Output of Java Program | Set 2

Predict the output of following Java programs.

Question 1

```
package main;

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

class Derived extends Base {
    public void Print() {
        System.out.println("Derived");
    }
}

class Main{
    public static void DoPrint( Base o ) {
        o.Print();
    }
    public static void main(String[] args) {
        Base x = new Base();
        Base y = new Derived();
        Derived z = new Derived();
        DoPrint(x);
        DoPrint(y);
        DoPrint(z);
    }
}
```

Output:

```
Base
Derived
Derived
```

Predicting the first line of output is easy. We create an object of type Base and call DoPrint(). DoPrint calls the print function and we get the first line.

DoPrint(y) causes second line of output. Like C++, assigning a derived class reference to a base class reference is allowed in Java. Therefore, the expression Base y = new Derived() is a valid statement in Java. In DoPrint(), o starts referring to the same object as referred by y. Also, unlike C++, functions are virtual by default in Java. So, when we call o.print(), the print() method of Derived class is called due to run time polymorphism present by default in Java.

DoPrint(z) causes third line of output, we pass a reference of Derived type and the print() method of Derived class is called again. The point to note here is: unlike C++, object slicing doesn't happen in Java. Because non-primitive types are always assigned by reference.

Question 2

```
package main;

// filename Main.java
class Point {
    protected int x, y;

    public Point(int _x, int _y) {
        x = _x;
        y = _y;
    }
}

public class Main {
    public static void main(String args[]) {
      Point p = new Point();
      System.out.println("x = " + p.x + ", y = " + p.y);
    }
}
```


Popular Posts

Output:
Compiler Error
In the above program, there are no access permission issues because the Test and Main are in same package and protected members of a class can be accessed in other classes of same package. The problem with the code is: there is not default constructor in Point. Like C++, if we write our own parametrized constructor then Java compiler doesn't create the default constructor. So there are following two changes to Point class that can fix the above program.
1) Remove the parametrized constructor.
2) Add a constructor without any parameter.
Java doesn't support default arguments, so that is not an option.

Please write comments if you find any of the answers/explanations incorrect, or want to share more information about the topics discussed above.

Related Tpoics:

- Output of Java Program | Set 4
- Output of Java Program | Set 3
- Output of C++ Program | Set 18

- Output of C++ Program | Set 18
- Output of C++ Program | Set 17
- Output of C++ Program | Set 16
- Output of C++ Program | Set 15
- Output of C++ Program | Set 14
- Output of C++ Program | Set 13

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

5 Comments GeeksforGeeks

Sort by Newest

mani • 11 months ago
Point p=new Point()
There also u will find error because there is no default unparametrized constru

Reply • Share ›

Sandeep • a year ago
Base y = new Derived();

is also called as Dynamic Method Dispatch

1 Reply • Share ›

mani • 2 years ago
package javaapplication2;

import java.util.Date;

Recent Comments

Abhi You live US or India?
Google (Mountain View) interview · 33 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

```
import java.lang.*;

public class DateApp {
public static void main(String args[]) {
// try{
Date today = new Date();
System.out.println("Date Format is "+ today);
// }
// catch(Exception e)
// {
System.out.println("run time Error.....");
// }
}
}
```

1 ▲ | ▼ • Reply • Share ›

giri • 3 years ago

Run time polymorphism is not present by default in C++ (virtual keyword has t polymorphism causes overhead and C++ cares about efficiency.

▲ | ▼ • Reply • Share ›

kartik ➚ giri • 3 years ago

The extra overhead can also be avoided in Java. final keyword is used

▲ | ▼ • Reply • Share ›