# Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

Before learning the Java abstract class, let's understand the abstraction in Java first.

### Abstraction in Java

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

### Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Abstract class in Java

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have constructors and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.

abstract class Bike{

abstract void run();

}

class Honda4 extends Bike{

void run()

{

System.out.println("running safely");

}

public static void main(String args[]){

Bike obj = new Honda4();

obj.run();

}

}

An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Java Interface also **represents the IS-A relationship**.

It cannot be instantiated just like the abstract class.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we can have **private methods** in an interface.

## Why use Java interface?

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

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

## How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

#### The 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**.



interface Printable

{

void print();

}

interface Showable

{

void show();

}

class MultpleInheri implements Printable,Showable

{

public void print()

{

System.out.println("Hello");

}

public void show()

{

System.out.println("Welcome");

}

public static void main(String args[]){

MultpleInheri obj = new MultpleInheri ();

obj.print();

obj.show();

}

}

Output:Hello

Welcome

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| Abstract class can **have abstract and non-abstract**methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| An **abstract class**can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| An **abstract class**can be extended using keyword "extends". | An **interface class**can be implemented using keyword "implements". |
| A Java**abstract class**can have class members like private, protected, etc. | Members of a Java interface are public by default. |
| **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%).