



# What is Diamond Problem in Java

In **Java**, the **diamond problem** is related to multiple inheritance. Sometimes it is also known as the **deadly diamond problem** or **deadly diamond of death**. In this section, we will learn **what is the demand problem in Java** and **what is the solution to the diamond problem**.

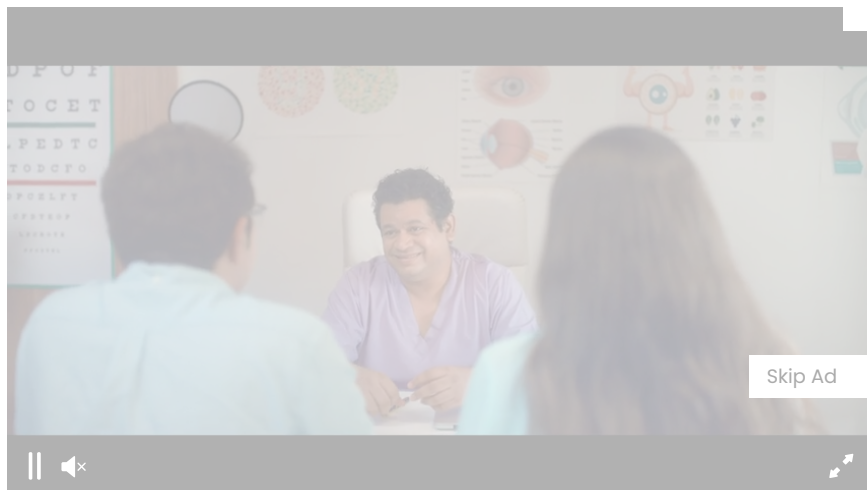
Before moving to the diamond problem let's have a look at **inheritance in Java**.

## Inheritance in Java

Inheritance is a relation between two classes, the parent and child class. The child class (sub-class) inherits all the properties of the parent class (super-class). To define the relation, we use **extends** keyword. For example:

```
public class A extends B
{
}
```

When we inherit the properties of a class into another class, a copy of the super-class (parent class) is created in the sub-class (child class) object. Hence, by using the sub-class object, we can access the member of super-class, also.



## Multiple Inheritance

It is a feature of an object-oriented concept, where a class can inherit properties of more than one parent class. The feature creates a problem when there exist methods with the same name and signature in both the super-class and sub-class. When we call the method, the compiler gets confused and cannot determine which class method to be called and even on calling which class method gets the priority.

Let's understand the concept through an example.

### A.java

```
class A
{
    public void display()
    {
        System.out.println("class A display() method called");
    }
}
```

```

}
class B extends A
{
@Override
public void display()
{
System.out.println("class B display() method called");
}
}
class C extends A
{
@Override
public void display()
{
System.out.println("class C display() method called");
}
}
//not supported in Java
public class D extends B,C
{
public static void main(String args[])
{
D d = new D();
//creates ambiguity which display() method to call
d.display();
}
}

```

- Class B and class C inherits the class A. The display() method of class A is overridden by the class B and class C.
- Class D inherits the class B and class C (which is invalid in Java). Assume that we need to call the display() method by using the object of class D, in this scenario Java compiler does not know which display() method to call. Therefore, it creates ambiguity.

When we compile the above program, it shows the **compiler error**, as we have shown below.

```

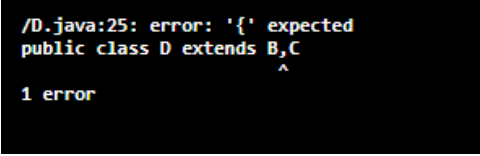
/D:/java:25: error: '{' expected
public class D extends B,C
                        ^
1 error

```

Due to these complications and ambiguities, Java does not support multiple inheritance. It creates problems during various operations, for example, constructor chaining and casting. Hence, it will be good to avoid it for making things simple.

## The Diamond Problem

The diamond problem is a common problem in Java when it comes to inheritance. Inheritance is a very popular property in an object-oriented programming language, such as C++, Java, etc. There are different types of inheritance such as, single, multiple, multi-level, and hybrid inheritance. But remember that **Java does not support the multiple inheritance** because of the diamond problem.



```
/D.java:25: error: '{' expected
public class D extends B,C
                        ^
1 error
```

As simple inheritance allows a child class to derive properties from one super-class. for example, if class B inherits properties from only one super-class A, then it is called simple inheritance, and Java supports them.

Multi-level inheritance allows a child class to inherit properties from a class that can inherit properties from some other classes. For example, class C can inherit its property from B class which itself inherits from A class. Java also supports them.

What Java does not allow is multiple inheritance where one class can inherit properties from more than one class. It is known as the **diamond problem**. In the above figure, we find that class D is trying to inherit from class B and class C, that is not allowed in Java.

It is an ambiguity that can rise as a consequence of allowing multiple inheritance. It is a serious problem for other OPPs languages. It is sometimes referred to as the **deadly diamond of death**.

## The Solution of Diamond Problem

The solution to the diamond problem is **default methods** and **interfaces**. We can achieve multiple inheritance by using these two things.

The default method is similar to the abstract method. The only difference is that it is defined inside the interfaces with the default implementation. We need not to override these methods. Because they are already implementing these interfaces.

The advantage of interfaces is that it can have the same default methods with the same name and signature in two different interfaces. It allows us to implement these two interfaces, from a class. We must override the default methods explicitly with its interface name.

```
InterfaceName.super.methodName();
```

### DemoClass.java

```
interface DemoInterface1
{
    public default void display()
    {
        System.out.println("the display() method of DemoInterface1 invoked");
    }
}

interface DemoInterface2
{
    public default void display()
    {
        System.out.println("the display() method of DemoInterface2 invoked");
    }
}

public class DemoClass implements DemoInterface1, DemoInterface2
{
    public void display()
    {
        DemoInterface1.super.display();
        DemoInterface2.super.display();
    }

    public static void main(String args[])
    {
        DemoClass obj = new DemoClass();
        obj.display();
    }
}
```

### Output:

```
The display() method of DemoInterface1 invoked  
The display() method of DemoInterface2 invoked
```

In the above example, we see that both the interfaces have the same method name and signature. We have invoked the method by an interface name that does not create any ambiguity between methods.

In the following example, we have removed the default methods.

#### DemoClass.java

```
interface DemoInterface  
{  
    //default method  
    default void display()  
    {  
        System.out.println("The display() method invoked");  
    }  
}  
//interface without default method  
interface DemoInterface1 extends DemoInterface  
{  
}  
//interface without default method  
interface DemoInterface2 extends DemoInterface  
{  
}  
//implementation class code  
public class DemoClass implements DemoInterface1, DemoInterface2  
{  
    public static void main(String args[])  
    {  
        DemoClass obj = new DemoClass();  
        //calling method  
        obj.display();  
    }  
}
```

#### Output:

```
The dispaly() method invoked
```

Therefore, we can achieve multiple inheritance by using the interface. It is also the solution to the diamond problem.

[← Prev](#)[Next →](#)[Learn more](#)

For Videos Join Our Youtube Channel: [Join Now](#)

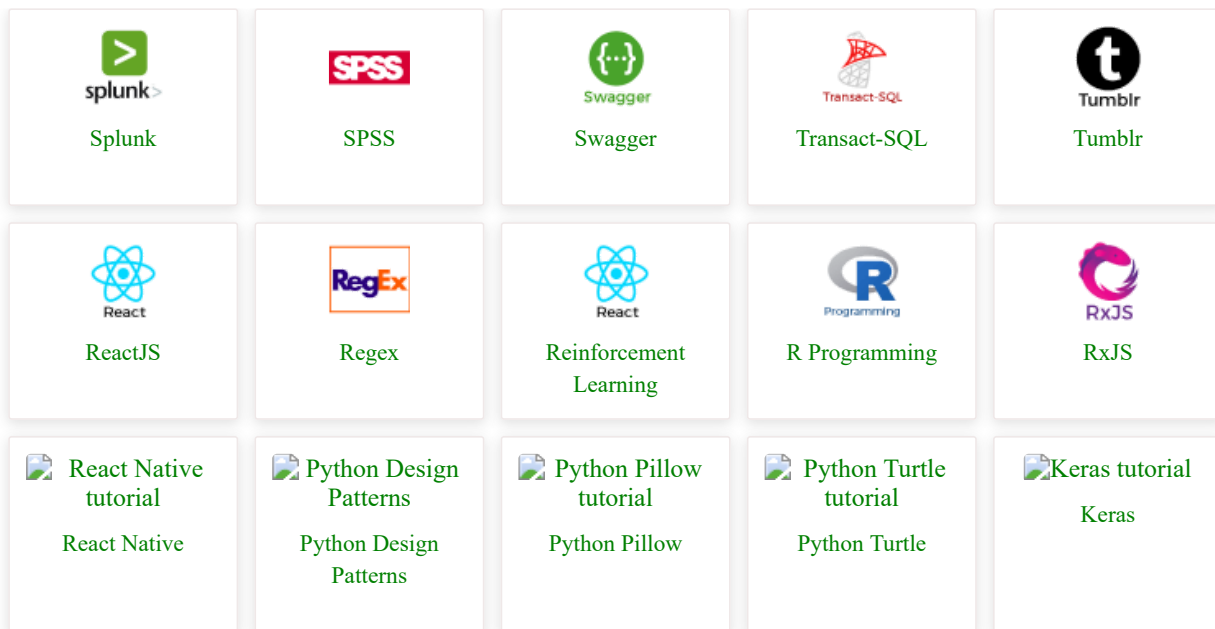
## Feedback

- Send your Feedback to [feedback@javatpoint.com](mailto:feedback@javatpoint.com)

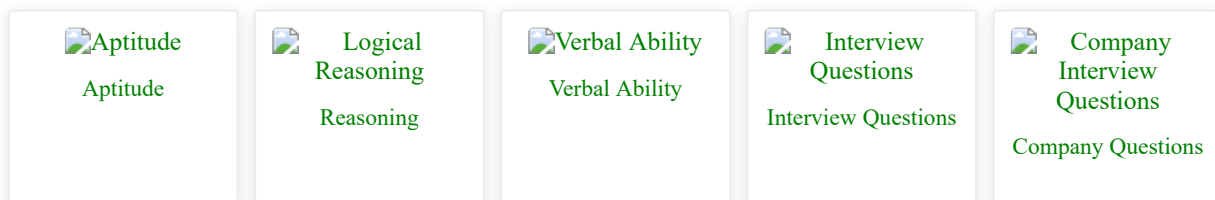
## Help Others, Please Share



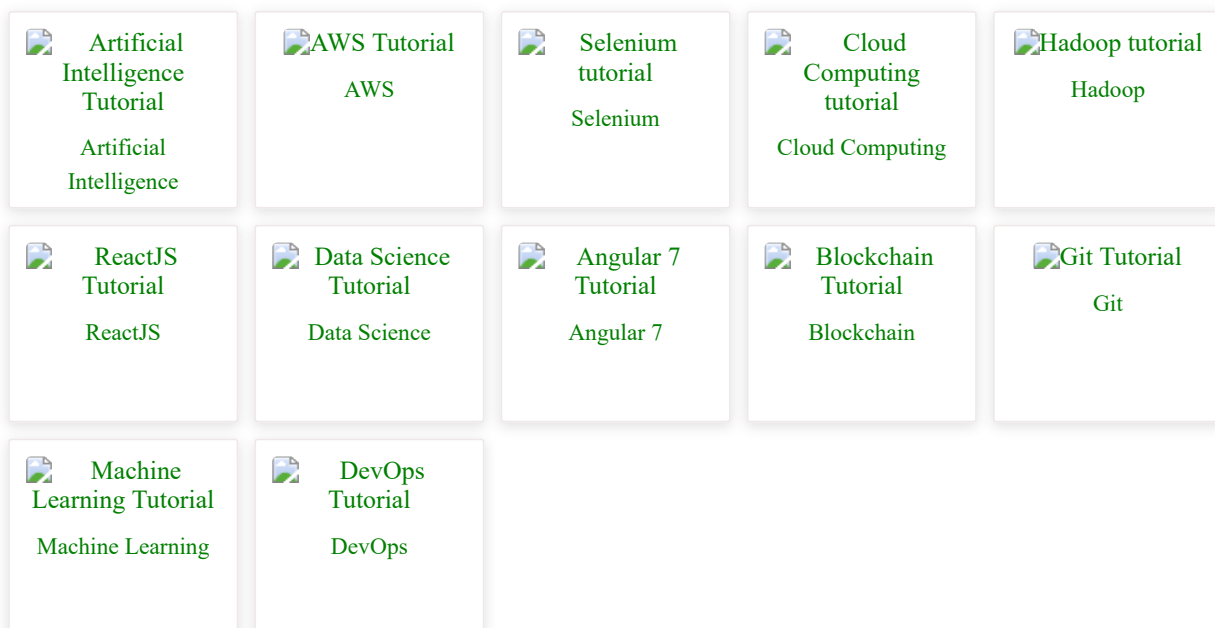
[Learn Latest Tutorials](#)



## Preparation




## Trending Technologies








## B.Tech / MCA

 DBMS tutorial  
DBMS


 Data Structures  
tutorial  
Data Structures


 DAA tutorial  
DAA

 Operating  
System tutorial  
Operating System


 Computer  
Network tutorial  
Computer Network

 Compiler  
Design tutorial  
Compiler Design


 Computer  
Organization and  
Architecture  
Computer  
Organization

 Discrete  
Mathematics  
Tutorial  
Discrete  
Mathematics

 Ethical Hacking  
Tutorial  
Ethical Hacking


 Computer  
Graphics Tutorial  
Computer Graphics


 Software  
Engineering  
Tutorial  
Software  
Engineering


 html tutorial  
Web Technology


 Cyber Security  
tutorial  
Cyber Security


 Automata  
Tutorial  
Automata


 C Language  
tutorial  
C Programming


 C++ tutorial  
C++

 Java tutorial  
Java

 .Net  
Framework  
tutorial  
.Net

 Python tutorial  
Python

 List of  
Programs  
Programs

 Control  
Systems tutorial  
Control System

 Data Mining  
Tutorial  
Data Mining

 Data  
Warehouse  
Tutorial  
Data Warehouse

