## What is OOP? Difference between class and objects?

OOP stands for Object Oriented Programming.

Object-oriented programming (OOP) is a programming paradigm that uses objects to model real-world entities and behaviors. Objects have data (attributes) and code (methods) that can interact with other objects. OOP makes it easier to create reusable, modular, and maintainable code.

Object-oriented programming (OOP) in Java is a programming methodology or paradigm (model) to design a computer program using classes and objects. It is an extension of procedural programming. Java OOP concept uses variables and methods as procedural programs do, but it focuses on the objects that contain variables and methods.

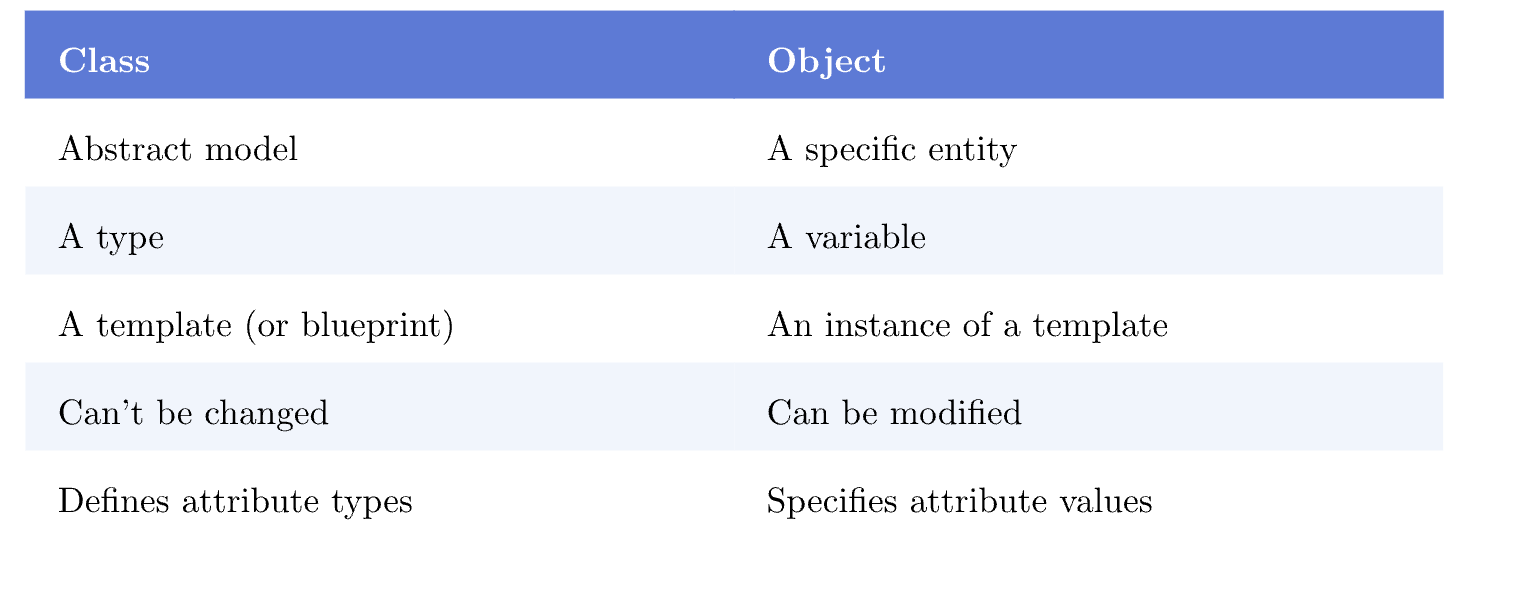
That’s why this new approach is called object-oriented approach in which we write programs using classes and objects. It comes with four main features like encapsulation, abstraction, inheritance, and polymorphism.

Some of the main concepts of OOP are:

* Class: A class is a blueprint or template for creating objects. It defines the common attributes and methods of a group of objects. For example, a class can be Animal, and it can have attributes like name, color, and methods like eat, sleep, etc.
* Object: An object is an instance or realization of a class. It has its own values for the attributes and can use the methods defined by the class. For example, an object can be Dog, and it can have name = “Spot”, color = “brown”, and use the methods eat, sleep, etc.
* Inheritance: Inheritance is a mechanism that allows a class to inherit the attributes and methods of another class. The class that inherits is called the subclass or child class, and the class that is inherited from is called the superclass or parent class. For example, a class can be Mammal, and it can inherit from Animal. Then, Mammal can have additional attributes and methods like hair, milk, etc.
* Polymorphism: Polymorphism is the ability of an object to take different forms depending on the context. It means that the same name can refer to different methods or behaviors in different classes or objects. For example, a method can be speak, and it can have different implementations for different classes or objects. A Dog can speak by barking, a Cat can speak by meowing, etc.
* Abstraction: Abstraction is the process of hiding the unnecessary or complex details and showing only the essential or relevant features of an object or a problem. It helps to reduce complexity and increase efficiency. For example, a class can be Car, and it can have attributes like model, color, and methods like start, stop, etc. The user does not need to know how the engine, brakes, or gears work internally, but only how to use the methods.
* Encapsulation: Encapsulation is the technique of wrapping the data and the code that operates on the data together in a single unit. It helps to protect the data from being accessed or modified by other parts of the program. It also makes the code more readable and maintainable. For example, a class can be BankAccount, and it can have attributes like balance, accountNumber, and methods like deposit, withdraw, etc. The user cannot directly access or change the balance, but only through the methods.

**Difference Between Class And Object:**

There are many differences between object and class. Some differences between object and class are given below:



| **Class** | **Object** |
| --- | --- |
| Class is used as a template for declaring and  creating the objects. | An object is an instance of a class. |
| When a class is created, no memory is allocated. | Objects are allocated memory space whenever they are created. |
| The class has to be declared first and only once. | An object is created many times as per requirement. |
| A class can not be manipulated as they are not available in the memory. | Objects can be manipulated. |
| A class is a logical entity. | An object is a physical entity. |
| It is declared with the class keyword | It is created with a class name in C++ and  with the **new** keywords in Java. |
| Class does not contain any values which  can be associated with the field. | Each object has its own values, which are associated with it. |
| A class is used to bind data as well as methods together as a single unit. | Objects are like a variable of the class. |
| **Syntax:** Declaring Class in C++ is as follows:  class <classname> {}; | **Syntax:** Instantiating an object for a Class in C++ is as follows:  class Student {     public:        void put(){            cout<<“Function Called”<<endl;        }  };   // The class is declared here  int main(){           Student s1;   // Object created           s1.put();  } |
| **Example:** Bike | **Example:** Ducati, Suzuki, Kawasaki |

## What are main features of OOP?

1. Higher priority is focused on data rather than functions.

2. Programs are divided into the number of entities known as objects.

3. Objects communicate with each other through functions (methods).

4. Methods that operate on Data of an object are closely bound together in the data structure.

5. An object is a group of data and methods.

6. Data is hidden in the OOP and cannot be accessed by external methods. Hence, it is safe from accidental modification.

7. Data cannot move freely out of the object.

8. New data and methods can be easily added whenever needs.

9. A bottom-up approach is adopted in programming design.

## What are advantages and disadvantages of OOP?

## **List of some advantages of Oops**

1. **Simplicity:** Oops is very easy to learn and very easy to use. The model of the software objects now becomes real-world objects, so the complexity is now reduced. As well as the program structure of oops is also very clear.
2. **Modularity:** Modularity provides to break a large structure into small chunks which make a program to understand very clearly. In oops each object forms a separate entity that works independently and during execution all combined together.
3. **Extensibility:** Using oops it is also very easy to add new features without changing the large operating environments. Just changing small parts or by introducing a few new objects and modifying some existing ones we can add new functionality and feature.
4. **Maintainability:** In Oops, it is also very easy to maintain objects separately. Using the object we can find the changing location easily and can fix them.
5. **Re-usability:** In Oops, we can reuse objects in different programs.
6. **Redundancy:**Opps have one feature known as Inheritance. It saves you from data redundancy. Suppose if you want the same functionality in multiple classes, you can take a class for the same functionality that you want in multiple classes. Now using the concept of inheritance using extends keyword just inherits the main class in a subclass.
7. **Data hiding:** Encapsulation provides data hidden feature. It protects data of one module from the other modules.
8. **Security:**Using data hiding and abstraction we are providing necessary data only it means we are maintaining security.

### **Disadvantage of the Object-Oriented Programming**

* The communication system between the objects happens in a complex system so it is difficult to implement.
* Basically oops involve more lines of code in comparison to procedural programs. Also, oop requires more memory to process at a great speed.

## What is Access specifier? What is their significance in OOP?

**Access modifiers** (or **access specifiers**) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components

Encapsulation means hiding the internal details of an object from the outside world and providing a public interface to interact with the object. Access specifiers help to achieve encapsulation by restricting the access to the internal data and behavior of an object.

Their significance in OOP is that they help to achieve one of the main principles of object-oriented programming, which is **encapsulation**. Encapsulation means hiding the internal details of an object from the outside world and providing a public interface to interact with the object. By using access specifiers, we can control the visibility and accessibility of the data and behavior of an object, and prevent unauthorized access or modification. This way, we can ensure the integrity and security of our object, and also make it easier to maintain and reuse.