In Java, when should I use "Object o" instead of generics?

Asked 16 years ago Modified 16 years ago Viewed 1k times

11

For example, I could write either of these:

```
class example <T>
{
    ...

    public void insert (T data)
    {
        ...
    }
}
```

or

```
class example
{
    ...

    public void insert (Object o)
    {
        ...
    }
}
```

Is there a signficant difference between the 2 in terms of performance? With generics I could restrict the type of the parameter and with the second approach I guess it wouldn't be necessary to define the type of the object as it is created.

Also, with the second approach I could basically insert anything into the class, right? Whereas with generics every element in the class would be of the same type.

Anything else I'm missing?

java generics

Share Improve this question Follow

asked Dec 10, 2008 at 2:48

master chief

5 Answers

Sorted by: Highest score (default)

9

The only reason to write the latter is if you must target an earlier JVM. Generics are implemented by type-erasure, so they have no runtime impact - only added compile time checking which will improve your code.

Of course if you need a collection which holds any old object, or a mix of several which don't have a common superclass, you need the plain Object variation (but then your class can still be generic and instantiated with new ...<Object>).

Share Improve this answer Follow

edited Dec 10, 2008 at 3:36

answered Dec 10, 2008 at 2:54

L. Cornelius Dol
63.9k ● 26 ● 141 ● 189

> I think generics are dum dum dum. They simply equate back to Object, so why not just use Object. I am still going to have to use cast, eg if I pass an array to a generic and print it. C++ templates are light years ahead. Generics are just nonsense. – user9599745 Aug 2, 2019 at 23:29

6

I think you pretty much nailed it. There is no performance difference. Generics are rationalized away (Type Erasure) when the code is compiled, and don't exist anymore at runtime. They just add casts when needed and do type-checking as you stated. Neal Gafter wrote a nice overview of how they work, of the current problems with Generics and how they could be solved in the next version of Java: http://gafter.blogspot.com/2006/11/reified-generics-for-java.html

Share Improve this answer Follow

edited Dec 10, 2008 at 3:19

answered Dec 10, 2008 at 2:55

Johannes Schaub - litb
506k ● 131 ● 917 ● 1.2k

3

There shouldn't be a performance difference.

However, Java does not offer parameter variance, so there are situations where you will be overriding pre-generics functions such as equals, compareTo, etc. where you will have to use Objects.

Share Improve this answer Follow

answered Dec 10, 2008 at 3:06

Uri
89.6k ● 51 ● 226 ● 322

> Yeap. This would be the only reason. If you want to mix up different types of data using generics won't do. For instance mixing ... mmhh ... Strings and Employees in the same place (

why would you do that, I don't know) – OscarRyz Dec 10, 2008 at 3:09

0

Some of the encounters where I had to use 'Object' instead of Generics were those of compulsion than of a choice. When working with pre-generic code or libraries built around pre-generic api, one has little choice. Dynamic proxies for example, Proxy.newProxy() returns Object type. Passing generic context (where a context can be anything) is another instance. Some of my friends argue that are as good as no-generics. As far as performance is concerned, there shouldn't be any overhead, considering type erasure.

Share Improve this answer Follow

answered Dec 10, 2008 at 9:37

questzen

3,287 ● 20 ● 21

0

Regarding performance, i agree with the people above.

Regarding this point of yours

"Also, with the second approach I could basically insert anything into the class, right? Whereas with generics every element in the class would be of the same type."

One more advantage of generics is there is a type check for assignment of the example instance itself.

Say for example you had an Example e1 of and another Example e2 of , type safety would be maintained and you would never be able to do e1=e2;

while with the object example, that would be possible.

Share Improve this answer Follow

answered Dec 10, 2008 at 11:32

Devesh Rao

81 ● 2