**Inheritance (or) IS-A Relationship:**

* Inheritance is the 1st Pillar of OOPS (Object Oriented Programming System)
* Inheritance means Acquire, Access, or Take etc.…...
* One Class is Acquiring Properties of Another Class is called as “Inheritance”

(OR)

* One Class Contains Members of Another Class without Re-writing it is called as “Inheritance”

Here, Properties means Methods & Variables.

* Inheritance is also called as “IS-A Relationship” or “Parent-Child Relationship”
* The Relationship is Defines through “**extends**” Keyword. Here, **extends** is a keyword which indicates we are Creating a New Class from an Existing Class.

**Super class and Sub class:**

* The class whose properties are acquired is called as “super class or parent class or Base class”
* The class who is acquiring the properties is called as “child class or Sub class or derive class”

Parent Class is Contains Only its Own Properties.

Child Class is Contains its Own Properties and Parent Class Properties.

If we create an Object for Parent Class, we can access Only Parent Class Properties But if we create an Object for Child Class, we can access Both Parent & Child Class Properties.

**Ex:**

class A    **//A is called superclass or parent class or base class**

{

//Body or implementation or logic

}

class B extends A **//B is called as subclass or child class or derived class**

{

//Body or implementation or logic

}

**Types Of Inheritance:**

There are different types of inheritance

1. Single Level inheritance
2. Multi-Level inheritance
3. Hierarchical inheritance
4. Multiple inheritance
5. Hybrid inheritance

**1. Single Level inheritance:**

It Contains “One Super Class and One Sub Class”

**2. Multi-Level inheritance:**

It Contains “Two Super Classes and Two Sub Classes”

(OR)

It Contains “One Grand Parent Class and One Parent Class and One Child Class”

**3. Hierarchical inheritance:**

It Contains “One super class and Two sub classes”

**4. Multiple inheritance:**

Multiple inheritance is “One class is inheriting two immediate super classes at the same time”

But in java, a class can extends only one class at a time.

So Multiple inheritance is not possible through classes because of

1. Ambiguity problem

2. Diamond problem

3. Constructor chaining problem

**1. Ambiguity problem:**

If one class extends two classes and in case, if both classes contain same method, then while calling a method, **JVM will get confuse** which class method to call this problem is known as Ambiguity problem

**2. Diamond problem:**

Since the structure/shape of class diagram is in diamond form it is also referred as Diamond problem

**3. Constructor chaining problem:**

In Multiple inheritance we Should call to Both Parent/Super Classes. By Using Constructor, it is not possible because in **One Constructor we Should have Only one call to Super(),** therefore here it leads to Problem of Constructor Chaining

**NOTE:**

Therefore, due to ambiguity problem, diamond problem and constructor chaining problem. **Multiple inheritance is not possible through class but it is possible through Interface.**

**5.Hybrid inheritance:**

It is a combination of Multiple and Hierarchical Inheritance since Multiple is not possible through classes. Hybrid is also not possible through classes.

**Advantages of Inheritance:**

* Reusability of code
* Avoid duplicity of code
* Breaking of Bigger logic into small logic
* Simple to Understand
* Easy to connect different logic at one place

--------------------------- \*\*\*\*\* ­--------------------------------------------------------------------------------

**Polymorphism:**

Poly means “Many” and morphism means “Forms”

One thing is showing multiple behavior is called as “Polymorphism”

(OR)

One entity showing different behaviour is called as “polymorphism”

**Types of Polymorphism:**

There are 2 Types

1. Compile Time Polymorphism
2. Run Time Polymorphism

**1. Compile Time Polymorphism:**

During compile time one thing is showing multiple behaviour is called as “compile time polymorphism”

**Ex:**  Method Overloading

**2. Run Time Polymorphism:**

During run/execution time one thing is showing multiple behaviour is called as “Run time polymorphism”

**Ex:** Method Overriding

--------------------------- \*\*\*\*\* ­--------------------------------------------------------------------------------

**Program For Armstrong Number:**

If we individually cube every digit of a number & sum them and then if we get a same number. Such number is called as “Armstrong Number”

**Ex:**

**public** **class** ArmstrongNumber

{

**public** **static** **void** main(String[] args)

{

**int** num=371,rem,arm=0,cp=num;

**while**(num>0)

{

rem=num%10;

num=num/10;

arm=arm+(rem\*rem\*rem);

}

System.***out***.println("Reverse Of a Number is: "+arm);

**if**(cp==arm)

{

System.***out***.println(cp+" Is An Armstrong Number");

}

**else**

{

System.***out***.println(cp+" Is Not An Armstrong Number");

}

}

}

**Output:** Reverse Of a Number is: 371

371 Is An Armstrong Number

--------------------------- \*\*\*\*\* ­--------------------------------------------------------------------------------