# Interfaces :-



1) Defining an Interface.
2) Implementing Interface
3) applying Interfaces
4) Variables in interface and Extending interfaces
5) Difference blw classes and Interfaces.

## 1) Defining an Interface :-

### Abstract :-

Abstraction is a process of hiding the Implementation details and showing only functionality to the user.

Ex:- sending SMS where you type the text and send the message. you don't know the internal processing about the message delivery.

There are two ways to achieve abstraction in java

1) Abstract class (0 to 100%)
2) Interface (100%)

### Interface :-

Interface is used to achieve abstraction and multiple Inheritance in Java. Interface is just like a class, which contains only abstract method or collections of abstract methods. It can't have a method body.

Note :- The interface is doesn't support object and normal methods and can't contain a Constructor.

→ Java interface also represents Is-A relationship.

Importance of interface :- why use Java. Interface?

1) It is used to achieve abstraction.

2) By interface, we can support the functionality of multiple inheritance.

3) It can be used to achieve loose coupling.

→ since Java 8, we can have default and static methods in an interface.

→ Since Java 9, we can have private methods in an Interface.

How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the Empty body.

Syntax :-

```
interface <interface_name> {
    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

* Interface methods are by default public & abstract.

* Interface fields or (variables) are by default public, static and final.

* Interface method must be ~~over~~ overridden inside the Implementing classes.

Compiles Internally:-

```
Interface Customer{
double sal = 20000.00;
void print();
}
```

Customer.java

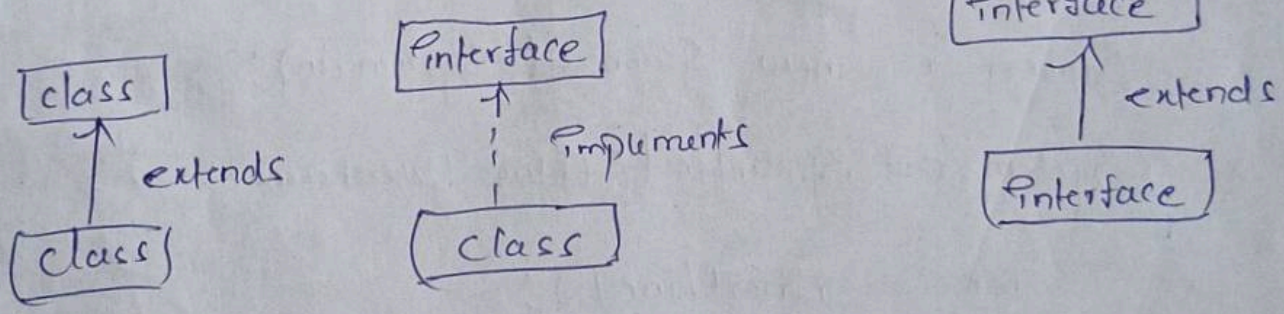
→ compiler →

```
interface Printable{
Public static final
double sal =
20000.00;
Public abstract
void Printer();
}
```

Customer.class

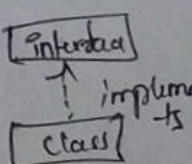
The relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an Interface extends another interface, but a class implements an interface.



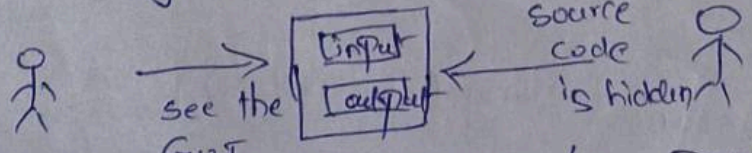

## (2) Implementing Interface:-

* Implementing Interface using "Implements" keyword. The class is 'implements' all the methods of interface. because the interface methods are abstract methods.




→ Interface nothing but deals between client & developer.

Ex:-




Client using Interface through ↓ because we are not implementing.

import java.util.Scanner;

Ex:-

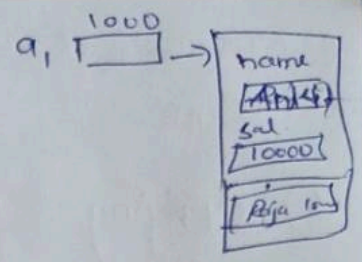
```
Interface client
 {
   void input(); // public + abstract
   void output();
 }
class Raju implements client
 {
   String name;   // public + static + final
   double sal;    } Global variables.

 Public void input();
    {
 Scanner r = new Scanner (System.in);
 System.out.println("Enter Username :");
     name = r.nextLine();
 System.out.println("Enter Salary: ");
     sal = r.nextLine();
                 Double
 }
 Public void output()
    {
 System.out.println (name + " " + sal);
    }
Public static void main (String [] args){
```

client
Reference

Client C = new Raju ;

Reference object name — new memory space — class object




C.input();

C.output();

}

O/P:- }

Enter Username :

Ankit

Enter Salary :

50,000.00

Ankit 50000.00

Another Example :-

Simple Example :-

```
interface Paintable {
  void paint();
}
class A6 implements Paintable {
  Public void paint() {
  System.out.println("Hello"); }
Public static void main (String args[]){
  A6 obj = new A6();
    obj.paint();
  }
}
```

O/P :- Hello

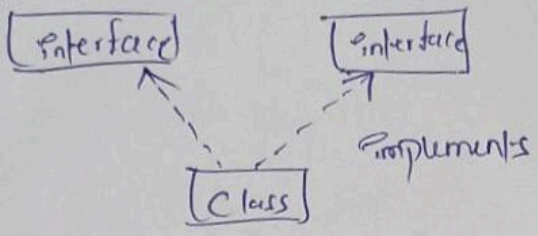
② Drawble (Rectangle, Circle).

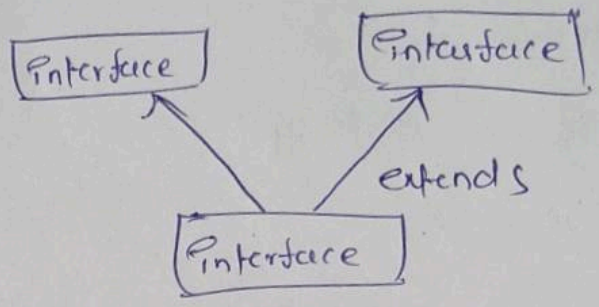
③ Bank

3) Applying Interface :- in multiple Inheritance.

## Multiple Inheritance in Java by Interface :-

If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.




Ex:-

```
interface Paintable{
    void paint();
}
Interface Showable{
    void show();
}
class A7 implements Paintable, Showable{
    Public void paint(){
    System.out.println("Hello");}
Public void show(){ System.out.println("welcome");}
Public static void main(String args[]){
    A7 obj = new A7();
    obj.paint();
    obj.show();
}}
```

Syntax

```
Interface Interface_name
{
// statement1
}
Interface Interface_name
{
//Public abstract void
    methodname();
}
Class implements Interface1, Interface2
{
statement2
}
```

o/p:- Hello
welcome

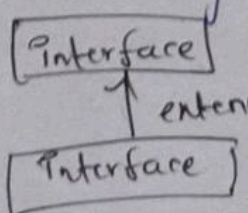
Multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.

Ex:- interface Paintable {
void paint();
}
Interface Shapable {
void paint();
}
class TestInterface3 implements Paintable, Shapable {
Public void paint() { System.out.Println("Hello"); }
Public static void main (String args[]) {
TestInterface3 obj = new TestInterface3();
obj.paint();
}
}

O/P:- Hello.

## Extending Interface:-

A class implements an interface, but one interface extends another Interface. Using 'extends' keyword.



Example Program:-

```
Interface Printable {
    void print();
}
Interface showable extends Printable {
    void show();
}
class Test1 implements showable {
Public void print() { System.out.println("Hello"); }
Public void show() { System.out.println("welcome"); }
Public static void main (String args[]) {
        Test1 obj = new Test1();
        obj.print();
        obj.show();
    }
}
```

O/P:- Hello
welcome

## Variables in Interfaces:-

The variables are default in public + static + final in Interface. lets go to understood variables in interface example Program.

```
~~ace CustomerRaj~~
interface CustomerRaj {
Pinl- amt=100;  // Public + static + final
void Purchase();  // Public + abstract
}
class sellerSanju implements CustomerRaj
{
  @Override // overs-writing base class method you can't- over-write compiler give warn -ing.
  Public void Purchase()
  {
    // amt=5 final
  System.out.println("Raj needs " + amt + " kg rice");
  }
}
class check
  {
    Public static void main (String[] args) {
      CustomerRaj c = new sellerSanju();
        c.Purchase();
      System.out.println (CustomerRaj.amt); // 100 steeck // static
    }
}
```

// Raj needs 100 kg rice.

memory is allocated for declared variables by the Compiler.

o/p :- Raj needs 100 kg rice

100

(5).

...re between a class and an Interface:

| class | Interface |
|---|---|
| 1) The keyword used to create a class is "class". | 1) The keyword used to create an Interface is "interface". |
| 2) A class can be instantiated i.e, objects of a class can be created. | 2) An Interface cannot be instantiated i.e, objects cannot be created. |
| 3) classes does not support multiple inheritance. | 3) Interface support multiple inheritance. |
| 4) It can be inherited by another class using the keyword 'extends'. | 4) It can be inherited by a class by using the keyword 'implements' and it can be inherited by an interface using the keyword 'extends'. |
| 5) It can contain Constructors. | 5) It cannot contain Constructors. |
| 6) It cannot contain abstract methods. | 6) It contains abstract methods only. |
| 7) variables and methods in a class can be declare using any access specifier (public, private, default, protected) | 7) All variables and methods in a Interface are declare as public |
| 8) variables in a class can be static, final or neither. | 8) All variables are static and final. |

# Interface Variables

```
Interface anu {
    int amount=50;
    void run();
}
```

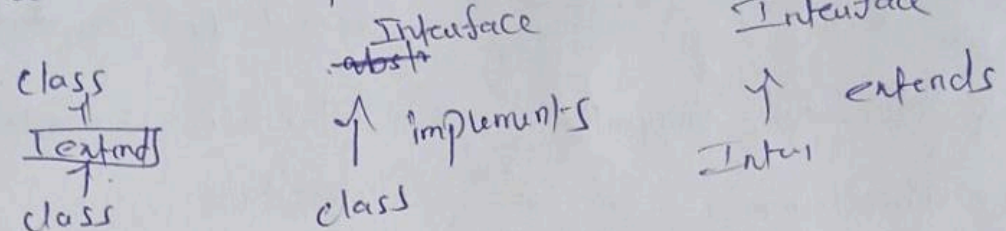
anu.java.

→

```
interface anu {
    Public static final int amount=50;
    Public abstract void run();
}
```

anu.class.

Inheritance Relationship between classes and interface.




Tightly Coupling:-

→ when two classes are highly dependent on each others, it is called tight coupling.

Loosely Coupling:-

→ the classes are independent of each other.

Definition Interface :-

→ An interface in java is a blueprint of a class. It has static constants and abstract methods. The interface in java is a mechanism to achieve abstraction.