**Assignment(1)**

**Class:BCA-II(Sem-III)**

**Subject:Object Oriented Programming Using C++**

**Topics:**Characteristics of OOPS,Constructor and characteristics of constructor

**Submitted To:**

Ekta Garg

Prof.of Comp.Sci.Department

**Ques1:Characteristics of Object Oriented Programming ?**

## Ans:Characteristics of an Object-Oriented Programming Language



**Class**

The building block of C++ that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object. For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, the Car is the class, and wheels, speed limits, and mileage are their properties.

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

We can say that a **Class in C++** is a blueprint representing a group of objects which shares some common properties and behaviors.

**Object**

An Object is an identifiable entity with some characteristics and behavior. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Objects take up space in memory and have an associated address like a record in pascal or structure or union. When a program is executed the objects interact by sending messages to one another. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other’s data or code, it is sufficient to know the type of message accepted and the type of response returned by the objects.

**Encapsulation**

In normal terms, Encapsulation is defined as wrapping up data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them. Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section, etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.

Encapsulation also leads to *data abstraction or data hiding*. Using encapsulation also hides the data. In the above example, the data of any of the sections like sales, finance, or accounts are hidden from any other section.

**Abstraction**

Data abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car. The man only knows that pressing the accelerator will increase the speed of the car or applying brakes will stop the car but he does not know how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of an accelerator, brakes, etc. in the car. This is what abstraction is.

* ***Abstraction using Classes***: We can implement Abstraction in C++ using classes. The class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to the outside world and which is not.
* ***Abstraction in Header files***: One more type of abstraction in C++ can be header files. For example, consider the pow() method present in math.h header file. Whenever we need to calculate the power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating the power of numbers.

**Polymorphism**

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. A person at the same time can have different characteristics. A man at the same time is a father, a husband, and an employee. So the same person possesses different behavior in different situations. This is called polymorphism. An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation. C++ supports operator overloading and function overloading.

* ***Operator Overloading***: The process of making an operator exhibit different behaviors in different instances is known as operator overloading.
* ***Function Overloading***: Function overloading is using a single function name to perform different types of tasks. Polymorphism is extensively used in implementing inheritance.

**Example**: Suppose we have to write a function to add some integers, sometimes there are 2 integers, and sometimes there are 3 integers. We can write the Addition Method with the same name having different parameters, the concerned method will be called according to parameters.

**Inheritance**

The capability of a class to derive properties and characteristics from another class is called [Inheritance](https://www.geeksforgeeks.org/inheritance-in-c/). Inheritance is one of the most important features of Object-Oriented Programming.

* **Sub Class**: The class that inherits properties from another class is called Sub class or Derived Class.
* **Super Class**: The class whose properties are inherited by a sub-class is called Base Class or Superclass.
* **Reusability**: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

**Example**: Dog, Cat, Cow can be Derived Class of Animal Base Class.

**Dynamic Binding**

In dynamic binding, the code to be executed in response to the function call is decided at runtime. C++ has [virtual functions](https://www.geeksforgeeks.org/virtual-functions-and-runtime-polymorphism-in-c-set-1-introduction/) to support this. Because dynamic binding is flexible, it avoids the drawbacks of static binding, which connected the function call and definition at build time.

## ****Message Passing****

Objects communicate with one another by sending and receiving information. A message for an object is a request for the execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function, and the information to be sent.

**Ques2:What is Constructor and Characteristics Of constructor in C++?**

**Ans: Constructor in C++** is a special method that is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally. The constructor in C++ has the same name as the class or structure. Constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object which is why it is known as constructors.

• Constructor is a member function of a class, whose name is same the classname.  
• Constructor is a special type of member function that is used to initialize the data members for an object of a class automatically, when an object of the same class is created.  
• Constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.  
• Constructor do not return value, hence they do not have a return type.

### ****Characteristics of the constructor:****

* The name of the constructor is the same as its class name.
* Constructors are mostly declared in the public section of the class though it can be declared in the private section of the class.
* Constructors do not return values; hence they do not have a return type.
* A constructor gets called automatically when we create the object of the class.
* Constructors can be overloaded.
* Constructor can not be declared virtual.
* Constructor cannot be inherited.
* Addresses of Constructor cannot be referred.
* Constructor make implicit calls to **new**and **delete**operators during memory allocation.