Interface: -

Why interface needed ?

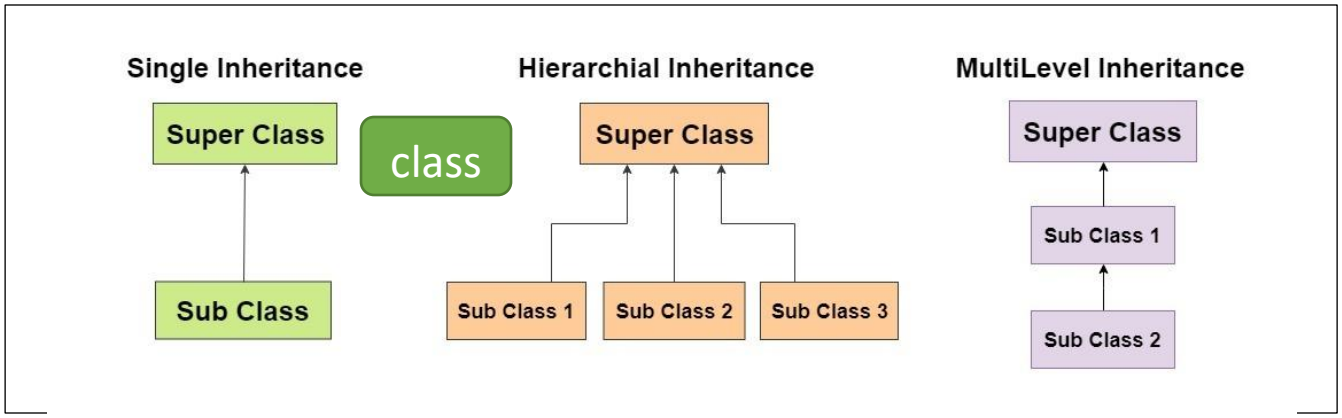
Sol:- To overcome the ambiguity in Multiple and Hybrid inheritance. Which can be not achieved by class. it's only possible by **using interface**. because they have 2 parents**. To understand the interface some basic concept to inheritance required.**

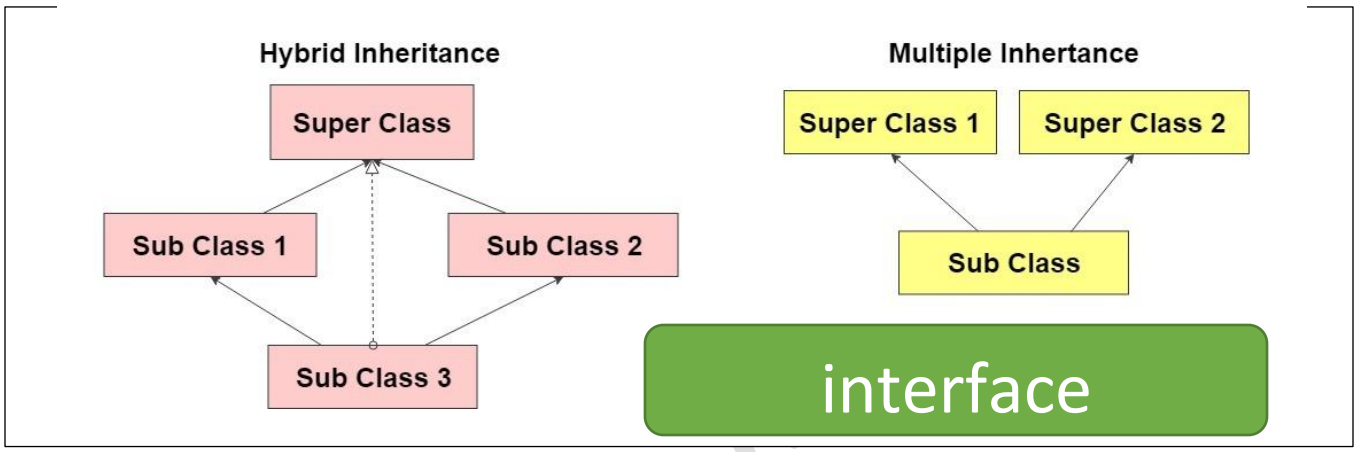
Types of inheritance:-

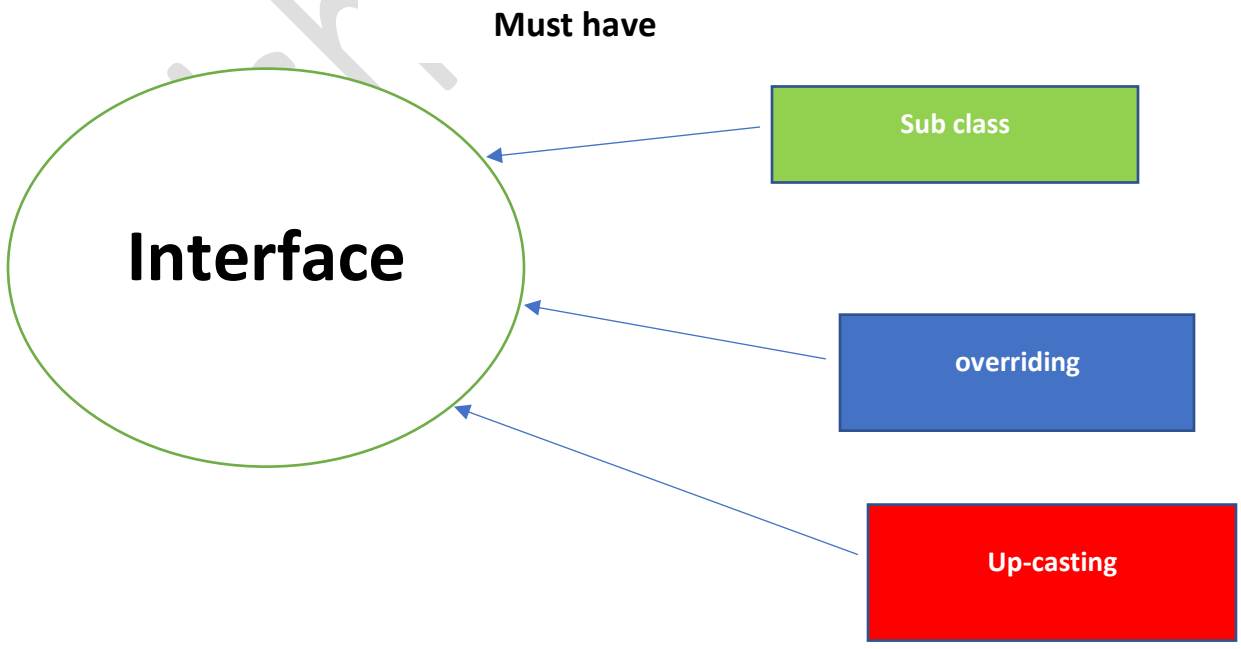
1. Single inheritance :- class.
2. Multi-level inheritance :-class.
3. Hierarchical inheritance:- class.
4. **Multiple inheritance:- interface.**
5. **hybrid inheritance :-interface**

extends and implements:-

1. Class to Class: - extends
2. **Interface to interface: - extends**
3. **Class to interface: - implements**
4. **Interface to interface: - Not possible**


Single Inheritance
Super Class
Sub Class
class
Hierarchial Inheritance
Super Class
Sub Class 1
Sub Class 2
Sub Class 3
MultiLevel Inheritance
Super Class
Sub Class 1
Sub Class 2


Hybrid Inheritance
Super Class
Sub Class 1
Sub Class 2
Sub Class 3
Multiple Inhertance
Super Class 1
Super Class 2
Sub Class
interface


Must have
Interface
Sub class
overriding
Up-casting

1. Interface is an intermediate between the **consumer and the service.**
2. An interface is used to achieve **100% abstraction class.**
3. An interface is also called as **rules repository or coding contract**.
4. Programmatically interface is declared by **using keyword interface**.
5. An interface **cannot inherit** either a class or abstract class and **not even object class.**
6. An interface has **no state.**
7. Any variables that you create **are really constants** that means there by default **public static finally**(so be careful about making mutable objects in interfaces).
8. Like **abstract classes, interfaces cannot be used to create objects.** Because interface doesn't contain constructor. But we can **make reference** of it that refers to the Object of its implementing class.
9. Interface can be used **as a refer** of any it sub class object.
10. A class can implement **more than one interface.**
11. Interfaces **specify what a class must do and not how**. An interface in java is **a blueprint of a class.** ➔ important
12. Interface can inherit any number of interfaces by using keyword **extends.**
13. Interface cannot super class even object class. Because if the interface inherits the super class then **all the concrete method from super class get inherit to the interface** which is violate the rules of interface
14. The **class can extends one super class at a same time** .but it can implements **multiple interface** at a same time . example

    The sequence must be like:-

    Class A extends parents implements Inf1,Inf2,Inf3,etc

15. A class which inherit the interface is called **class sub or implementation class.**
16. When a class implements an interface, **we must have to override all the** abstract method. If not, then we must make the sub class as abstract class.(incomplete class). Important.
17. It is good practice to **start with I or end with Inf.**

    For example :-

ISwitch **I**Requlator

Switch**Inf** Requlator**Inf**

18. All the method in interface are by default **public** and **abstract**

For example:-

```
public interface mouseinf
{
	void click();
	void doubleclick();
	void rightclick();
}
```

OR

Both are same

```
public interface mouseinf
{
	public abstract void click();
	public abstract void doubleclick();
	public abstract void rightclick();
}
```

19. There can be only **abstract methods in the Java interface**, not method body. **(only method signature, no body).**

For example :-

```
public k Smile();
public v eat();
```

BUT

Java Interface also **represents the IS-A relationship**.

It cannot be instantiated just like the abstract class.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we can have **private methods** in an interface

20. An interface **can also have Nested interface in it**.

For example:-

```
public interface Map
{
	public interface Entry
	{
		public k getkey();
		public v getvalue();
```

```
        }
}
```

Why multiple interface in not possible using class.

because of 2 reasons.

1) ambiguity while constructor chaining.
2) ambiguity while method execution.

Why we use the interface ?

1. It is used to **achieve multiple inheritance**.
2. It is used to **achieve loose coupling.**
3. **To achieve security** - hide certain details and only show the important details of an object (interface)
4. It is used to **achieve abstraction.**
5. Interfaces **specify what a class must do and not how.**
6. It can be used to **indicate jvm about certain activity** through marker interface.

TYPES OF INTERFACE:-

1. Marker interface/tagging.==➔NO method.
2. Regular interface.===➔user defined.
3. Functional interface.==➔ having 1 abstract method .

1.Marker interface:-

1. Marker interface is used to indicate the j.v.m about certain activity.
2. A marker interface is an empty interface which does not have any method in it .
3. Marker interface has some special meaning to the jvm.
4. We cannot create our own marker interface. If we create our own marker interface, then there is no used of it because it is empty.

5. Example:-
    1. Serializable.
    2. Clonnable.
    3. Remote interface.

Example :-

```
public interface Serializable
{
  // nothing here
}
```

2. Functional interface:-

1. A Functional interface is an interface which has only one abstract method .
2. Functional interface is used as rules repository or coding contract .which is used to impose or inject some rules.

Example :-
Comparable:- compareTo();
Comparator :-compare();
Runnable :- run();

An interface is similar to a class in the following ways −

1. An interface can contain any number of methods.
2. An interface is written in a file with a .java extension, with the name of the interface matching the name of the file.
3. The byte code of an interface appears in a .class file.
4. Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including −

1. You cannot instantiate an interface.
2. An interface does not contain any constructors.

3. All of the methods in an interface are abstract.
4. An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
5. An interface is not extended by a class; it is implemented by a class.
6. An interface can extend multiple interfaces

Program:-

s

```
package Amrit;
public interface mouse
{
	void click();
	void rightclick();
	void doubleclick();
}
```

interface

Override

```
package Amrit;
public class cpu implements mouse
{

	@Override
	public void click()
	{
		System.out.println("The resouce get selected");
	}
	@Override
	public void rightclick()
	{
		System.out.println("The resouce get selected");
	}
	@Override
	public void doubleclick()
	{
		System.out.println("Display option");
	}
}
```

Sub Class

```java
package Amrit;
public class computer
{
	void user()
	{
		mouse m=new cpu();
		m.click();
		m.doubleclick();
		m.rightclick();
	}
	public static void main(String[] args)
	{
		computer c=new computer();
		c.user();
	}
}
```

Up-Casting

some exter information:-