### Abstraction in Java

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

### 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](https://www.javatpoint.com/java-constructor) and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.

**Example of abstract class**

1. **abstract** **class** A{}

### Abstract Method in Java

A method which is declared as abstract and does not have implementation is known as an abstract method.

**Example of abstract method**

1. **abstract** **void** printStatus();//no method body and abstract
2. **abstract** **class** Bike{
3. **abstract** **void** run();
4. }
5. **class** Honda4 **extends** Bike{
6. **void** run()
7. {
8. System.out.println("running safely");
9. }
10. **public** **static** **void** main(String args[]){
11. Bike obj = **new** Honda4();
12. obj.run();
13. }
14. }
15. running safely
16. **abstract** **class** Shape{
17. **abstract** **void** draw();
18. }
19. //In real scenario, implementation is provided by others i.e. unknown by end user
20. **class** Rectangle **extends** Shape{
21. **void** draw()
22. {
23. System.out.println("drawing rectangle");
24. }
25. }
26. **class** Circle1 **extends** Shape{
27. **void** draw(){System.out.println("drawing circle");}
28. }
29. //In real scenario, method is called by programmer or user
30. **class** TestAbstraction1{
31. **public** **static** **void** main(String args[]){
32. Shape s=**new** Circle1();//In a real scenario, object is provided through method, e.g., getShape() method
33. s.draw();
34. }
35. }
36. drawing circle
37. **abstract** **class** Bank{
38. **abstract** **int** getRateOfInterest();
39. }
40. **class** SBI **extends** Bank{
41. **int** getRateOfInterest(){**return** 7;}
42. }
43. **class** PNB **extends** Bank{
44. **int** getRateOfInterest(){**return** 8;}
45. }
47. **class** TestBank{
48. **public** **static** **void** main(String args[]){
49. Bank b;
50. b=**new** SBI();
51. System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
52. b=**new** PNB();
53. System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
54. }}

Rate of Interest is: 7 %

Rate of Interest is: 8 %

### Abstract class having constructor, data member and methods

An abstract class can have a data member, abstract method, method body (non-abstract method), constructor, and even main() method.

*File: TestAbstraction2.java*

1. //Example of an abstract class that has abstract and non-abstract methods
2. **abstract** **class** Bike{
3. Bike(){System.out.println("bike is created");}
4. **abstract** **void** run();
5. **void** changeGear(){System.out.println("gear changed");}
6. }
7. //Creating a Child class which inherits Abstract class
8. **class** Honda **extends** Bike{
9. **void** run(){System.out.println("running safely..");}
10. }
11. //Creating a Te
12. st class which calls abstract and non-abstract methods
13. **class** TestAbstraction2{
14. **public** **static** **void** main(String args[]){
15. Bike obj = **new** Honda();
16. obj.run();
17. obj.changeGear();
18. }
19. }

bike is created

running safely..

gear changed

**next »« prev**

Interface in Java

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*](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).

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

## 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.

## 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.

1. **interface** <interface\_name>{
3. // declare constant fields
4. // declare methods that abstract
5. // by default.
6. }



1. **interface** printable{
2. **void** print();
3. }
4. **class** A6 **implements** printable{
5. **public** **void** print(){System.out.println("Hello");}
7. **public** **static** **void** main(String args[]){
8. A6 obj = **new** A6();
9. obj.print();
10. }
11. }
12. Hello
13. //Interface declaration: by first user
14. **interface** Drawable{
15. **void** draw();
16. }
17. //Implementation: by second user
18. **class** Rectangle **implements** Drawable{
19. **public** **void** draw(){System.out.println("drawing rectangle");}
20. }
21. **class** Circle **implements** Drawable{
22. **public** **void** draw(){System.out.println("drawing circle");}
23. }
24. //Using interface: by third user
25. **class** TestInterface1{
26. **public** **static** **void** main(String args[]){
27. Drawable d=**new** Circle();//In real scenario, object is provided by method e.g. getDrawable()
28. d.draw();
29. }}
30. drawing circle
31. **interface** Bank{
32. **float** rateOfInterest();
33. }
34. **class** SBI **implements** Bank{
35. **public** **float** rateOfInterest(){**return** 9.15f;}
36. }
37. **class** PNB **implements** Bank{
38. **public** **float** rateOfInterest(){**return** 9.7f;}
39. }
40. **class** TestInterface2{
41. **public** **static** **void** main(String[] args){
42. Bank b=**new** SBI();
43. System.out.println("ROI: "+b.rateOfInterest());
44. }}
45. ROI: 9.15

## Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



1. **interface** Printable{
2. **void** print();
3. }
4. **interface** Showable{
5. **void** show();
6. }
7. **class** A7 **implements** Printable,Showable{
8. **public** **void** print(){System.out.println("Hello");}
9. **public** **void** show(){System.out.println("Welcome");}
11. **public** **static** **void** main(String args[]){
12. A7 obj = **new** A7();
13. obj.print();
14. obj.show();
15. }
16. }
17. Output:Hello
18. Welcome

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 that are given below.

|  |  |
| --- | --- |
| Abstract class | Interface |
| 1) 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. |
| 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 only static and final variables. |
| 4) Abstract class can provide the implementation of interface. | Interface can't provide the implementation of abstract class. |
| 5) The abstract keyword is used to declare abstract class. | The interface keyword is used to declare interface. |
| 6) An abstract class can extend another Java class and implement multiple Java interfaces. | An interface can extend another Java interface only. |
| 7) An abstract class can be extended using keyword "extends". | An interface can be implemented using keyword "implements". |
| 8) A Java abstract class can have class members like private, protected, etc. | Members of a Java interface are public by default. |
| 9)Example: public abstract class Shape{ public abstract void draw(); } | Example: public interface Drawable{ void draw(); |