UNIT IV
INHERITANCE AND POLYMORPHISM

INTERFACES IN JAVA

An **Interface in Java** programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a behavior. A Java interface contains static constants and abstract methods.

What are Interfaces in Java?

The interface in Java is *a* mechanism to achieve abstraction. Traditionally, an interface could only have abstract methods (methods without a body) and public, static, and final variables by default. It is used to achieve abstraction and multiple inheritances in Java. In other words, interfaces primarily define methods that other classes must implement. Java Interface also represents the IS-A relationship.

In Java, the abstract keyword applies only to classes and methods, indicating that they cannot be instantiated directly and must be implemented.

When we decide on a type of entity by its behavior and not via attribute we should define it as an interface.

Syntax for Java Interfaces

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

To declare an interface, use the interface keyword. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static, and final by default. A class that implements an interface must implement all the methods declared in the interface. To implement the interface, use the implements keyword.

Uses of Interfaces in Java

Uses of Interfaces in Java are mentioned below:

- *It is used to achieve total abstraction.*
- *Since java does not support multiple inheritances in the case of class, by using an interface it can achieve multiple inheritances.*
- *Any class can extend only 1 class, but can any class implement an infinite number of interfaces.*
- *It is also used to achieve loose coupling.*
- *Interfaces are used to implement abstraction.*

So, the question arises why use interfaces when we have abstract classes?
The reason is, abstract classes may contain non-final variables, whereas variables in the interface are final, public, and static.

```
// A simple interface
interfacePlayer
{
   final int id = 10;
int move();
}
```

Relationship Between Class and Interface
A class can extend another class similar to this an interface can extend another interface. But only a class can extend to another interface, and vice-versa is not allowed.

Difference Between Class and Interface
Although Class and Interface seem the same there have certain differences between Classes and Interface. The major differences between a class and an interface are mentioned below:

Class	Interface
In class, you can instantiate variables and create an object.	In an interface, you must initialize variables as they are final but you can't create an object.
A class can contain concrete (with implementation) methods	The interface cannot contain concrete (with implementation) methods.
The access specifiers used with classes are private, protected, and public.	In Interface only one specifier is used- Public.

Implementation: *To implement an interface, we use the keyword* ***implements***
Java

```
// Java program to demonstrate working of
// interface

importjava.io.*;

// A simple interface
interfaceIn1 {

// public, static and final
finalint a = 10;

// public and abstract
```

```
void display();
}

// A class that implements the interface.
classTestClassimplements In1 {

// Implementing the capabilities of
// interface.
public void display(){
System.out.println("Geek");
    }

// Driver Code
publicstatic void main(String[] args)
    {
TestClass t = newTestClass();
t.display();
System.out.println(t.a);
    }
}
```

Output

```
Geek
10
```