5 Different Ways to Create Objects in Java

A list of five ways to create objects in Java, how they interact with constructors, and an example of how to utilize all of these methods.

by Naresh Joshi MVB · Jul. 12, 16 · Java Zone · Opinion

Being Java developers, we usually create lots of objects daily, but we always use dependency management systems e.g. Spring to create these objects. However, there are more ways to create objects, which we will study in this article.

There are five total ways to create objects in Java, which are explained below with their examples followed by bytecode of the line which is creating the object.

Using new keyword	} → constructor gets called
Using newInstance() method of Class class	} → constructor gets called
Using newInstance() method of Constructor class	} → constructor gets called
Using clone() method	} → no constructor call
Using deserialization	} → no constructor call

If you will execute program given in end, you will see method 1, 2, 3 uses the constructor to create the object while 4, 5 doesn't call the constructor to create the object.

1. Using new keywords

It is the most common and regular way to create an object and a very simple one also. By using this method we can call whichever constructor we want to call (no-arg constructor as well as parameterized).

2. Using newInstance() method of Class class

We can also use the newInstance() method of a Class class to create an object. This newInstance() method calls the no-arg constructor to create the object.

We can create an object by newInstance() in the following way:

Or

4. Using newInstance() method of Constructor class

Similar to the newInstance() method of Class class, There is one newInstance() method in the java.lang.reflect.Constructor class which we can use to create objects. We can also call parameterized constructor, and private constructor by using this newInstance() method.

Both newInstance() methods are known as reflective ways to create objects. In fact newInstance() method of Class class internally uses newInstance() method of Constructor class. That's why the later one is preferred and also used by different frameworks like Spring, Hibernate, Struts etc. To know differences between both newInstance() methods read Creating objects through Reflection in Java with Example.

4. Using clone() method:

Whenever we call clone() on any object, the JVM actually creates a new object for us and copies all content of the previous object into it. Creating an object using the clone method does not invoke any constructor.

To use clone() method on an object we need to implement Cloneable and define the clone() method in it.

Java cloning is the most debatable topic in Java community and it surely does have its drawbacks but it is still the most popular and easy way of creating a copy of any object until that object is full filling mandatory conditions of Java cloning. I have covered cloning in details in a 3 article long Java Cloning Series which includes (Java Cloning And Types Of Cloning (Shallow And Deep) In Details With Example, Java Cloning - Copy Constructor Versus Cloning, Java Cloning - Even Copy Constructors Are Not Sufficient), go ahead and read them if you want to know more about cloning.

5. Using deserialization:

Whenever we serialize and deserialize an object, the JVM creates a separate object for us. In deserialization, the JVM doesn't use any constructor to create the object.

To deserialize an object we need to implement a Serializable interface in our class.

As we can see in the above bytecode snippets, all 4 methods are called and get converted to invokevirtual (object creation is directly handled by these methods) except the first one, which got converted to two calls: one is new and other is invokespecial (call to constructor).

Example

Let's consider an Employee class for which we are going to create the objects:

In the below Java program we are going to create Employee objects in all 5 ways. You can also find the source code at GitHub.

In the below Java program we are going to create Employee objects in all 5 ways. You can also find the source code at GitHub.

This program will give the following output:

Topics: JAVA , CORE JAVA , REFLECTION API

Java Partner Resources