What is Interface?

The interface is a blueprint that can be used to implement a class. The interface does not contain any concrete methods (methods that have code). All the methods of an interface are abstract methods.

An interface cannot be instantiated. However, classes that implement interfaces can be instantiated. Interfaces never contain instance variables but, they can contain public static final variables (i.e., constant class variables)

What Is Abstract Class?

A class which has the abstract keyword in its declaration is called abstract class. Abstract classes should have at least one abstract method. , i.e., methods without a body. It can have multiple concrete methods.

Abstract classes allow you to create blueprints for concrete classes. But the inheriting class should implement the abstract method.

Abstract classes cannot be instantiated.

Important Reasons For Using Interfaces

1. Interfaces are used to achieve abstraction.
2. Designed to support dynamic method resolution at run time
3. It helps you to achieve loose coupling.
4. Allows you to separate the definition of a method from the inheritance hierarchy

Important Reasons For Using Abstract Class

1. Abstract classes offer default functionality for the subclasses.
2. Provides a template for future specific classes
3. Helps you to define a common interface for its subclasses
4. Abstract class allows code reusability.

Interface Vs. Abstract Class

Parameters	Interface	Abstract class
Speed	Slow	Fast
Multiple	Implement several Interfaces	Only one abstract class

Inheritances		
Structure	Abstract methods	Abstract & concrete methods
When to use	Future enhancement	To avoid independence
Inheritance/ Implementation	A Class can implement multiple interfaces	The class can inherit only one Abstract Class
Default Implementation	While adding new stuff to the interface, it is a nightmare to find all the implementors and implement newly defined stuff.	In case of Abstract Class, you can take advantage of the default implementation.
Access Modifiers	The interface does not have access modifiers. Everything defined inside the interface is assumed public modifier.	Abstract Class can have an access modifier.
When to use	It is better to use interface when various implementations share only method signature. Polymorphic hierarchy of value types.	It should be used when various implementations of the same kind share a common behavior.
Data fields	the interface cannot contain data fields.	the class can have data fields.
Multiple Inheritance Default	A class may implement numerous interfaces.	A class inherits only one abstract class.
Implementation	An interface is abstract so that it can't provide any code.	An abstract class can give complete, default code which should be overridden.
Use of Access modifiers	You cannot use access modifiers for the method, properties, etc.	You can use an abstract class which contains access modifiers.
Usage	Interfaces help to define the peripheral abilities of a class.	An abstract class defines the identity of a class.
Defined fields	No fields can be defined	An abstract class allows you to define both fields and constants
Inheritance	An interface can inherit multiple interfaces but cannot inherit a class.	An abstract class can inherit a class and multiple interfaces.
Constructor or destructors	An interface cannot declare constructors or destructors.	An abstract class can declare constructors and destructors.

Limit of Extensions	It can extend any number of interfaces.	It can extend only one class or one abstract class at a time.
Abstract keyword	In an abstract interface keyword, is optional for declaring a method as an abstract.	In an abstract class, the abstract keyword is compulsory for declaring a method as an abstract.
Class type	An interface can have only public abstract methods.	An abstract class has protected and public abstract methods.

<u>Sample code for Interface and Abstract Class in Java</u>

Interface Syntax

```
interface name{

//methods

}
```

Java Interface Example:

```
interface Pet {

   public void test();

}

class Dog implements Pet {

   public void test() {

      System.out.println("Interface Method Implemented");

   }

   public static void main(String args[]) {

      Pet p = new Dog();

      p.test();

   }
}
```

Abstract Class Syntax

```
abstract class name{
    // code
}
```

Abstract class example:

```
abstract class Shape {
    int b = 20;
    abstract public void calculateArea();
}
public class Rectangle extends Shape {
    public static void main(String args[]) {
        Rectangle obj = new Rectangle();
        obj.b = 200;
        obj.calculateArea();
    }
    public void calculateArea() {
        System.out.println("Area is " + (obj.b * obj.b));
    }
}
```