Can we override private methods in Java?

Let us first consider the following Java program as a simple example of Overriding or Runtime Polymorphism.

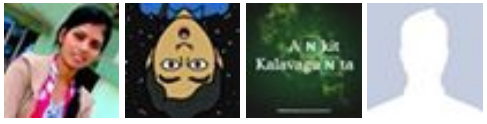
```
class Base {
  public void fun() {
     System.out.println("Base fun");
  }
}

class Derived extends Base {
  public void fun() {  // overrides the Base's fun()
     System.out.println("Derived fun");
  }
  public static void main(String[] args) {
      Base obj = new Derived();
      obj.fun();
  }
}
```

The program prints “Derived fun”.
The Base class reference ‘obj’ refers to a derived class object (see expression “Base obj = new Derived()”). When fun() is called on obj, the call is made according to the type of referred object, not according to the reference.

Is Overiding possible with private methods?
Predict the output of following program.

```
class Base {
  private void fun() {
     System.out.println("Base fun");
  }
```

```
}

class Derived extends Base {
  private void fun() {
     System.out.println("Derived fun");
  }
  public static void main(String[] args) {
      Base obj = new Derived();
      obj.fun();
  }
}
```

We get compiler error “fun() has private access in Base” (See this). So the compiler tries to call base class function, not derived class, means fun() is not overridden.

An inner class can access private members of its outer class. What if we extend an inner class and create fun() in the inner class?

An Inner classes can access private members of its outer class, for example in the following program, *fun()* of *Inner* accesses private data member *msg* which is fine by the compiler.

```
/* Java program to demonstrate whether we can override private method
   of outer class inside its inner class */
class Outer {
     private String msg = "GeeksforGeeks";
     private void fun() {
          System.out.println("Outer fun()");
     }

     class Inner extends Outer {
         private void fun()  {
               System.out.println("Accessing Private Member of Outer:
         }
     }

     public static void main(String args[])  {

          // In order to create instance of Inner class, we need an Ou
          // class instance. So, first create Outer class instance and
          // inner class instance.
          Outer o = new Outer();
          Inner  i   = o.new Inner();

          // This will call Inner's fun, the purpose of this call is t
          // show that private members of Outer can be accessed in Inn
          i.fun();
```

Geometric Algorithms

Popular Posts

```
            // o.fun() calls Outer's fun (No run-time polymorphism).
            o = i;
            o.fun();
        }
}
```

Output:

```
Accessing Private Member of Outer: GeeksforGeeks
Outer fun()
```

In the above program, we created an outer class and an inner class. We extended Inner from Outer and created a method fun() in both Outer and Inner. If we observe our output, then it is clear that the method fun() has not been overriden. It is so because ***private methods are bonded during compile time and it is the type of the reference variable – not the type of object that it refers to – that determines what method to be called.***. As a side note, private methods may be performance-wise better (compared to non-private and non-final methods) due to static binding.

Comparison With C++

1) In Java, inner Class is allowed to access private data members of outer class. This behavior is same as C++ (See this).

2) In Java, methods declared as private can never be overridden, they are in-fact bounded during compile time. This behavior is different from C++. In C++, we can have virtual private methods (See this).

This article is contributed 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 Overload or Override static 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.

12 Comments GeeksforGeeks

Sort by Newest

Join the discussion

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 35 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

nagarjuna • 6 months ago

Use Annotations for clear picture...look at http://javaojava.blogspot.in/2... for cl

1 ▲ | ▼ • Reply • Share ›

Madan Ram • 7 months ago

there is a typo "We extended Inner from Outer and created a method fun()" it s
from Inner and created a method fun()".

1 ▲ | ▼ • Reply • Share ›

Yuvaraj Velmayil • 8 months ago

This is completely wrong. We can reuse the private method signature. The ab
'public' to 'private' which is not allowed. When we declare 'private', the method
to override, but then, sub classes can reuse the same signature to declare its

1 ▲ | ▼ • Reply • Share ›

nagarjuna ↗ Yuvaraj Velmayil • 6 months ago

Yes, correct

▲ | ▼ • Reply • Share ›

GeeksforGeeks Mod ↗ Yuvaraj Velmayil • 8 months ago

Youvraj, There was a typo in second example. We have made fun() pri

▲ | ▼ • Reply • Share ›

Yuvaraj Velmayil ↗ GeeksforGeeks • 8 months ago

When a method is re-declared in a sub class with the same sig
member, the compiler will not throw any error(As the private me
the sub classes). Rather, in sub class, it will have its one privat

▲ | ▼ • Reply • Share ›

Noob • 8 months ago

Good explanation, does it mean that use of final keyword with private methods

1 • Reply • Share ›

Ajay Noob • 8 months ago

Yes it is redundant. As you have already defined the scope of the meth
any of the subclass for extension (overriding). Hence making a private

• Reply • Share ›

Yuvaraj Velmayil Noob • 8 months ago

Both serves different purpose. Final is to prevent the members/classes
visible(depending upon the access modifiers). Private is an access mo
from outside the class. It will not be available for even override.

• Reply • Share ›

Ruthong Yuvaraj Velmayil • a month ago

Thats very well said Yuvaraj. Both final and private have its owr

• Reply • Share ›

chandra prakash Noob • 8 months ago

Both final and private do the same thing i,e avoids inheriting methods h
or private alone is doing the same thing then no need of using both altc

• Reply • Share ›

Kartik Noob • 8 months ago

Yes, it is redundant. See http://www.geeksforgeeks.org/p...

• Reply • Share ›