Comparison of Inheritance in C++ and Java

The purpose of inheritance is same in C++ and Java. Inheritance is used in both languages for reusing code and/or creating is-a relationship. There are following differences in the way both languages provide support for inheritance.

1) In Java, all classes inherit from the Object class directly or indirectly. Therefore, there is always a single inheritance tree of classes in Java, and Object class is root of the tree. In Java, if we create a class that doesn't inherit from any class then it automatically inherits from Object class . In C++, there is forest of classes; when we create a class that doesn't inherit from anything, we create a new tree in forest.

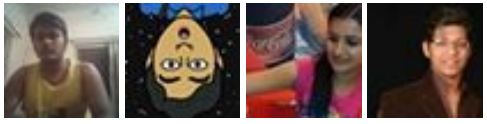
Following Java example shows that Test class automatically inherits from Object class.

```
class Test {
    // members of test
}
class Main {
  public static void main(String[] args) {
    Test t = new Test();
    System.out.println("t is instanceof Object: " + (t instanceof Obje
  }
}
```

Output:

```
t is instanceof Object: true
```

2) In Java, members of the grandparent class are not directly accessible. See this G-Fact for more details.



3) The meaning of protected member access specifier is somewhat different in Java. In Java, protected members of a class “A” are accessible in other class “B” of same package, even if B doesn’t inherit from A (they both have to be in the same package). For example, in the following program, protected members of A are accessible in B.

```
// filename B.java
class A {
    protected int x = 10, y = 20;
}

class B {
    public static void main(String args[]) {
        A a = new A();
        System.out.println(a.x + " " + a.y);
    }
}
```

4) Java uses *extends* keyword for inheritence. Unlike C++, Java doesn’t provide an inheritance specifier like public, protected or private. Therefore, we cannot change the protection level of members of base class in Java, if some data member is public or protected in base class then it remains public or protected in derived class. Like C++, private members of base class are not accessible in derived class.
Unlike C++, in Java, we don’t have to remember those rules of inheritance which are combination of base class access specifier and inheritance specifier.

5) In Java, methods are virtual by default. In C++, we explicitly use virtual keyword. See this G-Fact for more details.

6) Java uses a separte keyword *interface* for interfaces, and *abstract* keyword for abstract classes and abstract functions.

Following is a Java abstract class example.

```
// An abstract class example
abstract class myAbstractClass {

   // An abstract method
```


Popular Posts

```
    abstract void myAbstractFun();

    // A normal method
    void fun() {
        System.out.println("Inside My fun");
    }
}

public class myClass extends myAbstractClass {
    public void myAbstractFun() {
        System.out.println("Inside My fun");
    }
}
```

Following is a Java interface example

```
// An interface example
public interface myInterface {
    // myAbstractFun() is public and abstract, even if we don't use the
    void myAbstractFun();  // is same as public abstract void myAbstrac
}

// Note the implements keyword also.
public class myClass implements myInterface {
    public void myAbstractFun() {
        System.out.println("Inside My fun");
    }
}
```

7) Unlike C++, Java doesn't support multiple inheritance. A class cannot inherit from more than one class. A class can implement multiple interfaces though.

8) In C++, default constructor of parent class is automatically called, but if we want to call parametrized constructor of a parent class, we must use Initializer list. Like C++, default constructor of the parent class is automatically called in Java, but if we want to call parametrized constructor then we must use super to call the parent constructor. See following Java example.

```
package main;

class Base {
    private int b;
    Base(int x) {
```

```
        b = x;
        System.out.println("Base constructor called");
    }
}

class Derived extends Base {
    private int d;
    Derived(int x, int y) {
        // Calling parent class parameterized constructor
        // Call to parent constructor must be the first line in a Deri
        super(x);
        d = y;
        System.out.println("Derived constructor called");
    }
}

class Main{
    public static void main(String[] args) {
      Derived obj = new Derived(1, 2);
    }
}
```

Output:

```
Base constructor called
Derived constructor called
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

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

Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

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

numbers also. We...

Find subarray with given sum · 2 hours ago

Curious_Georgie • 3 months ago

You can add one more point about the Garbage Collector facility in Java.

  • Reply • Share ›

GeeksforGeeks • 10 months ago

Please take a closer look 3rd point is about protected members, not private m

  • Reply • Share ›

Vijay Prakash • 10 months ago

3 eg is not right we can not access a private member from another class.

  • Reply • Share ›

Shivprasad Shettar • a year ago

"Java doesn’t provide an inheritance specifier like public, protected or private. T
protection level of members of base class in Java, if some data member is pu
remains public or protected in derived class".
I&#039m not sure if this is right. AFAIK we can&#039t make the protection leve
restrictive. But something like Protected to Public is allowed.

  • Reply • Share ›

rahul • 2 years ago

Java doesn't support multiple inheritance (MI). They say that there are some d
what are the issues with MI?

  • Reply • Share ›

slvrhwk • 2 years ago

Nice post. Thanks.

But in the 6th point, when myClass extends an abstract class, you also need t
myAbstractFun method since it is an abstract method and does not have a de

• Reply • Share ›

GeeksforGeeks ➚ slvrhwk • 2 years ago

@slvrhwk: Thanks for pointing out the typo. We have corrected the fun

• Reply • Share ›