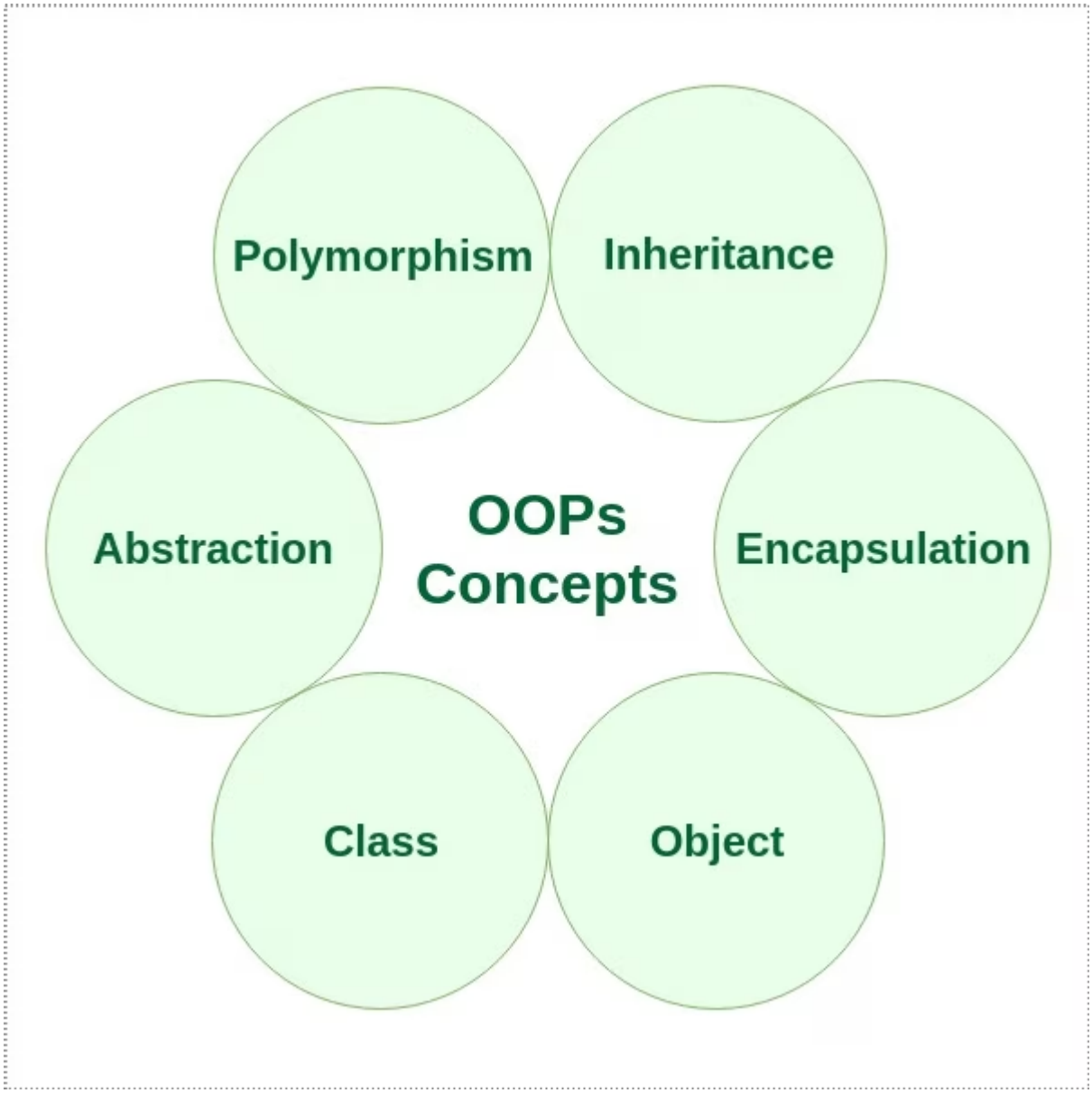
# Object Oriented Programming in C++

**TABLE OF CONTENT:**

1. Introduction
2. Class
3. Objects
4. Encapsulation
5. Abstraction
6. Polymorphism
7. Inheritance
8. Dynamic Binding
9. Message Passing

Object-oriented programming – As the name suggests uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

**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 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.

- 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.

- 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 blue-print representing a group of objects which shares some common properties and behaviours.



**Object:** An Object is an identifiable entity with some characteristics and behaviour. 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.

```
class person
{
    char name[20];
    int id;
public:
    void getdetails(){}
};

int main()
{
   person p1; // p1 is a object
}
```

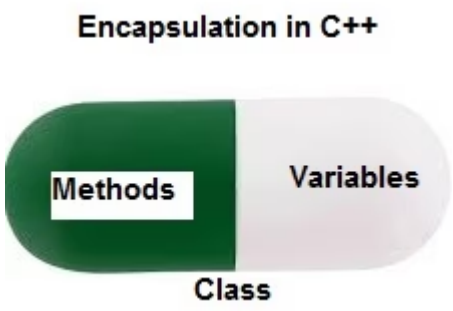
Object take up space in memory and have an associated address like a record in pascal or structure or union in C.

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 type of response returned by the objects.

**Encapsulation**: In normal terms, Encapsulation is defined as wrapping up of 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 in C++

Encapsulation also leads to *data abstraction or hiding*. As using encapsulation also hides the data. In the above example, the data of any of the section 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 accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of 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.

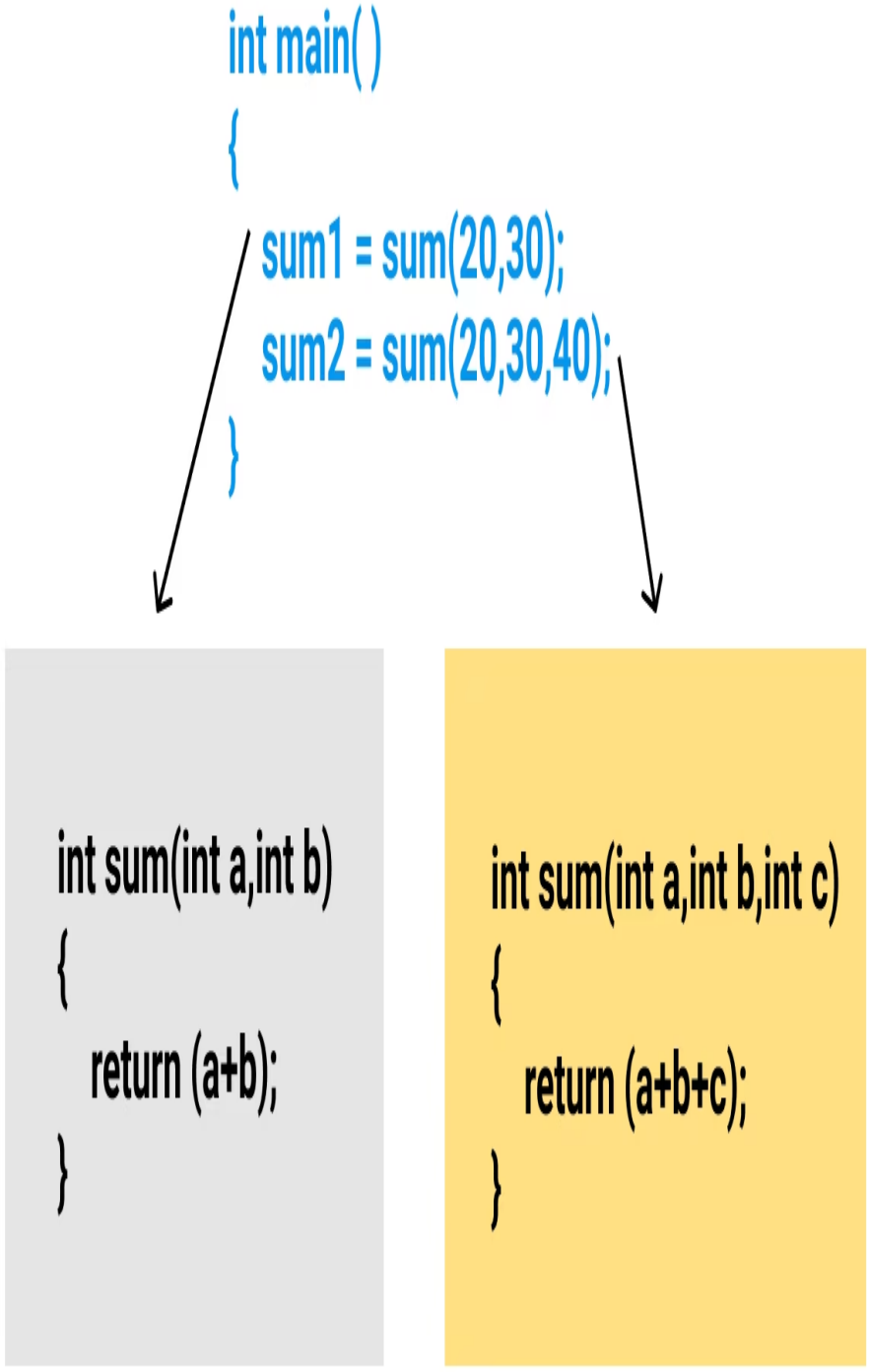
A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

An operation may exhibit different behaviours in different instances. The behaviour 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 to exhibit different behaviours 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.

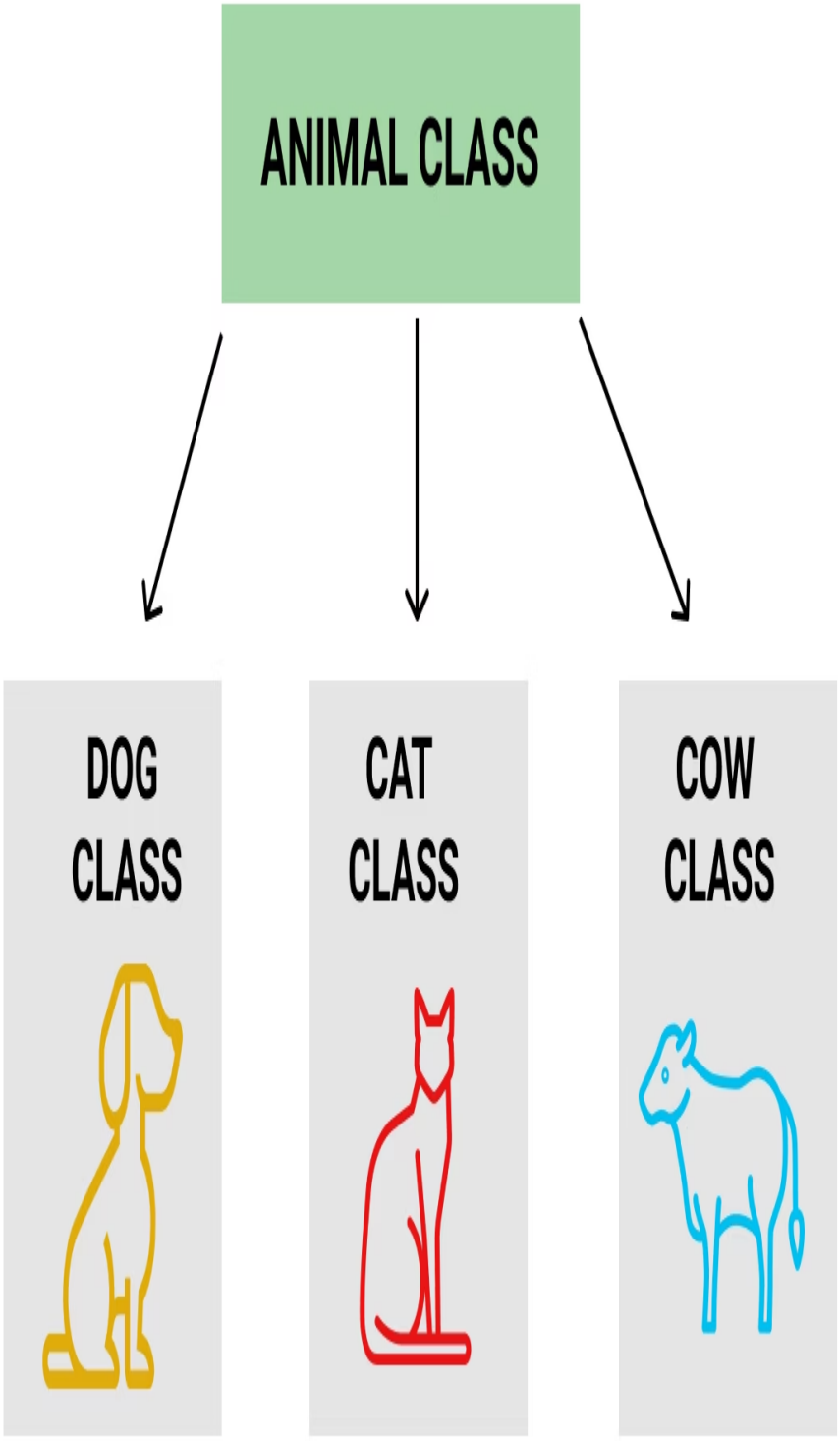
**Example**: Suppose we have to write a function to add some integers, some times there are 2 integers, some times there are 3 integers. We can write the Addition Method with the same name having different parameters, the concerned method will be called according to parameters.


int main( )
{
sum1 = sum(20,30);
sum2 = sum(20,30,40);
}
int sum(int a,int b)
{
return (a+b);
}
int sum(int a,int b,int c)
{
return (a+b+c);
}

**Inheritance**: The capability of a class to derive properties and characteristics from another class is called Inheritance. 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 sub class is called Base Class or Super class.
- **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 function call is decided at runtime. C++ has virtual functions to support this.

**Message Passing:** Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for 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.

**Related Articles**:

- Classes and Objects
- Inheritance
- Access Modifiers
- Abstraction

This article is contributed by **Vankayala Karunakar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

**Article Tags :** C++ School Programming CPP-Basics cpp-class cpp-inheritance

**Recommended Articles**

1. Object Oriented Programming in Python | Set 2 (Data Hiding and Object Printing)
2. Introduction of Object Oriented Programming
3. Why C++ is partially Object Oriented Language?
4. OOPs | Object Oriented Design
5. Can a C++ class have an object of self type?
6. Preventing Object Copy in C++ (3 Different Ways)
7. cerr - Standard Error Stream Object in C++
8. How to add reference of an object in Container Classes
9. Dynamic initialization of object in C++
10. Object Delegation in C++
11. Different ways to instantiate an object in C++ with Examples
12. Object Composition-Delegation in C++ with Examples
13. Object-Based Databases
14. Base class pointer pointing to derived class object

15. OOP in Python | Set 3 (Inheritance, examples of object, issubclass and super)
16. Exception Handling and Object Destruction in C++
17. How to Manipulate cout Object using C++ IOS Library?
18. Object Slicing in C++
19. Where is an object stored if it is created inside a block in C++?
20. getchar_unlocked() – Faster Input in C/C++ For Competitive Programming
21. Socket Programming in C/C++: Handling multiple clients on server without multi threading
22. Common mistakes to be avoided in Competitive Programming in C++ | Beginners
23. C++ Programming and STL Facts
24. Why is programming important for first year or school students?
25. CBSE Class 11 | Concepts of Programming Methodology

Read Full Article