# NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCE

## CL 1004 - Object Oriented Programming Lab

## Semester: Spring 2022

## Lab 03

**Instructor: Muhammad Sudais**

**Email: muhammad.sudais.v@nu.edu.pk**

## Outline

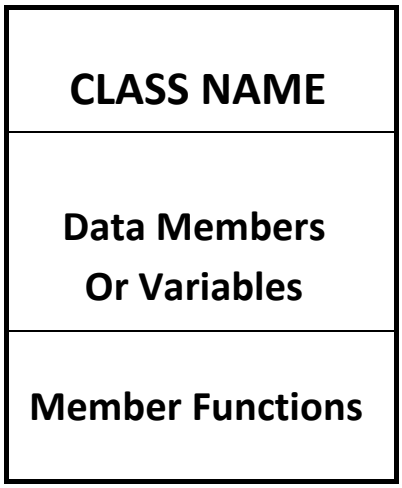
- Classes
- Objects
- Structures VS Classes
- Transformation from Procedural to Object Oriented Programming
- Example Programs
- Exercise

# CLASSES

A class is a programmer-defined data type that describes what an object of the class will look like when it is created. It consists of a set of variables and a set of functions.

Classes are created using the keyword **class**. A class declaration defines a new type that links code and data. This new type is then used to declare objects of that class.

In the UML, a class icon can be subdivided into compartments. The top compartment is for the name of the class, the second is for the variables of the class, and the third is for the methods of the class.




```
class class-name
{
access-specifier:  data

access-specifier: functions
};
```

## CLASS NAME

By convention, the name of a user-defined class begins with a capital letter and, for readability, each subsequent word in the class name begins with a capital letter.

## DATA MEMBERS

Consider the attributes of some real world objects:

**RADIO** – station setting, volume setting.
**CAR** – speedometer readings, amount of gas in its tank and what gear it is in.

These attributes form the data in our program. The values that these attributes take (the blue color of the petals, for example) form the state of the object.

## MEMBER FUNCTIONS

Consider the operations of some real world objects:

**RADIO** – setting its station and volume (invoked by the person adjusting the radio's controls) **CAR** – accelerating (invoked by the driver), decelerating, turning and shifting gears. These operations form the functions in program. Member functions define the class's behaviors.

## ACCESSOR AND MUTATOR FUNCTIONS

Accessor and mutator functions (a.k.a. set and get functions) provide a direct way to change or just access private variables. They must be written with the utmost care because they have to provide the protection of the data that gives meaning to a class in the first place. Remember the central theme of a class: data members are hidden in the private section, and can only be changed by the public member functions which dictate allowable changes. Let's see how we might code up these functions for our example class.

```
#include <iostream>
using namespace std;
class Student
{
    private:
        int id;
    public:
        int getId() //Accessor
        {
            return id;
        }
        void setId(int x) //Mutator
        {
            id = x;
        }
};

int main()
{
    Student s;
    int x = s.getId(); //Accesor
    cout<< "Student Id before mutation : "<<x<<endl << "Enter Student Id to Modify : ";
    int y;
    cin>> y;
    s.setId(y); //Mutator
    cout <<"Student Id after mutation : "<<s.getId();

}
```

# OBJECTS

In C++, when we define a variable of a class, we call it **instantiating** the class. The variable itself is called an **instance** of the class. A variable of a class type is also called an **object**. Instantiating a variable allocates memory for the object. **RADIO** r;

**CAR** c;

# Structures VS Classes

By default, all structure fields are public, or available to functions (like the main() function) that are outside the structure. Conversely, all class fields are private. That means they are not available for use outside the class. When you create a class, you can declare some fields to be private and some to be public. For example, in the real world, you might want your name to be public knowledge but your Social Security number, salary, or age to be private.

# TRANSFORMATION FROM PROCEDURAL TO OBJECT ORIENTED PROGRAMMING

```cpp
#include<iostream>
using namespace std;

double calculateBMI(double w, double h)
{
 return w/(h*h)*703;
}

string findStatus(double bmi)
{
 string status;
if(bmi < 18.5)
  status = "underweight";
else if(bmi < 25.0)
  status = "normal";// so on.
return status;
}

int main()
{
    double bmi, weight, height;
    string status;
    cout<<"Enter weight in Pounds ";
    cin>>weight;
    cout<<"Enther height in Inches ";
    cin>>height;
    bmi=calculateBMI(weight,height);
    cout<<"Your BMI is "<<bmi<<" Your status is "<<findStatus(bmi);

}
```

## Procedural Approach

```cpp
#include<iostream>
using namespace std;
class BMI
{
    double weight, height,bmi;
    string status;
    public:
        void getInput() {
            cout<<"Enter weight in Pounds ";
            cin>>weight;
            cout<<"Enther height in Inches ";
            cin>>height;   }
        double calculateBMI() {
             return weight / (height*height)*703; }
         string findStatus() {
            if(bmi < 18.5)
                status = "underweight";
            else if(bmi < 25.0)
                status = "normal";// so on.
            return status; }
        void printStatus()  {
            bmi = calculateBMI();
            cout<< "You BMI is "<< bmi<< "your status is " << findStatus(); }
};

int main()
{
    BMI bmi;
    bmi.getInput();
    bmi.printStatus();
}
```

## Object Oriented Approach

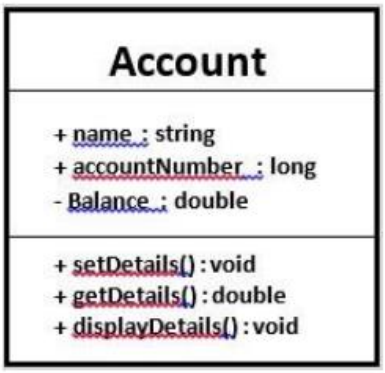
# EXAMPLE PROGRAM

```
#include<iostream>
using namespace std; class Account
{
private:
        double balance; // Account balance
public: //Public interface:
        string name; // Account holder long accountNumber;
        // Account number void setDetails(double bal)
        {
                balance = bal;
        }
        double getDetails()
        {
                return balance;
        }
        void displayDetails()
        {
                cout<<"Details are: "<<endl;
           cout<<"Account Holder:
           "<<name<<endl;
                cout<<"Account Number:
                "<<
           accountNumber <<endl; cout<<"Account
           Balance: "<<getDetails()<<endl;
        }
};

int main(){ double accBal; Account
currentAccount;
currentAccount.getDetails();

cout<<"Please enter the details"<<endl;
cout<<"Enter Name:"<<endl; getline(cin,
currentAccount.name); cout<<"Enter
Account Number:"<<endl;
cin>>currentAccount.accountNumber;

cout<<"Enter Account
Balance:"<<endl; cin>>accBal;
currentAccount.setDetails(accBal);
cout<<endl;
currentAccount.displayDetails();
return 0;
                }
```




Set and get functions to manipulate private data member

Publically available data:
Assigning values from

Private data:
Accessing private data using member function

# EXERCISE

## Task 1

Create a class named "Lab" that has the following members:

- Int LabNumber
- Int NoOfTasks
- String Topic
- Float Attendance s

Create a function getData() to get the details from user input.
Create a function display() to print the data members.

## Task 2

In the main program

- Create 4 objects of the Lab Class.
- Call get Data function for each class.
- Print the details of the only those classes who have attendance higher than 0.75 (i.e 75%).

## Task 3

Create a class Circle with attributes radius and pie, which defaults to 2 and 3.14 respectively. Provide member functions that calculate the circumference and the area of the circle. Also, provide set and get functions for the radius attribute. The set function should verify that radius is between 0 and 15.

## Task 4

Create a class called Student that includes three pieces of information as **private** data members—a first name (type char*), a last name (type string) and CGPA (type float). Your class should have 3 setter (mutator) functions that initializes the three data members. Provide a getter (accessor) function for each data member. If the CGPA is not positive, set it to 0. Write a test program that demonstrates class Student's capabilities. Create two Student objects and display each object's CGPA. Then give each Student a 0.25 point CGPA raise in the next semester and check if he is in warning (CGPA < 2.5) or not after the next semester. Identify and add any other related functions to achieve the said goal.

## Task 5

Create a class called Product that a superstore might use to represent a product purchased from the store. A Product should include four **private** data members—a Name (type string), a Type (type string), a quantity of the product being purchased (type int) and a price per item (type float). Your class should have four mutator functions that initializes the four data members. Provide accessor functions for getting the value of each data member. In addition, provide a member function named getProductAmount that calculates the amount (i.e., multiplies the quantity by the price per item), then returns the amount as a float value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0. In the main, calculate the sum of all products amount to create a final bill. Write a test program that demonstrates the class capabilities.

# SUBMISSION GUIDELINES:

1. Complete your tasks.
2. Make sure each task is compiling and running successfully.
3. After execution, copy the code of every task in a single text file.
4. Submit the text file on Google Classroom

Your .txt file must look like this

```
//Task 1

#include<iostream>
using namespace std;

int main()
{
-----
}

//Task 2

#include<iostream>
using namespace std;

int main()
{
-----
}

//Task 3

#include<iostream>
using namespace std;

int main()
{
-----
}
```