OOP's Concepts :

OOPs (Object-Oriented Programming System)

Object means a real-world entity such as a pen, chair, table, computer, watch, etc.
Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

What are the benefits of Object-Oriented Programming?

1. Improved productivity during software development
2. Improved software maintainability
3. Faster development sprints
4. Lower cost of development
5. Higher quality software

*** **Smalltalk** is considered the first truly object-oriented programming language.

Object :

Any entity that has a state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Class

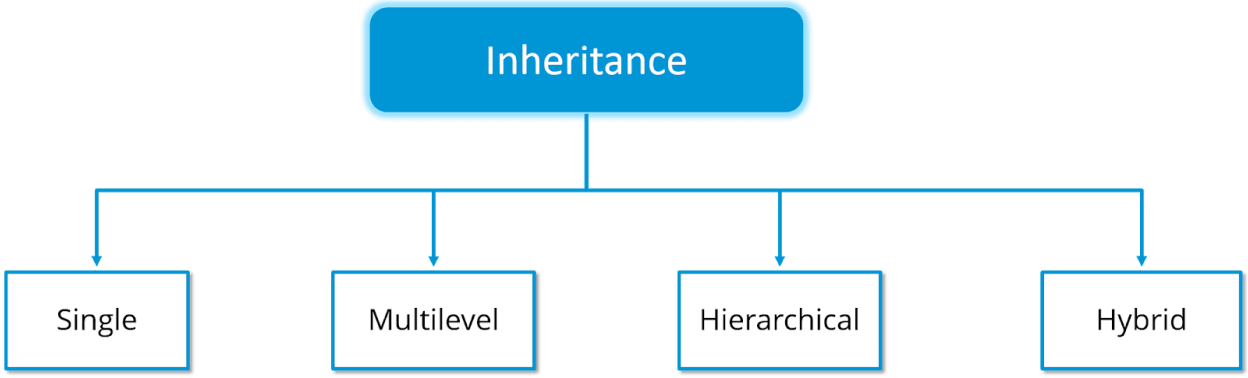
The collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

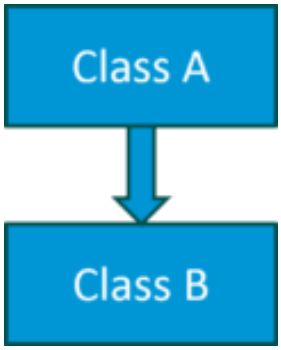
When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Inheritance is further classified into 4 types:



So let’s begin with the first type of inheritance i.e. Single Inheritance:

1. **Single Inheritance:**



In single inheritance, one class inherits the properties of another. It enables a derived class to inherit the properties and behavior of a single parent class. This will in turn enable code reusability as well as add new features to the existing code.

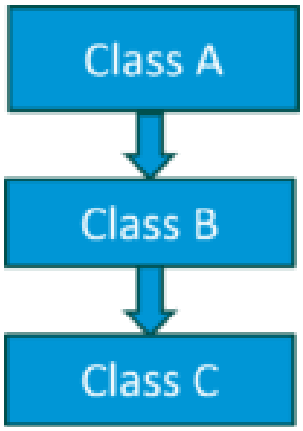
Here, Class A is your parent class and Class B is your child class which inherits the properties and behavior of the parent class.

Let’s see the syntax for single inheritance:

Explore Curriculum

```
1   Class A
2   {
3   ---
4   }
5   Class B extends A {
6   ---
7   }
```

2. Multilevel Inheritance:



When a class is derived from a class that is also derived from another class, i.e. a class having more than one parent class but at different levels, such type of inheritance is called Multilevel Inheritance.

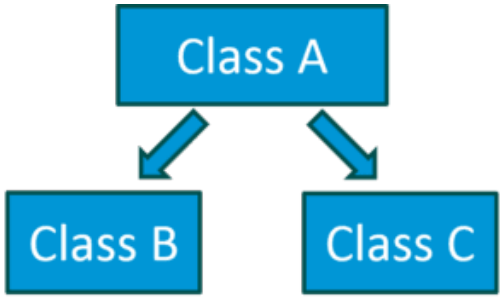
If we talk about the flowchart, class B inherits the properties and behavior of class A and class C inherits the properties of class B. Here A is the parent class for B and class B is

the parent class for C. So in this case class C implicitly inherits the properties and methods of class A along with Class B. That's what is multilevel inheritance.

Let's see the syntax for multilevel inheritance in Java:

```
Class A{
---
}
Class B extends A{
---
}
Class C extends B{
---
}
```

3. Hierarchical Inheritance:



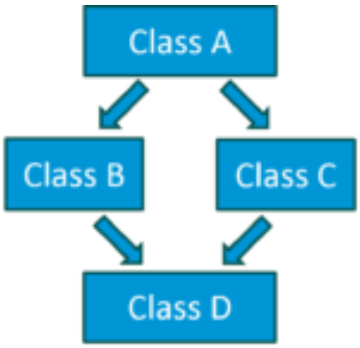
When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.

If we talk about the flowchart, Class B and C are the child classes which are inheriting from the parent class i.e Class A.

Let's see the syntax for hierarchical inheritance in Java:

```
1   Class A{
2   ---
3   }
4   Class B extends A{
5   ---
6   }
7   Class C extends A{
8   ---}
```

4. Hybrid Inheritance:



Hybrid inheritance is a combination of *multiple* inheritance and *multilevel* inheritance. Since multiple inheritances is not supported in Java as it leads to ambiguity, so this type of inheritance can only be achieved through the use of the interfaces.

If we talk about the flowchart, class A is a parent class for class B and C, whereas Class B and C are the parent class of D which is the only child class of B and C.

Polymorphism:

If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, a dog barks woof, etc.

Abstraction:

Hiding internal details and showing functionality is known as abstraction. For example phone calls, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

Encapsulation:

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule is wrapped with different medicines.

A java class is an example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.