

## What's faster: method call or public access?

2005-10-13 11:57:11

Several times when I was writing java classes, I wondered, what would lead to a better performance: calls to a getter method returning some private attributes or declaring the attribute public and accessing it directly.

From the viewpoint of object oriented programming creating getters and setters to a private attribute is always better. It encapsulates better and you have the chance to monitor changes to attributes through the setters, for example to implement an observer pattern. On the other side, calling methods should lead to programmatic overhead, like heap reservation, parameter passing etc. Classes like the `java.awt.Point` or `java.awt.Dimension` allow direct access to their public declared values. So I wanted to testify my assumptions and wrote a little benchmark:

```
public class X{ public double x; public X(double xx){ x=xx; } } public class Y{ private double x; public
Y(double xx){ x=xx; } public double getX(){ return x; } } public class Test { public static void main(String[]
args) { test(1000); testStat(100000000, 10); testStat(1000000000, 10); testStat(10000000000L, 10); }
public static void testStat(long amount, int statamount){ long [] sum = new long[2]; long [] single; for (int
i=0; i<statamount; i++){ single = test(amount); sum[0]+=single[0]; sum[1]+=single[1]; } sum[0] /=
statamount; sum[1] /= statamount; System.out.println("Avg. results: " +sum[0]+"; ms
"+sum[1]+"; ms"); } public static long [] test(long amount){ long [] result = new long[2];
System.out.println(); System.out.println("Testing "+Long.toString(amount)+"
times"); long starttime, endtime; X x = new X(100); Y y = new Y(100); double w;
starttime=System.currentTimeMillis(); for (long i=0; i<amount; i++) w = x.x;
endtime=System.currentTimeMillis(); result[0] = endtime-starttime; System.out.println(result[0]+";
ms"); starttime=System.currentTimeMillis(); for (long i=0; i<amount; i++) w = y.getX();
endtime=System.currentTimeMillis(); result[1] = endtime-starttime; System.out.println(result[1]+";
ms"); return result; } }
```

This benchmark lead to the following results:

- avg. results for 10<sup>9</sup> cycles  
Class X: 4545 ms Class Y: 4609 ms
- avg. results for 10<sup>10</sup> cycles  
Class X: 45454 ms Class Y: 45945 ms

Since this results are not 100% representative (I had several tasks in the background), I will verify them on other machines and constellations. (If you like, do so, too and post it here.)

But at all it seems, that the performance difference between method invocations and direct public access is not that big. At least small enough to prefer the object oriented approach through getters.