FACULTY OF COMPUTERS AND ARTIFICIAL INTELLIGENCE, CAIRO UNIVERSITY

CS213: Programming II
Year 2022-2023
First Semester

Assignment 2 – Version 1.0

Course Instructors:

Dr. Mohammad El-Ramly

Revision History

Version 1.0	By Dr Mohammed El-Ramly	26 Oct.2022	Main Doc

CS213: Object Oriented Programming
Assignment 2 (7 marks + 1.5 + 1.5 bonus) – Version 1.0

Cairo University, Faculty of Artificial Intelligence and Information

Objectives

This assignment aims to learn OOP concepts in C++ and how to use OOP modeling and design with C++ to build systems with intermediate complexity.

Instructions

1. **Deadline is Tues 12th of November 2022 @ 11:59 pm. Weight is 7 marks** + **3 bonus** marks.
2. Students will forms teams of three students **from the same lab/section**.
3. Please submit **only work that you did yourself**. If you copy work from your friend or book or the net **you will fail the course**. تسليم حلول منقولة من أى مصدر يؤدى إلى الرسوب فى هذا المقرر
لا تغش الحل أو تنقله من أى مصدر و اسألنى فى أى شئ لا تفهمه لكن لا تنقل الحلول من النت أو من زملائك أو أى مكان

Task 0 (0 marks)

1. Review OOP C++ concepts and syntax.
2. Create a **private GitHub** repo for the project and **use it for development.**

Task 1 (3 marks) - Classes, objects, abstraction and composition

Students should divide the work as suggested below and then **should integrate their code** together and make sure it works properly.

Different variations of types `int` and `float` exist in C++ and other languages. They are limited by min and max values depending on the number of bytes used to store the number. We need versions of these types with unlimited bounds. In this problem you will develop a C++ class that can hold a **real number** with unlimited number of digits and the supporting math functions.

We will use `BigDecimalInt` class and OOP composition. Use the given `BigDecimalInt` class. Composition means that an object is composed of objects of another kind. You need to think of an efficient way for doing so. E.g., you can choose to store the whole real number in a `BigDecimalInt` and the position of the decimal point in a separate attribute. Or you may store the integer and fraction parts in two separate `BigDecimalInt`. Think of a best solution.

`BigDecimalInt` has the following public interface:

```
a. BigDecimalInt (string decStr); // Initializes from string & rejects bad input
b. BigDecimalInt (int decInt);     // Initialize from integer
c. BigDecimalInt operator+ (BigDecimalInt anotherDec); // member fn
d. BigDecimalInt operator- (BigDecimalInt anotherDec); // member fn
e. bool operator< (BigDecimalInt anotherDec);          // member fn
f. bool operator> (BigDecimalInt anotherDec);          // member fn
g. bool operator==(BigDecimalInt anotherDec);          // member fn
h. BigDecimalInt operator= (BigDecimalInt anotherDec); // member fn
i. int size();                                         // member fn
j. int sign();                                         // member fn
   It also overrides the << operator as follows as friend:
k. friend ostream& operator << (ostream& out, BigDecimalInt b)
```

CS213: Object Oriented Programming
Assignment 2 (7 marks + 1.5 + 1.5 bonus) – Version 1.0

Cairo University, Faculty of Artificial Intelligence and Information

It is required to build a class **BigReal**. An object of this class represents a real number of an arbitrary length. The public interface of your class will be as follows. For functions where default implementation of C++ is enough and works fine, you do not need to re-implement them. **Break your code into: header file, implementation file and application file.**

```
class BigReal {
  private:
       ...
  public:
       BigReal (double realNumber = 0.0);   // Default constructor
       BigReal (string realNumber);
       BigReal (BigDecimalInt bigInteger);
       BigReal (const BigReal& other);      // Copy constructor
       BigReal (BigReal&& other);           // Move constructor
       BigReal& operator= (BigReal& other); // Assignment operator
       BigReal& operator= (BigReal&& other);     // Move assignment
       BigReal operator+ (BigReal& other);
       BigReal operator- (BigReal& other);
       bool operator<  (BigReal anotherReal);
       bool operator>  (BigReal anotherReal);
       bool operator== (BigReal anotherReal);
       int size();
       int sign();                                          /
       friend ostream& operator << (ostream& out, BigReal num);
       friend istream& operator >> (istream& out, BigReal num);
};
```

You should make maximum possible use of the functions already in **BigDecimalInt.**

It is required to separate your design into a header file and an implementation file. You should also validate that the passed parameter in case of string is a valid real number. Note that **1.** and **.1** are valid real numbers but **1.1.1** is not valid real number.

Your new class will support writing application programs like this:

```
BigReal n1 ("11.9000000000000000000000000000000001");
BigReal n2 ("2333333333339.1134322222222292");
BigReal n3 = n1 + n2;
cout << n3;
n3 = n3 + BigReal (0.9);
```

Test your class with a small application and 12 test cases at least the test all these functions.

Team member with the smallest ID will do the first colored set of functions, the second will do the second (+ and -) and the third will do the third.

What to deliver?

1- **Written code in standard C++ not using third-party libraries. Put in a separate directory.**
2- Name your file **A2_Task1_YourGroup_YourIDs.cpp** or **.zip** (if more than one file)
3- In the pdf report, include a detailed class diagram showing class, relations and attributes.

CS213: Object Oriented Programming
Assignment 2 (7 marks + 1.5 + 1.5 bonus) – Version 1.0

Cairo University, Faculty of Artificial Intelligence and Information

Task 1 (2 marks) - Classes, objects, abstraction and composition

Students should divide the work **as they like** and then **<u>should integrate their code</u>** together and make sure it works properly.

Banking System

Model, design and develop a banking application. The banking application allows the user, i.e., the bank employee to create a bank account for a specific client. It allows him to list all the available bank accounts. For each account, it allows him to display the account details, withdraw money and deposit money.

There are two types of bank accounts. The first type is the basic **`BankAccount`**. It holds the following data:

- **`account ID`**
- **`balance`**

The following methods apply to this class:

- **Constructor.** There are 2 constructors. The first sets the balance to a given value. The second is a no-argument constructor and it sets the balance to 0.
- **Setters and getters**. These methods allow accessing the private data fields.
- **`withdraw`**. It withdraws an amount of money from the account if the balance is sufficient.
- **`deposit`**. It deposits an amount of money in the account.

The second type of accounts **extends** the basic Bank Account and may have some extra data fields and operations. It is called **`SavingsBankAccount`**. This account requires the user to keep a minimum amount of money in the account, which is called the minimum balance, as long as the account is open. It also requires him to make deposits that are not less than 100 a time. So, it has the following additional data field:

- **`minimumBalance`** This minimum balance takes a default value of 1000 L.E.

It has the following methods plus those inherited from the parent class:

- **Constructor.** The constructor sets the value of the initial balance and the minimum balance. Initial balance should be >= min balance.
- **Setters and getters**. These methods allow accessing the private data fields.
- **`withdraw`**. It overrides the method withdraw to allow withdrawing money but not below the minimum balance.
- **`deposit`**. It deposits an amount of money in the account but only if the amount is >= 100 LE.

There is also a **`Client`** class which holds the basic information of a client like his name, address and phone number. It holds a pointer to his bank account. An account also points to its owner.

The main class that runs the application is **`BankingApplication`**. This class displays the main menu and accepts the user's choice. It maintains a list of accounts and clients. It allows the user to perform operations on a bank account.

A sample operation of this application looks like the following:

CS213: Object Oriented Programming
Assignment 2 (7 marks + 1.5 + 1.5 bonus) – Version 1.0

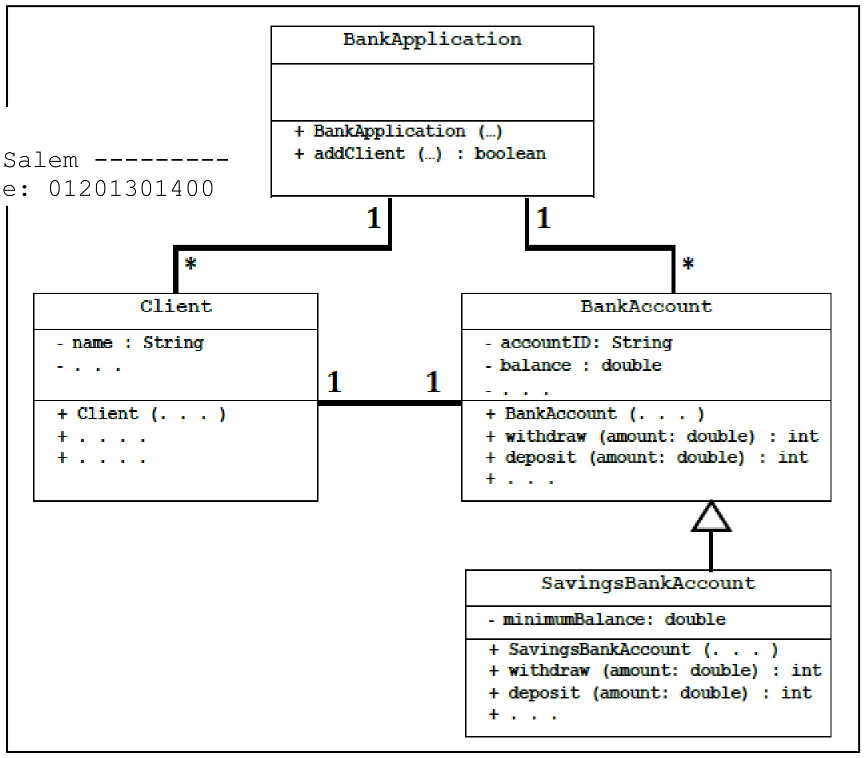
Cairo University, Faculty of Artificial Intelligence and Information

```
Welcome to FCAI Banking Application
1. Create a New Account
2. List Clients and Accounts
3. Withdraw Money
4. Deposit Money

Please Enter Choice =========> 1
Please Enter Client Name =========> Ahmed Ali Salem
Please Enter Client Address =======> 5 Batn Elzeer St., Giza
Please Enter Client Phone =======> 0120130140
What Type of Account Do You Like? (1) Basic (2) Saving – Type 1 or 2 =========> 1
Please Enter the Starting Balance =========> 1500
An account was created with ID FCAI-001 and Starting Balance 1500 L.E.
-----------------------------------------------------------------------
Welcome to FCAI Banking Application
1. Create a New Account
2. List Clients and Accounts
3. Withdraw Money
4. Deposit Money

Please Enter Choice =========> 3
Please Enter Account ID (e.g., FCAI-015) =========> FCAI-001
Account ID: FCAI-001
Acocunt Type: Basic
Balance: 1500
Please Enter The Amount to Withdraw =========> 1550
Sorry. This is more than what you can withdraw.
Please Enter The Amount to Withdraw =========> 40
Thank you.
Account ID: FCAI-001
New Balance: 1460
-----------------------------------------------------------------------
Welcome to FCAI Banking Application
1. Create a New Account
2. List Clients and Accounts
3. Withdraw Money
4. Deposit Money

Please Enter Choice =========> 2
-------------------------- Ahmed Ali Salem ---------
Address: 5 Batn Elzeer St., Giza Phone: 01201301400
Account ID: FCAI-001 (Basic)
Balance: 1460
-----------------------------------
```



Your task is:

1- Complete the given UML model for the application domain of the bank. Add any necessary missing details, attributes, methods, etc. to support the role of each class according to the description above. Draw it using a UML tool.

2- Develop the classes you designed in C++ and test them.

3- Integrate the classes together and develop an integrated banking application according to the description above.

4- Write 5 test cases, which involve creating 5 clients and 5 accounts of different types and use them to test all the functionalities of the program.

CS213: Object Oriented Programming
Assignment 2 (7 marks + 1.5 + 1.5 bonus) – Version 1.0

Cairo University, Faculty of Artificial Intelligence and Information

What to deliver?

1- **Working code in standard C++ not using third-party libraries in a separate directory.**
2- Name your file **A2_Task2_YourGroup_YourIDs.cpp** or **.zip** (if more than one file)
3- In the pdf report, include the detailed class diagram showing classes, relations and attributes.
4- In the report, include a work break-down table showing who did which part.

Task 3 (2 marks) – Individual problems from Sheet 2

Take the smallest ID % 6, e.g. 20210398 % 6 = 4. Then this team member will solve problems 4 and 1. Next ID will solve problems 5 and 2 and the third will solve 0 and 3. These problems will be in **Sheet 2 under Acadox.**

What to deliver?

- **Working code in standard C++ not using third-party libraries in a separate directory.**
- Name your file A2_SheetPb**XX**_YourID.cpp or .zip (if more than one file) (XX is pb num)
- In the report, write who did which problems.

Group Bonus 1: Task 3 (1.5 mark) – Static Code Analysis and Code Quality

Software is expensive to develop and is expected to live for several years. Many people over a long period of time might have to work on the same program. It is extremely important, **ethically**, **professionally** and **economically**, to develop high quality code.

In this task, it is required for **all team** to cooperate in using one of the static analysis and code quality tools to analyze their solution for tasks 1 and 2. They can use student version of PVS studio https://pvs-studio.com/en/order/for-students/ or anther tool they have access to, e.g. SonarCube. (A video on PVS is here but there are many https://www.youtube.com/watch?v=vYW6TOwFK2M)

These tools look for weak and vulnerable parts of the code, bad practices, misuse of language constructs **and can enforce the application of coding style.** (to some extent, for example, it cannot decide for you if variable names are good, but it can check indentation and spacing)

Students will analyze the code with tool, find all important quality issues, and then fix them and produce better code.

What to deliver:

1- A written report of the experience with the tool and how useful it is and easy / hard to use.
2- A list of the most important issues found (they can be 100s, only report the important ones)
3- A modified and clean version of the code.
4- A **work break-down** table explaining who did what in this part.

Group Bonus 2: Task 5 (1.5 mark) – Code Generation

Writing code is a costly and time consuming job. Software design is an essential step before writing code that involves specifying the different parts of the software, e.g., classes, database tables, algorithms, etc. Any respectable software development team would do software design before starting coding. **Would not be great if we can generate the code (even partially) from the artifacts produced in design like class diagram?**

Your task it to use a UML design tool (like MS Visual Paradigm https://www.visual-paradigm.com/download/) to model the class diagram of your banking system of task 2 up to the detailed level and generate the code for it. Then complete the code and run the app.

What to deliver:

1- A written report of the experience with the tool and how useful it is and easy / hard to use.
2- A comment of the quality of the code generated. Is it good quality? Is it clean / readable?
3- A **working banking application** that is generated from the tool.
4- A **work break-down** table explaining who did what in this part.

Submission Instructions

Team will submit into acadox the following:

1. A zip file with the following components.
2. A pdf report with the following items.
   - The document should have a cover page like this.
   - A screen shot for every GitHub account for shared projects.
   - The required class diagrams, work break-down tables and reports needed in bonus tasks.
3. Team will create a project in **GitHub** to upload code there.
4. The source code of each program in a separate folder with suitable name for the folder. For individual problems, each student should put his solutions divided into 2 folders, one for each problem and folder name should have his name and ID.
5. Each team member will work individually on his part. **But the team must provide ONE integrated and working program and report.**
6. Team members are expected to help each other but not do work of others.
7. Team members are responsible of testing all the programs and making sure they work.
8. **All team members must understand the details** of all programs and be able to explain it or even modify it if needed. TA can ask any team member about any of the programs developed**.**
9. **Ask your TA** about the discussion time of your work.

Cairo University
Faculty of Computers and Information

CS251

Software Engineering I

Project Description

2015

Version 1.0

Project Team

Dr Mohammad El-Ramly

m.elramly@fci-cu.edu.eg

Marking Criterion

1. 2.0 for developing your part in the group problem (Task 1) correctly.
2. 1.0 for integrating the code and developing one working program.

1. 1.0 for developing your part in the group problem (Task 2) correctly.
2. 1.0 for integrating the code and developing one working program.

3. 2.0 for developing a solution for the individual problems (Task 3).
4. -1 for not understanding the solutions of individual problems of others

5. 1.5 **Group Bonus1** for checking code quality by the tool and producing a high quality report
6. 1.5 **Group Bonus2** for producing high quality model and generating the banking system.