Interface in Java

What is interface ?

An interface in java is a blueprint of a class. Typically it has public static final data members and public n abstract methods only.

The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface (not method body)(true till JDK 1.7) . It is used to achieve full abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship.

It cannot be instantiated just like abstract class.

Why java interfaces?

1. It is used to achieve full abstraction.
2. By interface, we can support the functionality of multiple inheritance.
3. It can be used to achieve loose coupling.

(Interfaces allow complete separation between WHAT(specification or a contract) is to be done Vs HOW (implementation details) it's to be done

The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members.

---

syntax of interface

default(no modifier)/public interface NameOfInterface extends comma separated list of super interfaces
{
  //data members --- public static final : added implicitly by javac
  int DATA=100;
  //methods -- public abstract : added implicitly by javac
  double calc(double d1,double d2);

}

Implementing class syntax

default(no modifier)/public class NameOfClass extends SuperCls implements comma separated list of interfaces
{
  //Mandatory for implementation class to be non-abstract(concrete): MUST define/implement all abstract methods inherited from all i/fs.

}

eg : public class Circle extends Shape implements Computable,Runnable {...}

1. Relationship between classes and interfaces

A class inherits from another class(extends), an interface extends another interfaces(extends) but a class implements an interface.

2. Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.
eg :

```
Multiple inheritance in java
interface Printable{
void print();
}

interface Showable{
void show();
}

class A implements Printable,Showable{

public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
A obj = new A();
obj.print();
obj.show();
 }
}
```

Question

Multiple inheritance is not supported through class in java but it is possible by interface, why?

Multiple inheritance is not supported in case of class, since it can create an ambiguity. But it is supported in case of interface because there is no ambiguity as implementation is provided by the implementation class.

For example:

```
interface Printable{
void print();
}
interface Showable{
void print();
}

class TestTnterface1 implements Printable,Showable{
public void print(){System.out.println("Hello");}
```

```
public static void main(String args[]){
TestTnterface1 obj = new TestTnterface1();
obj.print();
 }
}
```

As you can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface1, so there is no ambiguity.

Interface inheritance

A class implements interface but one interface extends another interface .

```
interface Printable{
void print();
}
interface Showable extends Printable{
void show();
}
class Testinterface2 implements Showable{

public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
Testinterface2 obj = new Testinterface2();
obj.print();
obj.show();
 }
}
```

Q) What is marker or tagged interface?

An interface that has no member is known as marker or tagged interface. For example: Serializable, Cloneable, Remote etc. They are used to provide some essential information to the JVM(Run time marker) so that JVM may perform some useful operation.

```
//How Serializable interface is written?
public interface Serializable{
}
```

Nested Interface in Java

Note: An interface can have another interface i.e. known as nested interface.

eg :

```
interface printable{
 void print();
 interface MessagePrintable{
   void msg();
 }
}
```

Q . What is a functional i/f
An interface containing sing abstract methods (SAM)
eg : Comparator , Runnable , Consumer...

Difference between abstract class and interface

Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.

But there are many differences between abstract class and interface.

Abstract class Vs Interface

1) Abstract class can have abstract and non-abstract methods. Interface can have only abstract methods.
2) Abstract class doesn't support multiple inheritance. Interface supports multiple inheritance.
3) Abstract class can have final, non-final, static and non-static variables. Interface has onlypublic,static and final variables.
4) Abstract class can have static methods, main method and constructor. Interface can't have static methods, main method or constructor.
5) Abstract class can provide the implementation of interface. Interface can't provide the implementation of abstract class.
6) The abstract keyword is used to declare abstract class. The interface keyword is used to declare interface.
7) Example:

```
public abstract class Shape{
public abstract void draw();
}
```

Example:

```
public interface Drawable{
void draw();
}
```

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

Abstract Class vs. Interface

Java provides and supports the creation of abstract classes and interfaces. Both implementations share some common features, but they differ in the following features:

1. All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.

2.A class may implement a number of Interfaces, but can extend only one abstract class.

3.
In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
Abstract classes can implement interfaces without even providing the implementation of interface methods.

4.
Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.

5.
Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.

6.
An interface is absolutely abstract and cannot be instantiated, doesn't support a constructor. An abstract class also cannot be instantiated BUT can contain a constructor to be used while creating concrete(non abstract) sub class instance.