Lab 17 Object-Oriented Programming

Introduction to Computer Science I

# Objectives:

After performing this lab, the students should be able to

* design classes that models real-world entities
* write class declarations that contain member variables and methods with the appropriate visibility
* implement basic member functions, including getters and setters

# Names of Lab Group Members:

## Activities

### Part A

1. **Explain the relationship between a class and an object.**

A class is like a blueprint. An object is an instance of a class. As an example, a class would be the schematics/blueprints to build a car. An object is the car itself. You can have multiple versions of the object. i.e. BMW 325i, Ford F-150.

1. **Provide at least two different reasons for using object-oriented programming.**

I) A way to have modular programming. Easier to debug your program if it’s broken up into manageable pieces.

II) Easier to maintain and modify existing code as you can create new objects with small differences to existing ones.

III) Can use a collection of shared libraries. These libraries can be re-used throughout your program.

1. **Design a class that models the two main entities involved in the following scenario:**

**You are a software developer at a major technology company. You are designing an OOP program that helps patients with dementia remember their daily activities and information about people. The program helps the user keep track of the time, description, and location of each daily activity. It also helps the user maintain a list of important contacts, including their names, contact information, and pictures. The program provides easy ways to retrieve, record, and update the information. It automatically provides reminders about activities based on the user’s current location. It also provides information about people that the user encounters, where available, based on images captured in real-time (for example, the user could be wearing a wearable camera that automatically captures pictures of people with whom the user is interacting).**

**What are two main entities that you would implement with a class? For each entity, determine the data members and member functions and record them in the following tables. Provide both the variable/function declarations as well as their descriptions.**

|  |  |
| --- | --- |
| **Entity 1** | **Description** |
| **Daily Activity** |  |
| **Member Variables** | **Description** |
| **Time** | **A double variable used to represent time.** |
| **Description** | **A string variable used to describe the type of activity being performed.** |
| **Location** | **A string variable used to describe the location the activity is being performed at.** |
|  |  |
|  |  |
|  |  |
| **Member Methods** | **Description** |
| **Default constructor** | **Initializes daily activity variables with -99 for easily locating unpopulated variables.** |
| **CreateActivity** | **Creates an activity** |
| **PrintActivity** | **Prints an existing activity** |
| **UpdateActivity** | **Updates an existing activity** |
| **DeleteActivity** | **Deletes an existing activity** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Entity 2** | **Description** |
| **PeopleInformation** |  |
| **Member Variables** | **Description** |
| **Important contacts list** | **Array used to store a list of names, contact information, and pictures.** |
|  |  |
|  |  |
|  |  |
|  |  |
| **Member Methods** | **Description** |
| **Default constructor** | **Initializes PeopleInformation variables with -99 for easily locating unpopulated variables.** |
| **CreateContact** | **Creates Contact with supplied information** |
| **UpdateContact** | **Updates Contact information** |
| **DeleteContact** | **Deletes Contact** |
| **PrintContact** | **Prints Contact information** |
|  |  |

### Part B

Provide your C++ source code and screenshots of your program outputs.

Write a class definition for a BankAccount class that contains the following members:

* Member variables
  + **account ID** (of the form xxxxx-xxxxx, where x is a digit)
  + **account holder name** (e.g. John Joe)
  + **account balance**
* Member methods
  + **Initialize(ID, name, balance):** sets the account ID to a given ID, the account holder name to the given string, and account balance to a given number
  + **SetHolderName(name):** sets the account holder name to the given name
  + **IncreaseBalance(amount):** increases balance by the given amount
  + **DecreaseBalance(amount):** if the given amount is no more than the balance, remove it from the balance and return true; otherwise, return false
  + **Deactivate:** deactivates the account by setting the account ID to 00000-00000, holder name to the empty string, and balance to 0
  + **IsActive:** returns true if the account is active and false otherwise (an account is inactive if the account ID is 00000-00000 and the holder name is the empty string
  + **Print:** displays the state of the account (values of the member variables in an organized format)
  + **GetID**: gets the account ID
  + **GetHolderName**: gets the account holder name
  + **GetBalance**: gets the account balance

Implement all class member functions.

Write a simple drive program that tests the BankAccount class. The driver program should create a BankAccount object and invoke each of its member methods. Display the states of the BankAccount object where appropriate. By observing expected outputs from the driver program, you should obtain assurance that your class is implemented correctly.