Unit 1: Introduction to Object Oriented Programming

By : Prof. Abhijit Pande

Syllabus: Introduction to object oriented programming paradigm, procedure oriented programming vs OOP, features of OOP,benefits of OOP, defining class, instantiating a class. UML diagrams to represent class, objects and variousrelationships.

COURSE CODE: CT2202

COURSE NAME: OBJECT ORIENTED PROGRAMMING

COURSE OBJECTIVE

1. To introduce object oriented programming features and its diagrammatic representation of its model components.
2. To understand concept of class, handling its features and the reusability concept in object oriented language.
3. To understand the mechanism to make use of files and standard libraries.
4. To introduce the exception handling mechanism and the MVC architecture along with web components to design the software solution.
5. To introduce how to perform the event driven programming.

COURSE OUTCOME

1. Able to analyze the problem and can proposed the solution in OO approach.
2. Able to implement the solution using suitable reusability technique provided in OOP language.
3. Able to implement the solution using files and standard template library.
4. Able to design the error free software solution using the standard architecture patterns.
5. Able to design and implement the event driven solution for the problem.

Difference between Procedure Oriented Programming and Object Oriented Programming

Procedure Oriented Programming	***Object Oriented Programming***
In POP, program is divided into small parts called **functions**.	In OOP, program is divided into parts called **objects**.
In POP,Importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world**.
POP follows **Top Down approach**.	OOP follows **Bottom Up approach**.
POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data cannot move easily from function to function, it can be kept public or private so we can control the access of data.
Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.

<u>Introduction to Object Oriented Programming</u>

- **Object-Oriented Programming (OOP)** is the term used to describe a programming approach based on **objects** and **classes**.
- The object-oriented paradigm allows us to organise software as a collection of objects that consist of both data and behaviour.
- Object Oriented Programming (OOP) is a programming model where programs are organized around objects and data rather than action and logic.
- OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects.

1. The software is divided into a number of small units called objects. The data and functions are built around these objects.
2. The data of the objects can be accessed only by the functions associated with that object.
3. The functions of one object can access the functions of another object.

Application of OOP

Main application areas of OOP are

- User interface design such as windows, menu,…
- Real Time Systems
- Simulation and Modelling
- Object oriented databases
- AI and Expert System
- Neural Networks and parallel programming
- Decision support and office automation system

Benefits of OOP

- It is easy to model a real system as real objects are represented by programming objects in OOP. The objects are processed by their member data and functions. It is easy to analyze the user requirements.
- With the help of inheritance, we can reuse the existing class to derive a new class such that the redundant code is eliminated and the use of existing class is extended. This saves time and cost of program.
- In OOP, data can be made private to a class such that only member functions of the class can access the data. This principle of data hiding helps the programmer to build a secure program that cannot be invaded by code in other part of the program.

- With the help of polymorphism, the same function or same operator can be used for different purposes. This helps to manage software complexity easily.
- Large problems can be reduced to smaller and more manageable problems. It is easy to partition the work in a project based on objects.
- It is possible to have multiple instances of an object to co-exist without any interference i.e. each object has its own separate member data and function.

Features of OOP

The Objects Oriented programming language supports all the features of normal programming languages. In addition it supports some important concepts such as

- Objects
- Classes
- Data abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

1. Objects

- Objects are the basic run time entities in an object oriented system.
- They may represent a person, a place, a bank account or any item that the program has to handle.
- When a program is executed, the objects interact by sending messages to one another.
- Syntax:

  Classname objectname;
- Example :   *Student S;*

2. Classes

- Class is a user defined data type and it behaves like a built in data type.
- Class is a collection of data members and member functions.
- Syntax:

```
class classname
{
private:
Data members;
public:
Menber functions;
};
```

3. Data abstraction and Encapsulation

- The wrapping up of data and function into a single unit (class) is known as *encapsulation.*
- The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.
- *Abstraction* refers to the act of representing essential features without including the background details or explanations.
- Classes use the concept of abstraction and are defined as a list of abstract attributes and functions to operate on these attributes.

4. Inheritance

- Inheritance is the process by which objects of one class acquires the properties of objects of another class.
- In other words we may have common features or characteristics that may be needed by number of classes.

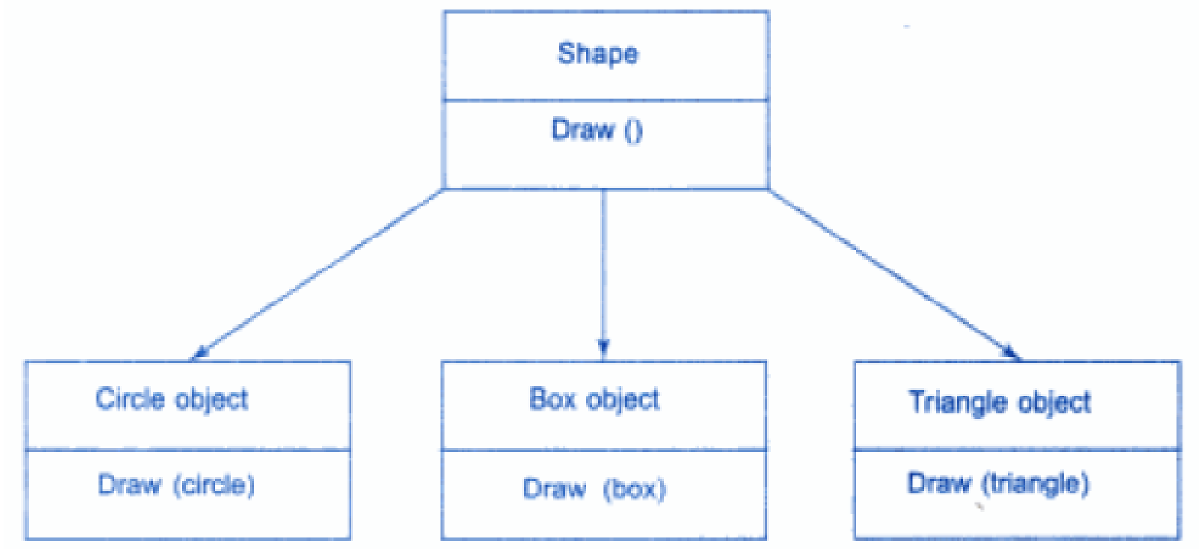
- So those features can be placed in a common tree class called base class and the other classes which have these characteristics can take the tree class and define only the new things that they have on their own in their classes. These classes are called derived class.
- The main advantage of using this concept of inheritance in Object oriented programming is it helps in reducing the code size since the common characteristic is placed separately called as base class and it is just referred in the derived class. This provide the users the important usage of terminology called as reusability.

5. <u>Polymorphism</u>

- Polymorphism means an ability to assume different forms at different places.
- In OOP, it is a language's ability to handle objects differently based on their runtime type and use. Polymorphism is briefly described as "one interface, many implementations".
- Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form.
- There are two types of polymorphism.

1. *Compile time polymorphism* - It is achieved by overloading functions and operators

2. *Run time polymorphism* - It is achieved by overriding virtual functions

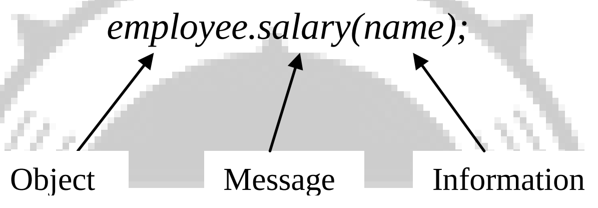


6. Dynamic Binding

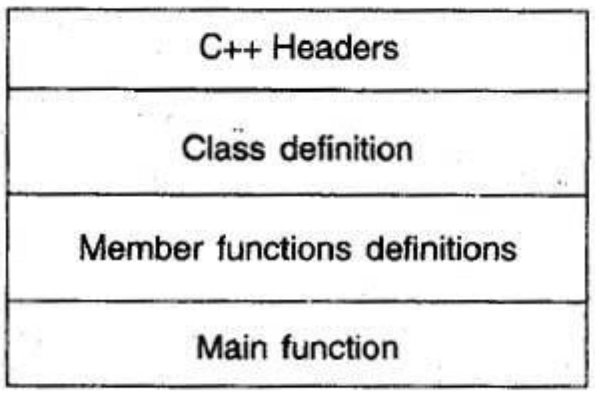
- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time.

7. Message Passing

- An object oriented program consists of a set of objects that communicate with each other.
- Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.
- Example:



Structure of C++ program



Structure of a C++ Program

C++ Headers

- In this section we need to include all the header files which we want to use.
- Example: iostream.h, conio.h

Class definition

- In this section we need to declare class with their data members and member functions.

Member function definition

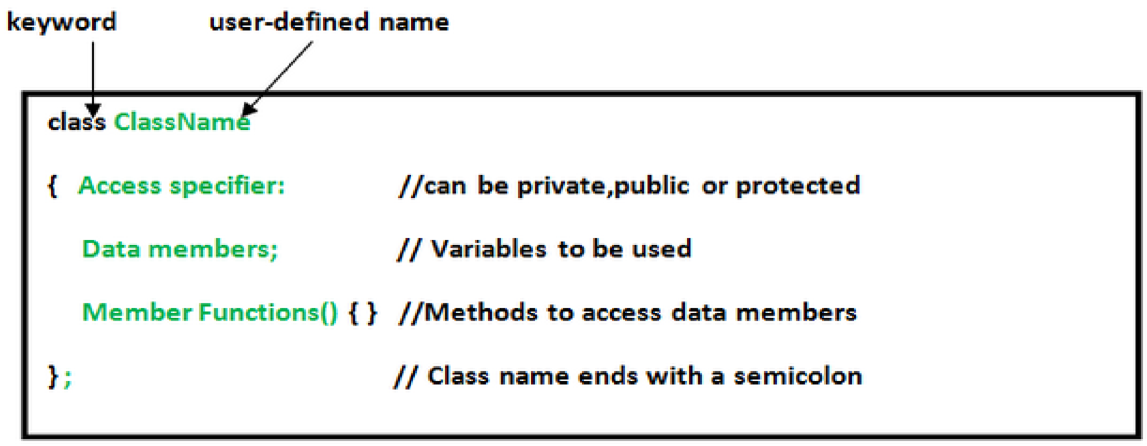
- Once we are declaring the member function inside the class, we need to define them.
- We can define them inside the class and outside the class.

Main function

- In main function we can create the objects of already declared class.
- With that object , we can call the member functions of the class and with the member functions we can call or access the data members of the class.

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 define the properties and behaviour of the objects in a Class.



```
class ClassName
{  Access specifier:          //can be private,public or protected
   Data members;              // Variables to be used
   Member Functions() { }     //Methods to access data members
};                            // Class name ends with a semicolon
```

- Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

- In the above example of class Car, the data member will be speed limit, mileage etc and member functions can apply brakes, increase speed etc.

Declaring Objects

- When a class is defined, only the specification for the object is defined; no memory or storage is allocated.
- To use the data and access functions defined in the class, you need to create objects.
- Syntax:

ClassName ObjectName;

Accessing data members and member functions

- The data members and member functions of class can be accessed using the dot('.') operator with the object.
- For example if the name of object is *obj* and you want to access the member function with the name *printName()* then you will have to write *obj.printName()* .

Accessing Data Members

- The public data members are also accessed in the same way given 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. This access control is given by Access modifiers in C++.
- There are three access modifiers : ***public, private and protected***.

Class access modifiers

- Data hiding is one of the important features of Object Oriented Programming which allows preventing the functions of a program to access directly the internal representation of a class type.
- The access restriction to the class members is specified by the labeled **public, private,** and **protected** sections within the class body.
- The keywords public, private, and protected are called access specifiers.
- The default access for members and classes is **private.**

1. The public members

- A **public** member is accessible from anywhere outside the class but within a program.
- You can set and get the value of public variables without any member function

2. The private members

- A **private** member variable or function cannot be accessed, or even viewed from outside the class.
- Only the class and friend functions can access private members.
- By default all the members of a class would be private
- Practically, we define data in private section and related functions in public section so that they can be called from outside of the class.

3. The protected members:

- A **protected** member variable or function is very similar to a private member but it provided one additional benefit that they can be accessed in child classes which are called derived classes.

Class member functions

- A member function of a class is a function that has its definition or its prototype within the class definition like any other variable.
- It operates on any object of the class of which it is a member, and has access to all the members of a class for that object.
- Member functions can be defined within the class definition or separately using **scope resolution operator, ::**.
- Defining a member function within the class definition declares the function **inline**, even if you do not use the inline specifier.

```
class Box
{
public:
double length; // Length of a box
double breadth; // Breadth of a box
double height; // Height of a box
double getVolume(void)
{
return length * breadth * height;
}
};
```

- If you like you can define same function outside the class using **scope resolution operator, ::** as follows:

```
double Box::getVolume(void)
{
return length * breadth * height;
}
```

- Here, only important point is that you would have to use class name just before :: operator.

- A member function will be called using a dot operator (**.**) on a object where it will manipulate data related to that object only as follows:

```
Box myBox; // Create an object
myBox.getVolume(); // Call member function for the object
```

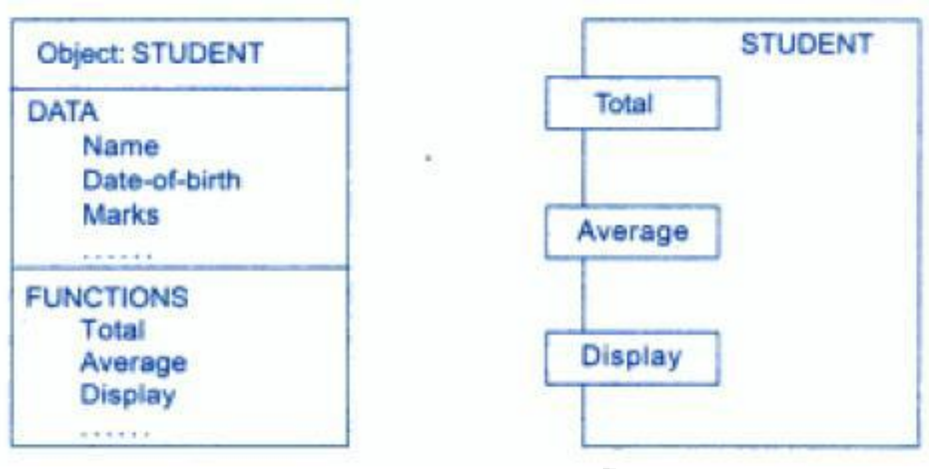
UML Representation of Class

A class notation consists of three parts:

1. **Class Name**
   - The name of the class appears in the first partition.
2. **Class Attributes**
   - Attributes are shown in the second partition.
   - The attribute type is shown after the colon.
   - Attributes map onto member variables (data members) in code.
3. **Class Operations** (Methods)
   - Operations are shown in the third partition. They are services the class provides.
   - The return type of a method is shown after the colon at the end of the method signature.
   - The return type of method parameters are shown after the colon following the parameter name.
   - Operations map onto class methods in code

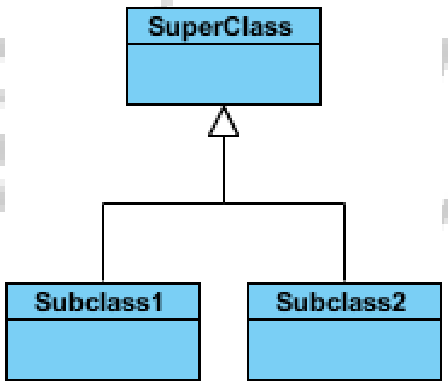
Class Name			
+	Attribute 1	:	int
-	Attribute 2	:	float
#	Attribute 3	:	char
+	Fun1( )	:	String
-	Fun2( )	:	int
#	Fun3( )	:	float

Object



Inheritance

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class


SuperClass
Subclass1
Subclass2