# why we can't extend more than one class in java?

# That's because Java does not support multiple inheritance in order to avoid circular dependencies or ambiguity. If class C extends Both A and B class, then the super class while calling invoke would become ambiguous.Java does not support multiple inheritance, that's why you can't extend a class from two different classes at the same time. Rather, use a single class to extend from, and use interfaces to include additional functionality.

can multiple inheritance possible in Java using abstract class?

 Basically, the rule says that you can inherit from (extend) as many classes as you want, but if you do, only one of those classes can contain concrete (implemented) methods. A class can extend at most one abstract class, but may implement many interfaces. That is, Java supports a limited form of multiple inheritance.

# 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).

Properties of interface

1. It always contains final data members.
2. It cannot be instantiated.
3. All methods of interface are abstract and public in nature.
4. The class which implements interface need to provide functionality for the methods declare in the interface.
5. One can use interface to implement PM (Partial multiple inheritance) and DMD (Dynamic memory dispatch).
6. Interface always implements in derived class.

Declaration of Interface

import java.util.\*;

interface interfacename

{

//define abstract methods

//define constants

}

#### Why And When To Use Interfaces?

1) To achieve security - hide certain details and only show the important details of an object (interface).

2) Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because the class can **implement** multiple interfaces. **Note:** To implement multiple interfaces, separate them with a comma (see example below).

### 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.

#### Points to Remember

* 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.

Difference between abstract class and interface:

|  |  |
| --- | --- |
| **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(); } |