



Course Code: CL-1004	Course : Object Oriented Programming Lab
-----------------------------	---

Lab # 03

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.

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

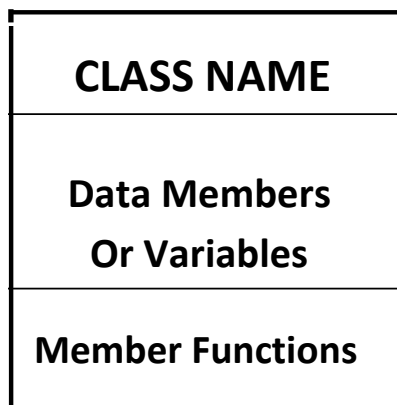
As, many houses can be made from the same description, we can create many objects from a class.

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.

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.

A class member can be defined as public, private or protected. By default members would be assumed as private.

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.

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.

Syntax to Define Object in C++

```
className objectVariableName;
```

RADIO r;

CAR c;

Accessing Public Data Members

The public data members of objects of a class can be accessed using the direct member access operator (.).

However the private data members are not allowed to be accessed directly by the object. Accessing a data member depends solely on the access control of that data member.

Accessing Private Data Members

To access, use and initialize the private data member you need to create getter and setter functions, to get and set the value of the data member.

The setter function will set the value passed as argument to the private data member, and the getter function will return the value of the private data member to be used. Both getter and setter function must be defined public.

```

C++ > oop.cpp > main()
1  #include<iostream>
2  #include <iomanip>
3  using namespace std;
4
5  class Student
6  {
7      private:    // private data member
8          int rollno;
9
10     public:
11         // public function to get value of rollno - getter
12         int getRollno()
13         {
14             return rollno;
15         }
16         // public function to set value for rollno - setter
17         void setRollno(int i)
18         {
19             rollno=i;
20         }
21 };
22
23 int main()
24 {
25     Student A;
26     A.rollno=1; //Compile time error
27     cout<< A.rollno; //Compile time error
28
29     A.setRollno(1); //Rollno initialized to 1
30     cout<< A.getRollno(); //Output will be 1
31 }
32

```

Getter & Setter , this Keyword in C++ Programming

The getter function is used to retrieve the variable value and the setter function is used to set the variable value. They this is a keyword that refers to the current instance of the class. They are getters and setters the standard way to provide access to data in Java classes. Setters and Getters allow for an object to contain private variables which can be accessed and changed with restrictions.

CL1004 - Object Oriented Programming Lab 2023

```
#include<iostream>

using namespace std;

class student{
private:
    string name;
    int age;
public:
    student(string n,int a) {
        this->setName(n);
        this->setAge(a);
    }
    string getName() {
        return this->name;
    }
    void setName(string n) {
        this->name=n;
    }
    int getAge() {
        return this->age;
    }
    void setAge(int a) {
        this->age=a;
    }
    void printDetails(){
        cout<<"Name : "<<name<<endl;
        cout<<"Age : "<<age<<endl;
    }

int main(){
    student o("Ali",25);
    o.printDetails();
    o.setName("M.Ali");
    o.printDetails();
    cout<<o.getName()<<endl;
    return 0;}
```

Output

Name : Ali

Age : 25

Name : M.Ali

Age : 25

M.Ali

Member Functions in Classes

There are 2 ways to define a member function:

- Inside class definition
- Outside class definition

1. Inside class definition

With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function.

Note that all the member functions defined inside the class definition are by default **inline**, but you can also make any non-class function inline by using keyword inline with them. Inline functions are actual functions, which are copied everywhere during compilation, like pre-processor macro, so the overhead of function calling is reduced.

2. Outside class definition

To define a member function outside the class definition we have to use the scope resolution :: operator along with class name and function name.

```
C++ > oop.cpp > main()
1  #include <iostream>
2  using namespace std;
3  class Student
4  {
5      public:
6      string StudentName;
7      int id;
8
9      // printname is not defined inside class definition
10     void printname();
11
12     // printid is defined inside class definition
13     void printid()
14     {
15         cout << " Student id is: " << id;
16     }
17 };
18
19 // Definition of printname using scope resolution operator ::
20 void Student::printname()
21 {
22     cout << " Student name is: " << StudentName;
23 }
24 int main()
25 {
26     Student obj1;
27     obj1.StudentName = "xyz";
28     obj1.id=15;
29
30     // call printname()
31     obj1.printname();
32     cout << endl;
33
34     // call printid()
35     obj1.printid();
36     return 0;
37 }
```

```
C:\Users\Administrator\Documents\C++>cd "c:\Users\Administrator\Documents\C++\" && g++ oop.cpp -o oop && "c:\Users\Administrator\Documents\C++\"
Student name is: xyz
Student id is: 15
```

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

```
#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<<"Enter height in Inches ";
    cin>>height;
    bmi=calculateBMI(weight,height);
    cout<<"Your BMI is "<<bmi<<" Your status is "<<findStatus(bmi);
}
```

Procedural Approach

```
#include<iostream>
using namespace std;
class BMI
{
    double weight, height,bmi;
    string status;
public:
    void getInput() {
        cout<<"Enter weight in Pounds ";
        cin>>weight;
        cout<<"Enter 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);
```



Set and get functions to manipulate
private data member

Publically available data:
Assigning values from

Private data:

```
cout<<endl;  
currentAccount.displayDetails();  
return 0;  
}
```

Accessing private data using member function

Exercise Lab 03

Question # 01:

Imagine you are tasked with developing a real estate management system in C++. Design a class named **RealEstateProperty** to represent a real estate property. The class should encapsulate attributes such as the property's address, price, square footage, and a member function to display property details. Add one more function property sold out.

Question # 02:

You are developing a hotel management system in C++. Design a class named **HotelService** to represent a service provided by the hotel. The class should encapsulate attributes such as the service name, cost, and a member function to display service details. Implement a constructor to initialize these attributes.

Question # 03:

Imagine you are tasked with designing a C++ class for a character in the "Injustice: Gods Among Us" game. Assume each character has attributes such as name, health, power level, and special abilities. Design a C++ class called **InjusticeCharacter** that encapsulates these attributes.

Question # 04:

Create a class called Smartphone with the following attributes:

- Company
- Model
- Display Resolution
- RAM
- ROM
- Storage

Create getter and setter methods for your attributes. A smartphone has some specific actions that it can perform. For example:

1. Make a phone call
2. Send a message
3. Connect to the wifi
4. Browse the internet

Create different smartphone objects. Set their attributes using the setter functions, and display their attributes after using the getter functions to fetch the attributes.

Question # 05:

You are tasked with creating a simplified program to model a ride-sharing service, similar to Uber, in C++. Design a class named **UberRide** to represent a ride in the service. The class should encapsulate attributes such as the rider's name, driver's name, distance, and fare. Implement a constructor to initialize these attributes. Put some Karachi e.g. Saddar to Gulistan block 13. Finally, based on that distance calculate the fare.