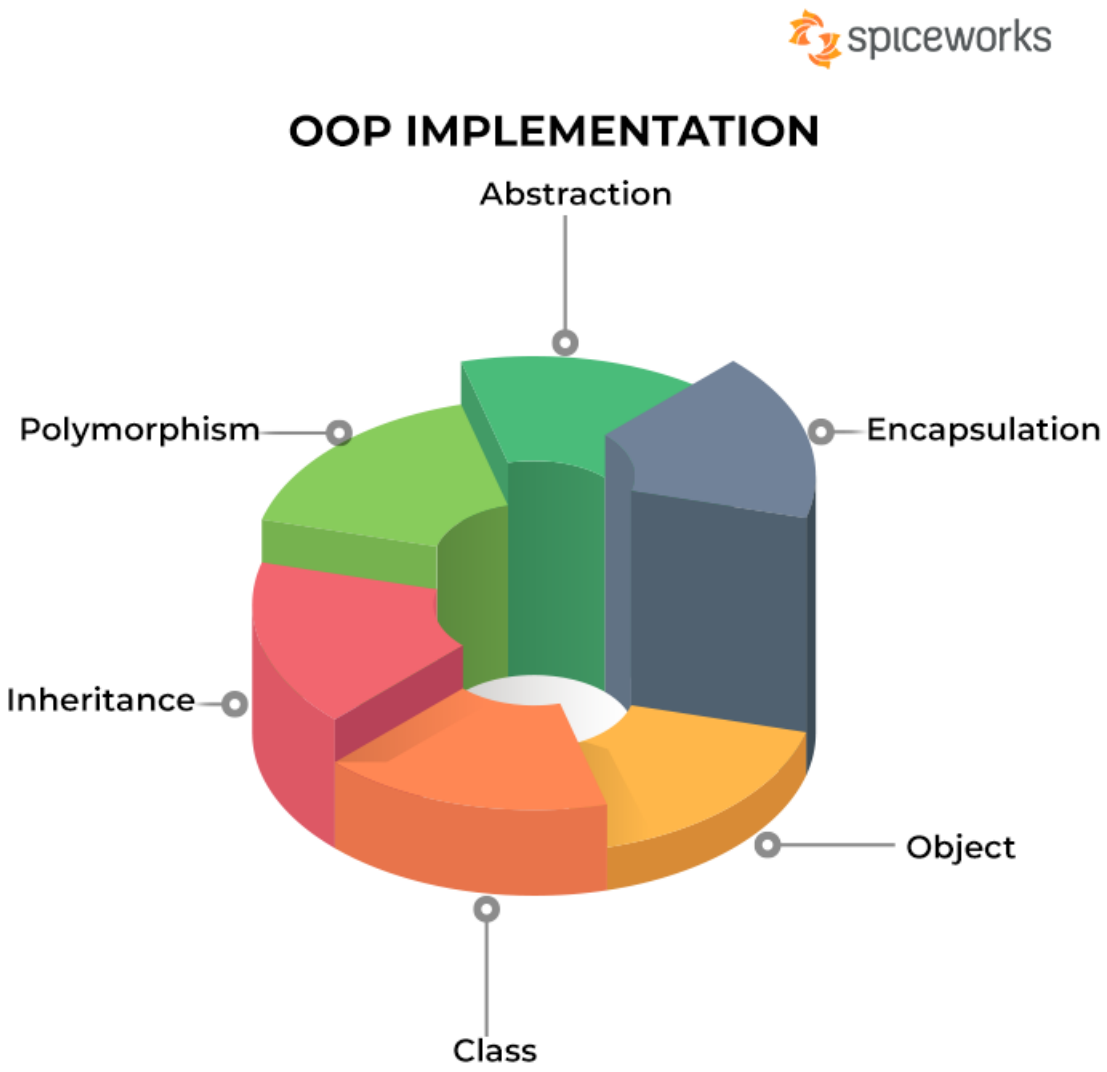
OOPS:- ***Object-oriented programming (OOP) is defined as a programming paradigm (and not a specific language) built on the concept of objects, i.e., a set of data contained in fields, and code, indicating procedures – instead of the usual logic-based system. This article explains the fundamental concepts of OOP and its most significant advantages.***

As the name suggests, Object-Oriented Programming or Java OOPs concept refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code. Objects are seen by the viewer or user, performing tasks you assign.
Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming. The main aim of OOPs 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.

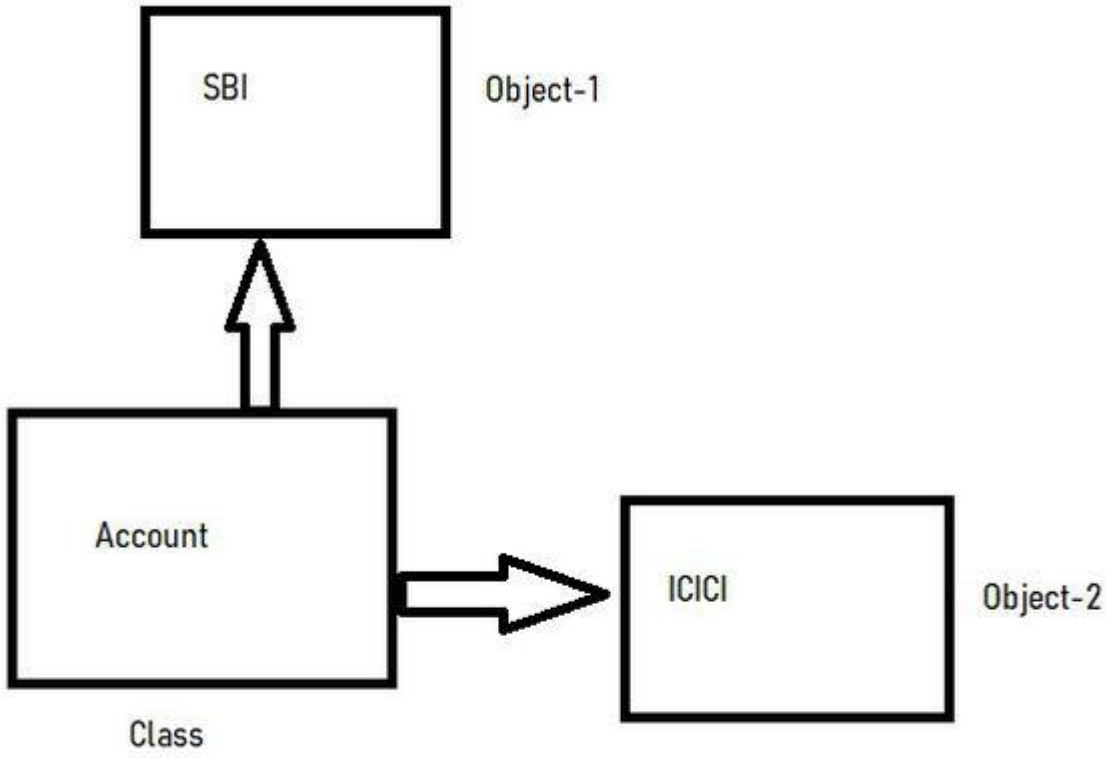
Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.



Difference Between Object And Class

- **Class** is a detailed description, the definition, and the template of what an object will be. But it is not the object itself. Also, what we call, a class is the building block that leads to Object-Oriented Programming. It is a user-defined data type, that holds its own data members and member functions, which can be accessed and used by creating an instance of that class. It is the blueprint of any object. Once we have written a class and defined it, we can use it to create as many objects based on that class as we want. In **Java**, the class contains fields, constructors, and methods. For example, consider the **Class** of **Accounts**. There may be *many accounts with different names and types*, but all of *them will share some common properties*, as all of them will have some *common attributes* like ***balance***, ***account holder name***, etc. So here, the **Account** is the **class**.

Object is an instance of a class. All data members and member functions of the class can be accessed with the help of objects. When a class is defined, no memory is allocated, but memory is allocated when it is instantiated (i.e. an object is created). For Example, considering the **objects** for the class **Account** are **SBI Account**, **ICICI account**, etc.



Difference between object and class

There are many differences between object and class. A list of differences between object and class are given below:

No.	Object	Class
1)	Object is an **instance** of a class.	Class is a **blueprint or template** from which objects are created.
2)	Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair	Class is a **group of similar**

	etc.	**objects**.
3)	Object is a **physical** entity.	Class is a **logical** entity.
4)	Object is created **many times** as per requirement.	Class is declared **once**.
5)	Object **allocates memory when it is created**.	Class **doesn't allocated memory when it is created**.

Implementation:-

C++ Classes/Objects

C++ is an object-oriented programming language.

Everything in C++ is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an **object**. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

Attributes and methods are basically **variables** and **functions** that belongs to the class. These are often referred to as "class members".

A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.

Example of Class and Objects

Class: Human **Object:** Man, Woman

Class: Fruit **Object:** Apple, Banana, Mango, Guava wtc.

Class: Mobile phone **Object:** iPhone, Samsung, Moto

Class: Food **Object:** Pizza, Burger, Samosa

Difference Between C and C++

Parameter	C	C++
Definition	It is a structural programming language that doesn't provide any support for classes and objects.	It is an object-oriented programming language, and it provides support for the concept of classes and objects.
History	Dennis Ritchie developed the C language at the AT&T Bell Laboratories in around 1969.	Bjarne Stroustrup developed the C++ language in 1979-1980 at Bell Labs.
Type of Programming Language	C primarily supports procedural programming for developing codes. Here, it checks the code line by line.	C++ supports both programming paradigms- procedural as well as object-oriented. It is, thus, known as a hybrid language.
Support for OOPs Feature	C has no support for the OOPs concept. Thus, it does not support encapsulation, polymorphism, and inheritance.	The C++ language supports encapsulation, inheritance, and polymorphism because it is an object-oriented programming language.
Supported Features	C has no support for functions and operator overloading. It also does not have any namespace feature and functionality of reference variables.	C++, on the other hand, supports both of the functions and operator overloading. It also has the namespace feature and the functionality of reference variables.
Driven Type	The C is a function-driven language because it is procedural programming.	The C++ language, on the other hand, is object-driven because it is OOP (object-oriented programming).
Data Security	C is vulnerable to manipulation via outside code. It is because it does not support encapsulation- leading to its data behaving as a free entity.	C++, on the other hand, is a very secure language. It supports encapsulation of data in the form of objects- thus hiding the information and ensuring that one uses the structures and operators as intended.
Type of Subset	It is a subset of the C++ language. It cannot run the	It is a superset of the C language. It is capable of running 99% of the C

	codes used in C++.	language codes.
Segregation of Data and Functions	Since C is a procedural programming language, the data and functions stay separate in it.	In the case of C++, the data and functions stay encapsulated in an object's form.
Hiding Data and Information	C does not support the hiding of data and information.	C++ language hides the data through encapsulation. This process ensures that a user utilizes the structures as operators as intended.
Built-in Data Types	The C language does not support built-in data types.	The C++ language supports built-in data types.
Function Inside Structures	In the case of C, it does not define the functions inside structures.	In the case of C++, it uses functions inside a structure.
Reference Variables	It does not support any reference variables.	It supports reference variables.
Overloading of Functions	Function overloading allows a user to have more than one function with different parameters but the same name. The C language does not support it.	The C++ language supports function overloading.
Overriding of Functions	Function overriding provides the specific implementation to any function that is defined already in the base class. The C language does not support it.	The C++ language supports function overriding.
Header File	C language uses the **<stdio.h>** header file.	C++ language uses the **<iostream.h>** header file.
Namespace Features	The namespace feature groups various entities like objects, classes, and functions under a specific name. No namespace features are present in the C language.	The C++ language uses the namespace features to help avoid name collisions.
Virtual and Friend	The C language does not support virtual and friend	The C++ language supports virtual and

Functions	functions.	friend functions.
Primary Focus	C language focuses on the process or method instead of focusing on the data.	C++ language focuses on the data instead of focusing on the procedure or method.
Inheritance	The inheritance feature assists the child class in reusing the parent class's properties. The C language offers no support for inheritance.	The C++ language provides support for inheritance.
Allocation and Deallocation of Memory	The C language provides calloc() and malloc() for dynamic allocation of memory and free() for deallocation of memory.	The C++ language provides a new operator for the allocation of memory and a delete operator for the deallocation of memory.
Exception Handling	It does not provide any direct support for exceptional handling. C language requires the usage of functions that support exception handling.	It provides direct support for exceptional handling. The C++ language uses a try-catch block.
Access Modifiers	The structures in C have no access modifiers.	The structures in C++ do have access modifiers.
Type of Approach	C language follows a top-down approach. It functions to break down the main module into various tasks. Then it breaks these tasks into sub-tasks, and so on.	C++ language follows the bottom-up approach. It means that it first develops the lower-level modules and then moves on to the next-level modules.
Function for Input/Output	The C language uses the scanf() and printf() functions for the input and output operations.	In the C++ language, it uses the cin and cout for the input and output operations.

Create a Class in C++

To create a class, use the `class` keyword:

Example

Create a class called "`MyClass`":

```
class MyClass {        // The class
  public:              // Access specifier
    int myNum;         // Attribute (int variable)
    string myString;   // Attribute (string variable)
};
```

Example explained

- The `class` keyword is used to create a class called `MyClass`.
- The `public` keyword is an **access specifier**, which specifies that members (attributes and methods) of the class are accessible from outside the class. You will learn more about access specifiers later.
- Inside the class, there is an integer variable `myNum` and a string variable `myString`. When variables are declared within a class, they are called **attributes**.
- At last, end the class definition with a semicolon `;`.

Create an Object in C++

In C++, an object is created from a class. We have already created the class named `MyClass`, so now we can use this to create objects.

To create an object of `MyClass`, specify the class name, followed by the object name.

To access the class attributes (`myNum` and `myString`), use the dot syntax (`.`) on the object:

Example

Create an object called "`myObj`" and access the attributes:

```
class MyClass {        // The class
  public:              // Access specifier
    int myNum;         // Attribute (int variable)
    string myString;   // Attribute (string variable)
};

int main() {
  MyClass myObj;  // Create an object of MyClass
```

```
  // Access attributes and set values
  myObj.myNum = 15;
  myObj.myString = "Some text";

  // Print attribute values
  cout << myObj.myNum << "\n";
  cout << myObj.myString;
  return 0;
}
```

Try it Yourself »

Multiple Objects

You can create multiple objects of one class:

Example

```
// Create a Car class with some attributes
class Car {
  public:
    string brand;
    string model;
    int year;
};

int main() {
  // Create an object of Car
  Car carObj1;
  carObj1.brand = "BMW";
  carObj1.model = "X5";
  carObj1.year = 1999;

  // Create another object of Car
  Car carObj2;
  carObj2.brand = "Ford";
  carObj2.model = "Mustang";
  carObj2.year = 1969;

  // Print attribute values
  cout << carObj1.brand << " " << carObj1.model << " " <<
carObj1.year << "\n";
```

```
  cout << carObj2.brand << " " << carObj2.model << " " <<
carObj2.year << "\n";
  return 0;
}
```

In Java

Java Classes/Objects

Java is an object-oriented programming language.

Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class in Java

To create a class, use the keyword `class`:

Main.javaGet your own Java Server

Create a class named "`Main`" with a variable x:

```
public class Main {

  int x = 5;

}
```

Remember from the Java Syntax chapter that a class should always start with an uppercase first letter, and that the name of the java file should match the class name.

Create an Object in Java

In Java, an object is created from a class. We have already created the class named `Main`, so now we can use this to create objects.

To create an object of `Main`, specify the class name, followed by the object name, and use the keyword `new`:

Example

Create an object called "`myObj`" and print the value of x:

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Python

Python Classes/Objects

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class in Python

To create a class, use the keyword `class`:

Example[Get your own Python Server]

Create a class named MyClass, with a property named x:

```
class MyClass:
  x = 5
```

Try it Yourself »

Create Object

Now we can use the class named MyClass to create objects:

Example

Create an object named p1, and print the value of x:

```
p1 = MyClass()
print(p1.x)
```