* An **interface** is a collection of abstract methods and constants.
* It is used to achieve fully abstraction and multiple inheritance in Java.
* Java Interface also **represents IS-A relationship**.

1. An interface is not a class.
2. Writing an interface is similar to writing a class, but they are two different concepts.
3. A class describes the attributes and behaviors of an object.
4. An interface contains behaviors that a class implements.

**An interface is similar to a class in the following ways:**

* An interface can contain any number of methods.
* An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
* The bytecode of an interface appears in a **.class** file.
* 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:

* You cannot instantiate an interface.
* An interface does not contain any constructors.
* All of the methods in an interface are abstract.
* An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
* An interface is not extended by a class; it is implemented by a class.
* An interface can extend multiple interfaces.

## Declaring Interfaces:

The **interface** keyword is used to declare an interface. Here is a simple example to declare an interface:

## Example:

Let us look at an example that depicts encapsulation:

import java.lang.\*;

//Any number of import statements

public interface NameOfInterface

{

//Any number of final, static fields

//Any number of abstract method declarations\

}

**Interfaces have the following properties:**

**all methods in interface are by default public and abstract.**

**all fields in interface are by default public static final.**

## Why use Java interface?

**There are mainly three reasons to use interface. They are given below.**

* **It is used to achieve fully abstraction.**
* **By interface, we can support the functionality of multiple inheritance.**
* **It can be used to achieve loose coupling.**

#### **Understanding 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.**relationship between class and interface

**Example:**

/\* File name : Animal.java \*/

interface Animal {

public void eat();

public void travel();

}

/\*File name : MammalInt.java \*/

public class MammalInt implements Animal{

public void eat(){

System.out.println("Mammal eats");

}

public void travel(){

System.out.println("Mammal travels");

}

public int noOfLegs(){

return 0;

}

public static void main(String args[]){

MammalInt m = new MammalInt();

m.eat();

m.travel();

}

}

Extending Interfaces:

* An interface can extend another interface, similarly to the way that a class can extend another class.
* The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

The following Sports interface is extended by Hockey and Football interfaces.

//Filename: Sports.java

public interface Sports

{

public void setHomeTeam(String name);

public void setVisitingTeam(String name);

}

//Filename: Football.java

public interface Football extends Sports

{

public void homeTeamScored(int points);

public void visitingTeamScored(int points);

public void endOfQuarter(int quarter);

//Filename: Hockey.java

public interface Hockey extends Sports

{

public void homeGoalScored();

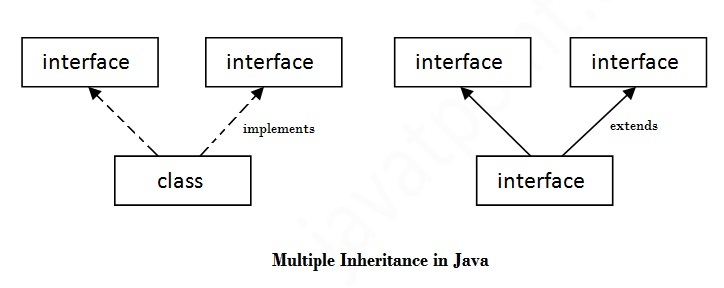
public void visitingGoalScored();

public void endOfPeriod(int period);

public void overtimePeriod(int ot);

}

**Multiple inheritance in Java by interface**



**Notes:-**

* A class can extend only one class, but implement many interfaces.
* An interface can extend one or more interface, The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.
  + **interface D extends B,C,A**
* If a abstract class implements interface it need not to implement abstract method of interface.
* When overriding the interface methods in the subclass, the signature of interface methods has to be maintained in the sub class.
* Checked exceptions should not be declared on implementation methods other than the ones declared by the interface method or subclasses of those declared by the interface method.