Interface

Review Abstract

- abstract keyword
- We may create a new class all the time when we need the abstract .
- An abstract class may or may not contain the abstract method.
- If a class contains abstract method , the class should be declared an abstract class.
- We cannot instantiate the abstract class.
- To access the data of an abstract class, we use inheritance.
- toString() method :- String representation of the class.

Interface

- An interface is a reference type in java. It is a collection of abstract methods.
- It is similar to the class.
- A class implements an interface using ’implements’ keyword, thereby inheriting the abstract methods of the interface.
- It may contain any number of methods.
- An interface may also contain constants, default methods, static methods and nested types.
- Method bodies exists only for default methods and static methods.

Interface not like a Class

- 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.

Properties of Interface

- An interface is implicitly abstract. You do not need to use the **abstract** keyword while declaring an interface.
- Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
- Methods in an interface are implicitly public.

Declaration of Interface

- public interface interfaceName

{

// methods

Methods are public abstract method.

}

Overriding interface

When overriding methods defined in interfaces, there are several rules to be followed −

- 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.
- The signature of the interface method and the same return type or subtype should be maintained when overriding the methods.
- An implementation class itself can be abstract and if so, interface methods need not be implemented

Implementing interface

- A class can implement more than one interface at a time.
- A class can extend only one class but implement many interfaces.
- An interface can extend another interface, in a similar way as a class can extend another class.

For Example

For Example:

```
public interface Emp{
public void empInfo();
}
public interface Department extends Emp{
public void empInfo();
public void getEmpPosition();
}
Public interface Accounts extends Department{
public void empInfo();
public void getEmpPosition();
public void salary();
}
```

Extending multiple interfaces

- A Java class can only extend one parent class. Multiple inheritance is not allowed. Interfaces are not classes, however, and an interface can extend more than one parent interface.
- The extends keyword is used once, and the parent interfaces are declared in a comma-separated list.

For example:

public interface Department extends Emp , Accounts;

Difference between Interface & Abstract

Interface

- ‘interface’ keyword is used to declare.
- ‘implements’ keyword used to implement.
- When we only know specification but not implementation .
- Implements as many interface as you want multiple inheritance is achieved.
- Does not contain any constructor
- Variables are public static final.

Abstract

- ‘abstract’ keyword used to declare.
- ‘extends’ keyword used to implement .
- When we know partial implementation of method
- Usually represents the one concept does not support multiple inheritance.
- Contains constructor
- Variables could be any.

Diiference

interface

- Class Emp{
- getEmpsal();
- getEMpInfo();
- Interface a
- Interface b
- Interface extends a,b{

abstract

- Class Emp{
- getEmpInfo(){
- }
- getEmpSal();
- Class A
- Class b
- Class c extends a,b