

## Abstract class vs interface in Java

**Difference No.1:** **Abstract class** can **extend** only **one class** or **one abstract class** at a time

```
class Example1{
    public void display1(){
        System.out.println("display1 method");
    }
}
abstract class Example2{
    public void display2(){
        System.out.println("display2 method");
    }
}
abstract class Example3 extends Example1{
    abstract void display3();
}
class Example4 extends Example3{
    public void display3(){
        System.out.println("display3 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display3();
    }
}
```

Output: display3 method

**Interface** can **extend** any number of **interfaces** at a time

```
//first interface
interface Example1{
    public void display1();
}
//second interface
interface Example2 {
    public void display2();
}
//This interface is extending both the above interfaces
interface Example3 extends Example1, Example2{
}
class Example4 implements Example3{
    public void display1(){
        System.out.println("display2 method");
    }
    public void display2(){
        System.out.println("display3 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display1();
    }
}
```

Output: display2 method

**Difference No.2: Abstract class** can be **extended(inherited)** by a **class** or **abstract class**

```
class Example1{
    public void display1(){
        System.out.println("display1 method");
    }
}

abstract class Example2{
    public void display2(){
        System.out.println("display2 method");
    }
}

abstract class Example3 extends Example2{
    abstract void display3();
}

class Example4 extends Example3{
    public void display2(){
        System.out.println("Example4-display2 method");
    }
    public void display3(){
        System.out.println("display3 method");
    }
}

class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display2();
    }
}
```

Output: Example4-display2 method

**Interfaces** can be **extended** only by **interfaces**. **Classes** has to **implement** them instead of **extend**

```
interface Example1{
    public void display1();
}

interface Example2 extends Example1{
}

class Example3 implements Example2{
    public void display1(){
        System.out.println("display1 method");
    }
}

class Demo{
    public static void main(String args[]){
        Example3 obj=new Example3();
        obj.display1();
    }
}
```

```
}
}
```

Output: display1 method

**Difference No.3: Abstract class** can have both **abstract** and **concrete** methods

```
abstract class Example1 {
    abstract void display1();
    public void display2(){
        System.out.println("display2 method");
    }
}
class Example2 extends Example1{
    public void display1(){
        System.out.println("display1 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}
```

**Interface** can only have abstract methods, they cannot have concrete methods

```
interface Example1{
    public abstract void display1();
}
class Example2 implements Example1{
    public void display1(){
        System.out.println("display1 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}
```

Output: display1 method

**Difference No.4: In abstract class**, the keyword 'abstract' is mandatory to declare a method as an abstract

```
abstract class Example1{
    public abstract void display1();
}
class Example2 extends Example1{
    public void display1(){
        System.out.println("display1 method");
    }
    public void display2(){
        System.out.println("display2 method");
    }
}
```

```

}
class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}

```

In **interfaces**, the keyword **'abstract'** is **optional** to declare a method as an abstract **because** all the methods are abstract by default

```

interface Example1{
    public void display1();
}

class Example2 implements Example1{
    public void display1(){
        System.out.println("display1 method");
    }
    public void display2(){
        System.out.println("display2 method");
    }
}

class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}

```

**Difference No.5: Abstract class** can have **protected** and **public abstract** methods

```

abstract class Example1{
    protected abstract void display1();
    public abstract void display2();
    public abstract void display3();
}

class Example2 extends Example1{
    public void display1(){
        System.out.println("display1 method");
    }
    public void display2(){
        System.out.println("display2 method");
    }
    public void display3(){
        System.out.println("display3 method");
    }
}

class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}

```

```
}
```

**Interface** can have only **public abstract methods**

```
interface Example1{
    void display1();
}
class Example2 implements Example1{
    public void display1(){
        System.out.println("display1 method");
    }
    public void display2(){
        System.out.println("display2 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}
```

**Difference No.6: Abstract class** can have **static, final** or **static final** variables with any access specifier

```
abstract class Example1{
    private int numOne=10;
    protected final int numTwo=20;
    public static final int numThree=500;
    public void display1(){
        System.out.println("Num1="+numOne);
    }
}
class Example2 extends Example1{
    public void display2(){
        System.out.println("Num2="+numTwo);
        System.out.println("Num2="+numThree);
    }
}
class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
        obj.display2();
    }
}
```

**Interface** can have only **public static final (constant) variable**

```
interface Example1{
    int numOne=10;
}
class Example2 implements Example1{
```

```
    public void display1(){
        System.out.println("Num1="+numOne);
    }
}
class Demo{
    public static void main(String args[]){
        Example2 obj=new Example2();
        obj.display1();
    }
}
```