Cairo University, Faculty of Computers and Information

FACULTY OF COMPUTERS AND INFROMATION, CAIRO UNIVERSITY

CS213: Programming II
Year 2018-2019
First Semester

Assignment 3 – Version 2.0

Course Instructors:

Dr. Mohammad El-Ramly
Dr. Mohammad Nassef

Revision History

Version 1.0	By Dr Mohammed El-Ramly	29 October 2018	Main Doc

CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

Cairo University, Faculty of Computers and Information

Objectives

This assignment aims to:

1- Train students on modeling a system using OOP and UML.
2- Train students on programming a system using OOP C++.
3- Train students on using recursion to develop solutions for computational problems.
4- Train students on team work and cooperation.

Instructions

1. This assignment has short time. Start now. لا تنتظر لآخر لحظة و ابدأ الان
2. These instructions must be followed to get the full mark. يجب اتباع هذه التعليمات بكل دقة
3. **Deadline is Sat 17th of November 2018 @ 11:59 pm. Weight is 4 marks.**
4. **Form teams of only two students. Work with a team from the same group.** Team consists of students whose IDs do not end with the same digit**.** For example, 2017023 and 20170433 cannot be in one team because they both have IDs ending with 3.
5. If you have any problem or difficulty, ask your TA or the professor. **We love to help you.**
6. 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 1 (2.5 marks) – OOP Modeling and Implementation Using C++ - Group

In this task, you will develop a system according to your ID. Add your IDs and divide by 3.

If the remainder (ID1 + ID2) % 3 is 0, solve problem 0. If it is 1, solve problem 1. If it is 2, solve problem 2.

For each problem, you will do the following:

1- Follow the approach explained in the lecture to model object-oriented systems and find classes.

2- Draw a **complete and detailed** UML class diagram for your system using a UML modeling tool like ArgoUML or StarUML or Visio. The diagram should show all the classes, all their attributes and methods and the relation between classes. It should match the actual code.

3- Use all the concepts you learned in OOP like inheritance relation, association relation, polymorphism, virtual functions, overloading, friend functions, constructors, etc.

4- Implement the system and separate header from implementation from application.

5- Team should create together the definition of the classes and functions (*.hpp file) and when they all agree on the **public interface of each class and header of each function,** each of them can work on his part and **it will be easy to integrate the work together.**

6- Add comments at the start and within the program files.

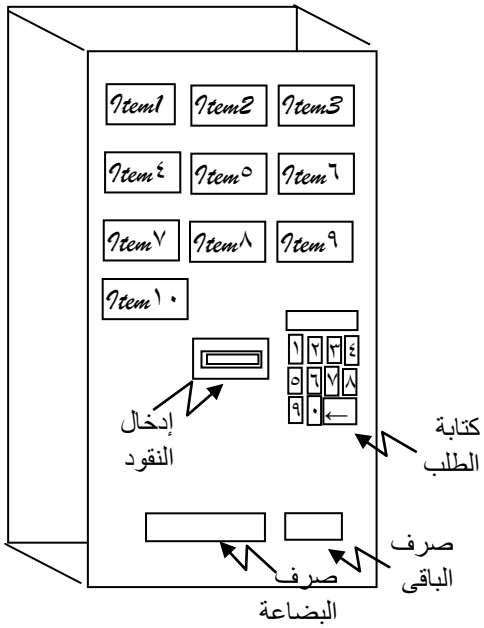
7- Write a report in which you provide (1) Cover page, (2) Your UML class diagram (image take from the modeling tool), (3) Explanation of the role of each class and the relation between classes, and (4) How you used all OOP features.

CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

Cairo University, Faculty of Computers and Information

Problem 0 – Vending Machine Simulator

وصف النظام المطلوب



- المطلوب بناء نموذج لمحاكى ماكينة البيع الذاتى Vending Machine للمشروبات الباردة و الحلوى و هى الماكينة التى توضع فيها النقود و يتم طلب أحد الأشياء التى تبيعها فينزل المطلوب و باقى النقود للمشترى وتعمل كالتالى:
- عند تحميل البرنامج يبدأ بقيم تلقائية Default Values لنوعية الحلوى و المشروبات الموجودة فيه و كذلك بقيم تلقائية لكمية النقود المتاحة فيها من حيث فئات النقدية و العملات و عددها.
- الماكينة تحمل 10 أنواع مختلفة و من كل نوع تحمل عشرة وحدات و محدد لديها سعر الوحدة
- للماكينة نوع واحد من المستخدمين و هم المشترون.
- المشترى يقوم بوضع عملات معدنية أو أوراق نقدية من فئة نصف جنيه و جنيه و خمسة جنيهات و عشرة جنيها تو عشرين جنيها.
- ثم يقوم المشترى بضغط رقم النوع الذى يريده و إذا كان قد وضع نقودا كافية فإن الماكينة تصرف له العنصر المطلوب و تصرف له باقى المبلغ المدفوع بأكبر فئة نقدية متاحة فمثلا لو باقى له ٦.٥ جنيه فإنها تصرف له ورقة فئة خمسة جنيهات ثم ورقة أو عملة فئة جنيه وورقة أو عملة فئة نصف جنيه.
- إذا كان النوع المطلوب موجودا يتم صرفه و صرف باقى النقود.
- إذا كان النوع المطلوب غير موجود فإن الماكينة تظهر رسالة لتخبر المستخدم بعدم وجود النوع المطلوب و تطلب منه إما إختيار نوع آخر أو إستعادة النقود.
- إذا أراد المشترى إلغاء العملية بعد إدخاله النقود فإنه يقوم بإدخال الإختيار ٠ فتتم إعادة نقوده له.
- إذا انتهت كل الأنواع و تم إستهلاك كل البضاعة فى الماكينة فإنها تتوقف عن قبول النقود و الإختيارات من المشترين.

Description of the Required System

- It is required to build a model for a vending machine simulator that simulates the operation of a vending machine which. Do not write one big **`main`** or use structured programming. **Model the system using OOP and classes. Find the main classes and how they interact.** See the example made in lectures.
- The machine sells food items and drinks by inserting money in it and requesting an available item, then the machine dispatches this item and the rest of the money.
- When the machine (or model) starts, it starts with a default quantity of each item it sells and a default amount of money it terms of which coins and bills are available and the number available of each coin or bill.
- The machine can carry up to 10 items, each has 10 unit maximum. Each item has a specific price.
- There is only one type of users of the machine, which are buyers.
- A buyer puts coins (1/2 LE, or 1 LE) and/or bills +(5, 10 or 20 LE) and presses the number of the item he wants.
- If he puts enough money, the machine dispatches the required item and the change.
- If the required item is not available, the machine will display a message informing him that the item is not available and gives him the choice of picking another item or returning his money.
- If the customer wants to cancel an operation, he presses 0 and his money is returned.
- If the machine runs out of stock in all items. It stops accepting money or choices from buyers.

CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

Cairo University, Faculty of Computers and Information

Problem 1 - Appointments Calendar

Create class **`Time`** that represents a time value (in hour, minute and AM/PM format like 03:18 PM). In this class, you should overload the operators **`<, >, ==, !=`** in order to be able to compare two time objects. In addition, you should overload the **`>>, <<`** operators to facilitate getting and displaying time values from/to user.

After that, create class **`Appointment`** that can represent the time interval of some appointment. An object from this class should have a start time and an end time. You should overload the **`<=, >=, <, >, ==, !=`** relational operators in order to be able to compare two appointments. Moreover, you should overload the **`>>, <<`** operators to facilitate inputting and displaying appointment values from/to user. These two operators should make use of the same operators overloaded for the **`Time`** class. End time must be after start time.

Create a class named **`DayAppointments`** that contains a dynamic array of appointments. An object from this class should overload the **`=, ==, !=, [], >>, <<`** operators, the copy constructor and the destructor. You should not allow the user to enter two overlapped appointments in the same day.

Create a class named **`Calendar`** that contains an array of **`DayAppointments`**. This class should be able to keep track of the appointments belonging to the seven days of the week.

Write a **`main`** function that loops until the user exists and takes the user's choice and perform the chosen operation as many times as the user wishes. The user can insert an appointment for a specific day or for multiple days in the same week. He can display his appointments for a day or for the week. He can delete an appointment. He can make the appointments of a day equal to another day. No two appointments should overlap. He can check if two days have equal appointments.

The classes' member variables can be like this:

```
class Time
{
      private:
            int hour;
            int minute;
            string daynight;
      public:
            …
};

class Appointment
{
      private:
            Time start;
            Time end;
      public:
            …
};
```

```
class DayAppointments
{
      private:
            Appointment * appointments;
            int numOfAppointments;
            string weekDay;
            // like Saturday, Monday … etc
      public:
            …
};

class Calendar
{
      private:
            DayAppointments * days;
            int numOfDays;
      public:
…
};
```

CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

Cairo University, Faculty of Computers and Information

Problem 2 – 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 to deposit is 100 or more.

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:

CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

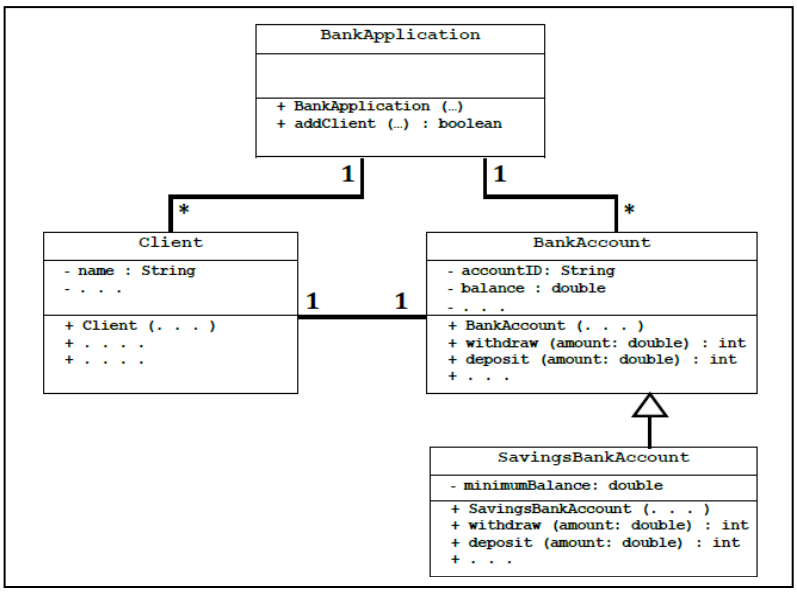
Cairo University, Faculty of Computers and Information

```
Welcome to FCI 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 FCI-001 and Starting Balance 1500 L.E.
-----------------------------------------------------------------
Welcome to FCI 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., FCI-015) =========> FCI-001
Account ID: FCI-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: FCI-001
New Balance: 1460
-----------------------------------------------------------------
Welcome to FCI 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: 0120130140
Account ID: FCI-001 (Basic)
Balance: 1460
---------------------------------
```



Your task is:

1- Complete the given UML model for the application domain of the described 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.

CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

Cairo University, Faculty of Computers and Information

Task 2 (1.5 marks) – Individual

1. Each team member will solve one of the following recursion problems according to their ID.
2. Some problems are taken from this book: **Problem Solving with C++**, ninth edition and **Programming Abstractions in C++,** 2012.

3. Remember the three important laws of recursion:
   - A recursive algorithm must have a **base case**.
   - A recursive algorithm must change its state and move **toward the base case**.
   - A recursive algorithm must **call itself**, recursively.
4. Each student **must solve the correct problem** according to the following: IDs ending with X will solve the corresponding problem, where X is:
   0. Problem 6 on page 829 (Problem Solving)
   1. Problem 7 on page 829 (Problem Solving)
   2. Problem 2 on page 828 (Problem Solving)
   3. Problem 9 on page 831 (Problem Solving) or problem 10 page 385 (Prog. Abstractions)
   4. Problem 9 on page 351 (Programming Abstractions)
   5. Problems 7 and 8 on page 350 (Programming Abstractions)
   6. Problem 6 on page 382 (Programming Abstractions)
   7. Problem 8 OR 9 on page 384 (Programming Abstractions) (only one of them)
   8. Problem 4 on page 380 (Programming Abstractions)
   9. Problem 5 on page 381 (Programming Abstractions)

Task 3 (0 mark if done and -1 if not done) – Good Quality Code & GitHub

There is no mark for this task, but if not done, you lose marks.

GitHub. GitHub is a version control server that allows a group of programmers to work together and track who changed which files. It is also a place where programmers show off their best projects and attract employers to hire them.

Quality. No program stays the same. It will need to change to fix bugs, add new features, etc. So, it is very important to write high quality readable code so that you or other developers are able to review and modify this code in the future. In this task, you will:

1- Add a header to your program saying who the author is, the purpose of the program, etc.

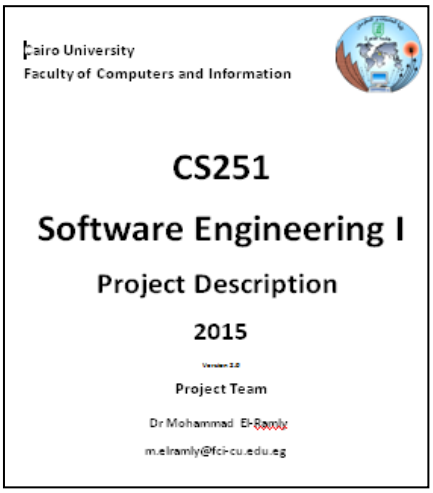
CS111: Fundamentals of CS
Assignment 3 (4 marks) – Version 2.0

Cairo University, Faculty of Computers and Information

2- Add a header for every function explaining what it does, what parameters it takes and what value it returns.
3- Write the code following C++ coding style. http://geosoft.no/development/cppstyle.html
4- Add comments to any part that is difficult to understand.

Submission Instructions

1. **Team will submit into acadox the following:**
   - A zip file with a pdf document with their names and IDs with a cover page like this **and report of Task 1.**
   - The source code of each program in a separate folder.
   - A screen shot for every GitHub account.
2. **Zip file name must be:**
   FCI-CS213-A3-2018-TAName-GroupNum-ID1-ID2.zip
3. Team will create a project in **GitHub** to upload code there.
4. Each team member will work individually on his part. **But the team must provide ONE integrated and working program for task 1.**


Cairo University
Faculty of Computers and Information

CS251

Software Engineering I

Project Description

2015

Project Team

Dr Mohammad El-Ramly

m.elramly@fci-cu.edu.eg

5. Team members are expected to help each other but not do work of others.
6. **All team members must understand the details** of all programs and be able to explain it or even modify it if needed.
7. Team members are responsible of testing all the programs and making sure they work.
8. TA can ask any team member about any of the programs developed and its code**.**

Marking Criterion

1. 0.75 for developing the UML class diagram and its description report **and how OOP features** Were used.
2. 1.75 for developing a working system using an OOP approach. (minor mistakes are forgiven)
3. Zero for any team member who did not do at least 25% of the code.
4. 1.5 for developing a working solution for the individual problem of Task 2.
5. -0.5 for **not** writing good quality code using coding style and adding headers and comments
6. -0.5 for **not** putting the code in a project in GitHub.
7. -0.25 for **not** naming the file properly.