique that can objectively test them.

Task 4 (40%) You have been given a partial implementation of the prototype system, in the Java language. **Complete this implementation**, so that it compiles on a department computer and satisfies the requirements. Look for "//Todo:" in order to find areas of the source files that require completion. You may use the `CreditUnionUser.java` file to test your program, ensuring it meets the requirements. Use of the `instanceOf()` operator is permitted. Your submitted code should be well laid out and well commented, following good practise conventions.

Task 5 (10%) Create a **UML Sequence Diagram** for the application and opening of a Loan. Use your knowledge of the system obtained over this assignment to correctly identify the objects you must consider. For drawing the sequence diagrams, you can use software or hand draw them **neatly** on plain paper, and scan them in. Please be aware that if they can not be read due to bad handwriting, bad quality scan, etc.... then the mark will be 0 for that diagram.

3 Deliverables

You are required to deliver the following items:

1. PDF Format Document containing the diagrams and written answers for Tasks 1,2,3, and 5. It should have your Name, University Username, and Student ID at the top.

2. Java Source code files (*.java files) for the answer to Task 4. You should submit all source files that are required, even if you have not edited them from the original version. Source files should contain your Name, University Username, and Student ID at the top, in the comment block space provided.

4 Submission

You are to submit a zip file (`"studentID.zip"`, where `studentID` is your numerical 9-digit student ID number, found on your Student ID card) electronically via SAM, no later than the deadline stated in Section 1. This zip file should contain that which is detailed in Section 3.
You can access the SAM submission system at the following URL: `https://sam.csc.liv.ac.uk/`