Relationship Between Java Classes :-

=> Use of relationship between java classes :-

1. Code Reusability
2. Less Execution Time
3. Less Memory Usage

=> Types of relationships :-

1. IS-A Relationship (Inheritance)

-> IS-A relationship is one in which data members of one class is obtained into another class through the concept of inheritance

-> Types of IS-A relationship :-

1.1 Single Inheritance
1.2 Multilevel Inheritance
1.3 Hierarchical Inheritance
1.4 Multiple Inheritance
1.5 Hybrid Inheritance

2. HAS-A Relationship (Association)

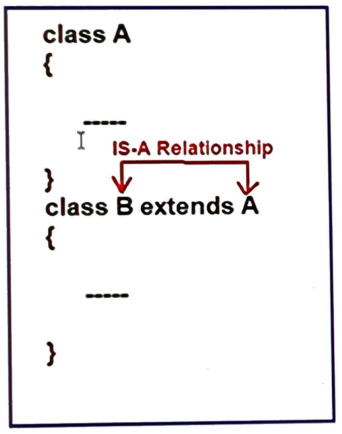
-> HAS-A relationship is one in which an object of one class is created as a data member into another class.

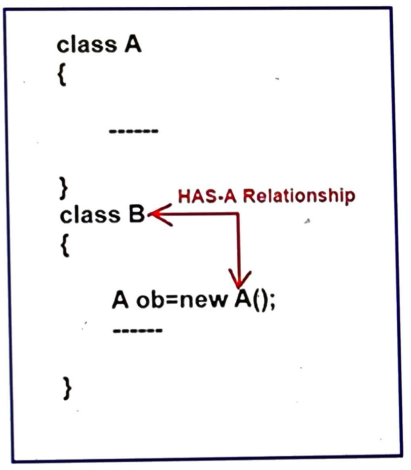
-> Types of HAS-A relationship :-

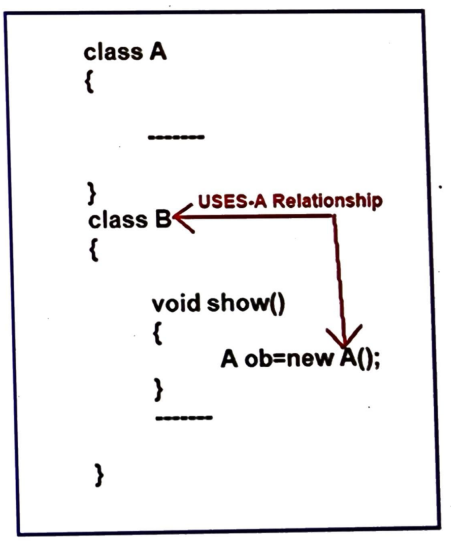
- 2.1 Aggregation
- 2.2 Composition

3. USES-A Relationship (Dependence)

-> USES-A relationship is one in which a method of one class is using an object of another class







=> Terms used for inheritance :-

1. Class
2. Sub-Class / Child Class
3. Super-Class / Parent Class
4. Reusability

=> IS-A Relationship :-

-> It is also known as "Inheritance"

-> IS-A Relationship or Inheritance is achieved by using "extends" keyword

-> All java classes except Object class will always have one parent class thus we can say that the total java API is implemented based on inheritance concept

=> Use of inheritance :-

1. Code Reusability
2. For Method Overriding to achieve runtime polymorphism

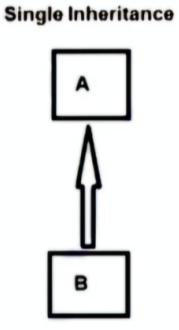
=> Syntax of inheritance :-

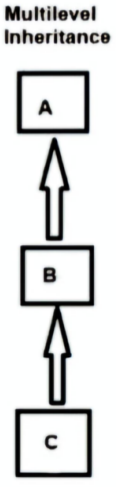
```
class Sub-Class extends Super-Class
    {
        //body
    }
```

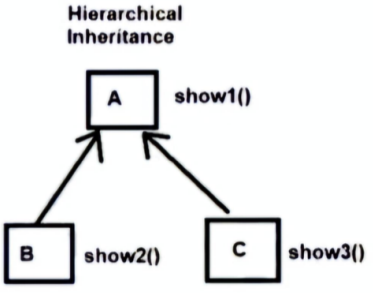
=> Types of Inheritance :-

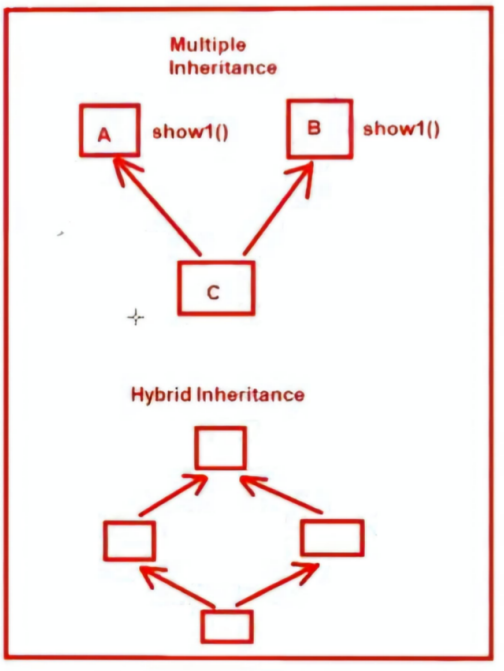
-> Total there are 5 types of inheritance :-

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance
5. Hybrid Inheritance









Single Inheritance:

```
public class A {
    public void show1(){
        System.out.println("Hi from class A");
    }
}


public class B extends A{
    public void show2(){
        System.out.println("Hi from class B");
    }
}

public class SingleInheritance {
    public static void main(String[] args) {
        A a = new A() ;
        a.show1();

        System.out.println("====================================");

        B b  = new B() ;
        b.show2();
        b.show1();

    }
}
```

==========

```
public class Animal {
    public void run(){
        // 1000 lines of code
        System.out.println("I am Eating");
        // 1000 lines of code
    }
}

public class Dog extends Animal {
    void eat(){
        System.out.println("Dog is Eating");
    }
}

public class InheritanceMain {
    public static void main(String[] args) {
        Dog dog = new Dog() ;
        dog.run();
        dog.eat();
    }
}
```

Multilevel Inheritance:

```
public class A {
    public void show1(){
        System.out.println("Hi from class A");
    }
}


public class B extends A{
    public void show2(){
        System.out.println("Hi from class B");
    }
}

public class C extends B{
    public void show3(){
        System.out.println("Hi From class C");
    }
}

public class multilevelInheritance {
    public static void main(String[] args) {
        A a = new A() ;
        a.show1();

        System.out.println("====================================");

        B b  = new B() ;
        b.show2();
        b.show1();

        System.out.println("================================");

         C c = new C() ;
         c.show1();
         c.show2();
         c.show3();

    }
}
```

Hierarchical inheritance

```
public class A {
    void showA(){
        System.out.println("Hi From A");
    }
}

public class B extends A{
    void showB(){
        System.out.println("Hi From B");
    }
}

public class C extends A{
    void showC(){
        System.out.println("Hi From C");
    }
}
```

Multiple Inheritance(Not supported by Java)

```
public class A {
    void show1(){
        System.out.println("Hi From A");
    }
}

public class B{
    void show1(){
        System.out.println("Hi From B");
    }
}

public class C extends A,B{

}

public class InhertanceMain {
    public static void main(String[] args) {
        C c = new C() ;
        c.show1();        //compiler gets confused which show1() method to call
    }
}
```

=> Points to Remember :-

-> Default Parent Class :- By default if any java class does not inherit any parent class then it inherits Object class

-> **Parent class can only be one** :- There can be only one parent class for every class and due to this java does not support multiple inheritance

-> **Which part is not inherited :-**

1. Private members of parent class is not inherited in child class
2. Constructors are not inherited because constructors are not the part of class members (class members are only fields, methods, nested classes)

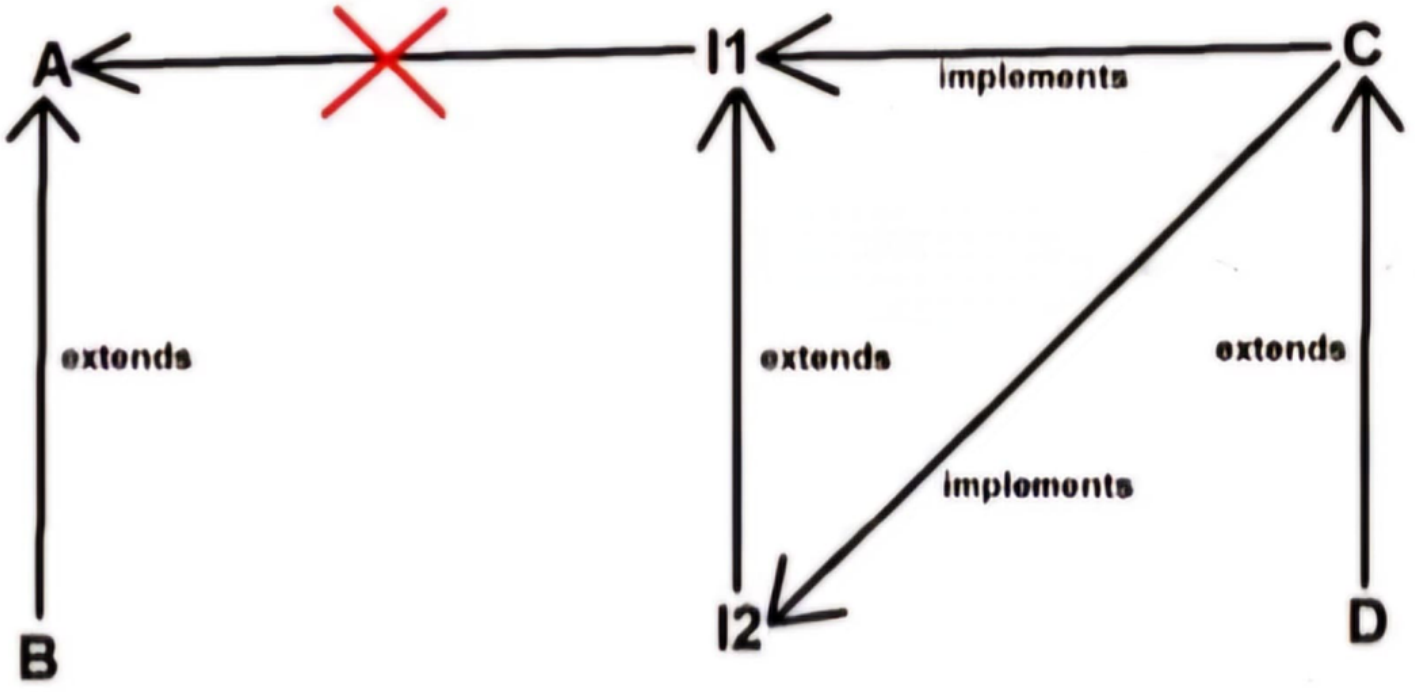
-> Cyclic inheritance is not possible

1. class A extends A
2. class A extends B { - }
   class B extends A { - }

-> Multiple and Hybrid inheritance is not possible in case of classes but it is possible in case of interfaces

Interview Questions :-

1. Why java does not support multiple inheritance
2. How we can achieve Multiple and Hybrid inheritance
3. Various possible combinations for inheritance



Various Possible combinations for inheritance :

1. class B extends A
2. interface I2 extends I1
3. interface I2 implements I1
4. interface I1 extends A ----> bcoz class property cannot be inherited to interface
5. interface I1 implements A ----> bcoz class property cannot be inherited to interface
6. class C implements I1
7. class C extends I1
8. class C implements I1, I2
9. class C extends B implements I1, I2
10. class C implements I1, I2 extends B