What happens when more restrictive access is given to a derived class method in Java?

In Java, it is compiler error to give more restrictive access to a derived class function which overrides a base class function. For example, if there is a function *public void foo()* in base class and if it is overridden in derived class, then access specifier for *foo()* cannot be anything other than public in derived class. If *foo()* is private function in base class, then access specifier for it can be anything in derived class.

Consider the following two programs. Program 1 fails in compilation and program 2 works fine.

Program 1

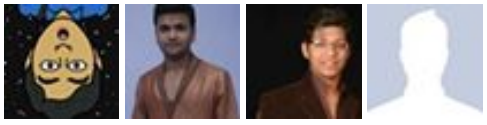
```
// file name: Main.java
class Base {
    public void foo() {}
}

class Derived extends Base {
    private void foo() {} // compiler error
}

public class Main {
    public static void main(String args[]) {
        Derived d = new Derived();
    }
}
```

Program 2

```
// file name: Main.java
class Base {
    private void foo() {}
```

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
}

class Derived extends Base {
    public void foo() {} // works fine
}

public class Main {
    public static void main(String args[]) {
        Derived d = new Derived();
    }
}
```

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?
- Can we Overload or Override static methods in java ?
- Static class in Java

Popular Posts

- Do we need forward declarations in Java?
- Checked vs Unchecked Exceptions in Java
- Bitwise right shift operators in Java

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

5 Comments GeeksforGeeks

Sort by Newest

alfasin • 8 months ago

It will compile alright, but that doesn't mean that it work...

:)

Let's dive a bit deeper:

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

class Derived extends Base {
  public void foo() {
    System.out.println(" Derived ");
  } // works fine
}
```

```
public class Child {
```

see more

• Reply • Share ›

Swaroop • 3 years ago

program-2:
Here when you extended the base class to derived class, actually the private f
hence the foo() method which is present in the derived class is totally a new m
I feel here we are not bothered about the access specifier concept.
Please correct me if i am wrong?

1 • Reply • Share ›

kartik ➦ Swaroop • 3 years ago

No, it's not like that. You can try with protected. Following program alsc

```
// file name: Main.java
class Base {
    protected void foo() {}
}

class Derived extends Base {
    public void foo() {} // works fine
}

public class Main {
    public static void main(String args[]) {
        Derived d = new Derived();
    }
}
```

705

Subscribe

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

Shekhu • 3 years ago

Can we change the access specifier in C++? If yes, the what is the rule in C+

jia ➦ Shekhu • 3 years ago

@Swaroop that is not case..
fact is that java does not allow to make weaker access privileges of a r
privileges are increasing..
prog1.....
import java.io.*;
// file name: Main.java
class Base {
protected void foo() {}
}

class Derived extends Base {
public void foo() {} // compiler error
}

public class Main {
public static void main(String args[]) {
Derived d = new Derived();
}
}