Can we Overload or Override static methods in java ?

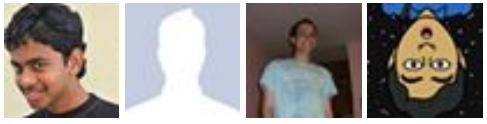
Let us first define Overloading and Overriding.

Overriding : Overriding is a feature of OOP languages like Java that is related to run-time polymorphism. A subclass (or derived class) provides a specific implementation of a method in superclass (or base class).
The implementation to be executed is decided at run-time and decision is made according to the object used for call. Note that signatures of both methods must be same.

Overloading: Overloading is also a feature of OOP languages like Java that is related to compile time (or static) polymorphism. This feature allows different methods to have same name, but different signatures, especially number of input parameters and type of input paramaters. Note that in both C++ and Java, methods cannot be overloaded according to return type.

Can we overload static methods?
The answer is 'Yes'. We can have two ore more static methods with same name, but differences in input parameters. For example, consider the following Java program.

```
// filename Test.java
public class Test {
    public static void foo() {
        System.out.println("Test.foo() called ");
    }
    public static void foo(int a) {
        System.out.println("Test.foo(int) called ");
    }
    public static void main(String args[])
    {
        Test.foo();
        Test.foo(10);
    }
```

```
}
```

Output:

```
Test.foo() called
Test.foo(int) called
```

Can we overload methods that differ only by static keyword?
We cannot overload two methods in Java if they differ only by static keyword (number of parameters and types of parameters is same). See following Java program for example. This behaviour is same in C++ (See point 2 of this).

```
// filename Test.java
public class Test {
    public static void foo() {
        System.out.println("Test.foo() called ");
    }
    public void foo() { // Compiler Error: cannot redefine foo()
        System.out.println("Test.foo(int) called ");
    }
    public static void main(String args[]) {
        Test.foo();
    }
}
```

Output: Compiler Error, cannot redefine foo()

Can we Override static methods in java?
We can declare static methods with same signature in subclass, but it is not considered overriding as there won't be any run-time polymorphism. Hence the answer is 'No'.
If a derived class defines a static method with same signature as a static method in base class, the method in the derived class hides the method in the base class.

```
/* Java program to show that if static method is redefined by
   a derived class, then it is not overriding. */

// Superclass
class Base {

    // Static method in base class which will be hidden in subclass
    public static void display() {
        System.out.println("Static or class method from Base");
    }
```


Popular Posts

```
     // Non-static method which will be overridden in derived class
     public void print()  {
          System.out.println("Non-static or Instance method from Base")
     }
}

// Subclass
class Derived extends Base {

     // This method hides display() in Base
     public static void display() {
          System.out.println("Static or class method from Derived");
     }

     // This method overrides print() in Base
     public void print() {
          System.out.println("Non-static or Instance method from Derive
    }
}

// Driver class
public class Test {
     public static void main(String args[ ])  {
        Base obj1 = new Derived();

        // As per overriding rules this should call to class Derive's s
        // overridden method. Since static method can not be overridden
        // calls Base's display()
        obj1.display();

        // Here overriding works and Derive's print() is called
        obj1.print();
     }
}
```

Output:

```
Static or class method from Base
Non-static or Instance method from Derived
```

Following are some important points for method overriding and static methods in Java.

1) For class (or static) methods, the method according to the type of reference is called, not according to the abject being referred, which means method call is decided at compile time.

2) For instance (or non-static) methods, the method is called according to the type of object being referred, not according to the type of reference, which means method calls is decided at run time.

3) An instance method cannot override a static method, and a static method cannot hide an instance method. For example, the following program has two compiler errors.

```
/* Java program to show that if static methods are redefined by
   a derived class, then it is not overriding but hidding. */

// Superclass
class Base {

    // Static method in base class which will be hidden in subclass
    public static void display() {
        System.out.println("Static or class method from Base");
    }

     // Non-static method which will be overridden in derived class
     public void print()  {
         System.out.println("Non-static or Instance method from Base")
    }
}

// Subclass
class Derived extends Base {

    // Static is removed here (Causes Compiler Error)
    public void display() {
        System.out.println("Non-static method from Derived");
    }

    // Static is added here (Causes Compiler Error)
    public static void print() {
        System.out.println("Static method from Derived");
   }
}
```

4) In a subclass (or Derived Class), we can overload the methods inherited from the superclass. Such overloaded methods neither hide nor override the superclass methods — they are new methods, unique to the subclass.

Recent Comments

References:
http://docs.oracle.com/javase/tutorial/java/landl/override.html

This article is contrubuted by **Chandra Prakash**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Inner class in java
- How to compare two arrays in Java?
- Can we override private methods in Java?
- Static class in Java
- Do we need forward declarations in Java?
- Checked vs Unchecked Exceptions in Java
- Bitwise right shift operators in Java
- Private and final methods in Java

Writing code in comment? Please use **ideone.com** and share the link here.

11 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

steve • a month ago
static main method in java

kodingexamples.blogspot.com/20...

Reply • Share ›

kishor classic • 2 months ago
The static method is called by using Class _name only. So in above program I
objeect? Is it possible to call a static method by object?

If i run the program it shows following warning but run successfully and show
The static method display() from the type Base should be accessed in a static

Reply • Share ›

vikas grover • 2 months ago
we can't override static method we can just hide them ...
static method are loaded before the class loading and when these methods ar
method. and on the bases of class reference it called the particular static metl
compile time polymorphism not run time polymorphism ,, so we can' t override

Reply • Share ›

peter nortan • 5 months ago
static method cannot be override, it can hide only,
http://muhammadkhojaye.blogspo...

Manoj Kumar • 8 months ago

Method Overriding VS Method Hiding in java

If and instance method in a subclass has same signature and return type as ir
called overriding.Method signature means name, and the number and the type
parameters should be same in method written in super class and sub class.

overriding method can also return a subtype of the type returned by the super
variant return type.
Foe example

public class Animal {

public void test() {

}
}

public class Dog extends Animal {

public void test() {

see more

2 • Reply • Share ›

World Traveler • 10 months ago

Difference between Overloading and Overriding
Using overloading and overriding, you can get the concept of polymorphism. P
multiple forms. Using one name you can do multiple of actions in multiple plac

Difference between Overloading and Overriding are here-
http://cybarlab.blogspot.com/2...

• Reply • Share ›

kunal dexit • 11 months ago

You said, "If a derived class defines a static method with same signature as a method in the derived class hides the method in the base class. "

But the output you shown is =>.
Static or class method from Base.

This is contradicting the above sentence. Or am I getting it wrong? Please cle

Manish Gautam ➦ kunal dexit • 11 months ago

Base obj1 = new Derived();
Here reference variable is of type base class right??
so since it is calling a static method, this method will be called with its
That's why you are getting output as :
"Static or class method from Base."

aditya gupta ➦ kunal dexit • 11 months ago

Check out the source given in the end....it tells the distinction between

```
/* Paste your code here (You may delete these lines if not wri
```

kartik ➦ kunal dexit • 11 months ago

That is how it is. It is called hiding only. You can look at this way. If ther
base class would have been called everywhere even for derived class.
class, it hides method of base class and all calls on derived class/obje
specific method. It is just a name other than overriding because there i

Harendra kunal dexit • 11 months ago

static mean "static", static doesn't know what is base or what is derive
remain with the class definition[bytecode]. Please correct me if i am w

2 • Reply • Share ›

Subscribe Add Disqus to your site