What is a class - A class can be considered as a blueprint using which you can create as many objects as you like

Inheritance and it's types - Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

* **Super Class:**The class whose features are inherited is known as super class(or a base class or a parent class).
* **Sub Class:** The class that inherits the other class is known as sub class(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
* Single Inheritance – Single Inheritance is the simple inheritance of all, When a class extends another class(Only one class) then we  call it as **Single inheritance. Class A ->Class B.**

Eg :-

public class ClassA

{

public void dispA()

{

System.out.println("disp() method of ClassA");

}

}

public class ClassB extends ClassA

{

public void dispB()

{

System.out.println("disp() method of ClassB");

}

public static void main(String args[])

{

//Assigning ClassB object to ClassB reference

ClassB b = new ClassB();

//call dispA() method of ClassA

b.dispA();

//call dispB() method of ClassB

b.dispB();

}

}

* Multiple Inheritance **(Through**[**Interface**](https://javainterviewpoint.com/interface-java/)**) –**  **Multiple Inheritance** is nothing but **one** class **extending** **more** **than** one class. Multiple Inheritance is basically not supported by many **Object-Oriented Programming** languages such as **Java, Small Talk, C# etc.** As the **Child** class has to manage the dependency of more than one **Parent** class**. Multiple inheritance is not supported because it leads to deadly diamond problem.**
* Multilevel Inheritance - In **Multilevel Inheritance** a derived class will be **inheriting a parent class** and as well as the derived class **act as the parent class** to other class. As seen in the below diagram. **ClassB** inherits the property of **ClassA** and again **ClassB** act as a parent for **ClassC**. In Short **ClassA** parent for **ClassB** and **ClassB** parent for **ClassC**.

public class ClassA

{

public void dispA()

{

System.out.println("disp() method of ClassA");

}

}

public class ClassB extends ClassA

{

public void dispB()

{

System.out.println("disp() method of ClassB");

}

}

public class ClassC extends ClassB

{

public void dispC()

{

System.out.println("disp() method of ClassC");

}

public static void main(String args[])

{

//Assigning ClassC object to ClassC reference

ClassC c = new ClassC();

//call dispA() method of ClassA

c.dispA();

//call dispB() method of ClassB

c.dispB();

//call dispC() method of ClassC

c.dispC();

}

}

Hierarchical Inheritance – In **Hierarchical inheritance** one parent class will be inherited by **many** sub classes. As per the below example **ClassA** will be inherited by **ClassB, ClassC**and**ClassD**. **ClassA** will be acting as a parent class for **ClassB, ClassC**and**ClassD**.

public class ClassA

{

public void dispA()

{

System.out.println("disp() method of ClassA");

}

}

public class ClassB extends ClassA

{

public void dispB()

{

System.out.println("disp() method of ClassB");

}

}

public class ClassC extends ClassA

{

public void dispC()

{

System.out.println("disp() method of ClassC");

}

}

public class ClassD extends ClassA

{

public void dispD()

{

System.out.println("disp() method of ClassD");

}

}

public class HierarchicalInheritanceTest

{

public static void main(String args[])

{

//Assigning ClassB object to ClassB reference

ClassB b = new ClassB();

//call dispB() method of ClassB

b.dispB();

//call dispA() method of ClassA

b.dispA();

//Assigning ClassC object to ClassC reference

ClassC c = new ClassC();

//call dispC() method of ClassC

c.dispC();

//call dispA() method of ClassA

c.dispA();

//Assigning ClassD object to ClassD reference

ClassD d = new ClassD();

//call dispD() method of ClassD

d.dispD();

//call dispA() method of ClassA

d.dispA();

}

}

* Hybrid Inheritance **(Through Interface) -** Hybrid Inheritance is the combination of both Single and Multiple Inheritance. Again Hybrid inheritance is also not directly supported in Java only through interface we can achieve this. Flow diagram of the Hybrid inheritance will look like below. As you can **ClassA** will be acting as the **Parent** class for **ClassB & ClassC** and **ClassB & ClassC** will be acting as **Parent** for **ClassD.**

What is oop - Object-oriented programming System(OOPs) is a programming prototype based on the concept of “objects” that contain data and methods. The primary purpose of object-oriented programming is to increase the flexibility and maintainability of programs. Object oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works.

Exceptions - An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an *exception object*, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called *throwing an exception*.

Overloading and overriding - Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes

Why multiple inheritance is not allowed

Are you willing to relocate? Yes

Are you ready for learning any kind of technology? Yes

Are you ready to do night shifts? Yes