**Different ways to create an object in Java**

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 the new keyword | The constructor gets called |
| Using [newInstance()](https://docs.oracle.com/javase/8/docs/api/java/lang/Class.html" \l "newInstance--" \t "_blank) method of Class class | The constructor gets called |
| Using [newInstance()](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Constructor.html" \l "newInstance-java.lang.Object...-" \t "_blank) method of Constructor class | The constructor gets called |
| Using clone() method | No constructor call |
| Using deserialization | No constructor call |

You must be aware of creating an object of a class by using the new keyword, but that is not the only way to create an Object.

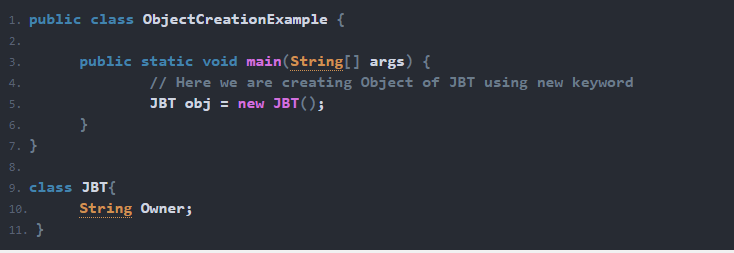
There are several other ways to create an object of a class :

* Using new keyword
* Using new Instance (Reflection)
* Using Clone
* Using Deserialization
* Using ClassLoader

## Using new Keyword

Using a new keyword is the most basic way to create an object. ***new***keyword can be used to create an object of a class.

Employee emp1 = new Employee();



## Using New Instance (Reflection)

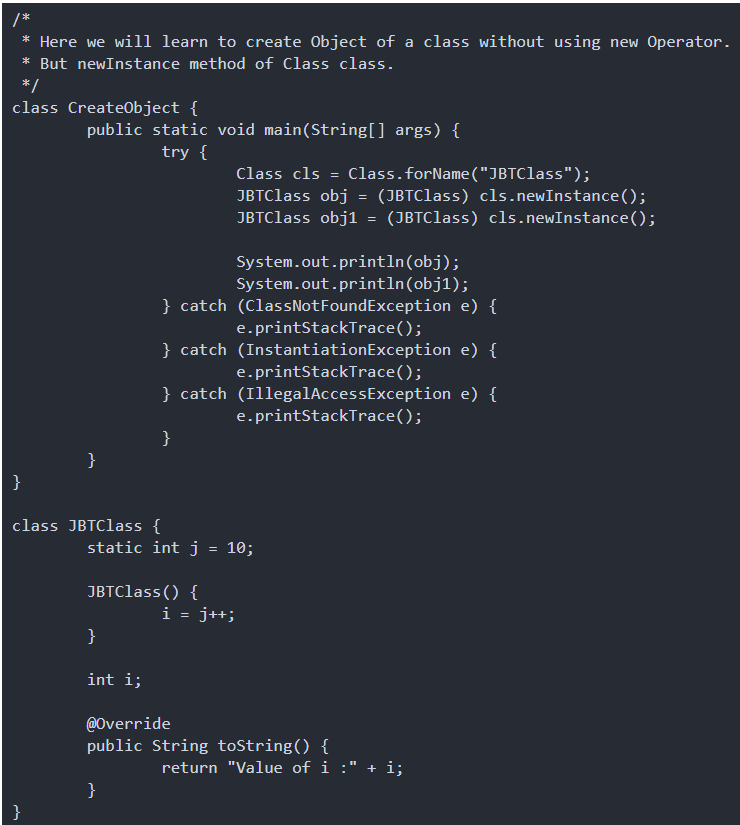
Have you ever tried to connect to a DB using the JDBC driver in Java? If your answer is yes, then you must have seen “***Class.forName***“. We can also use it to create the object of a class. **Class.forName**actually loads the class in java but doesn’t create any object. To create an object, you have to use the ***newInstance***method of the Class class.

**If you want to create the object in this way class needs to have a public default constructor.**

Employee emp2 = (Employee) Class.forName("org.programming.mitra.exercises.Employee").newInstance();

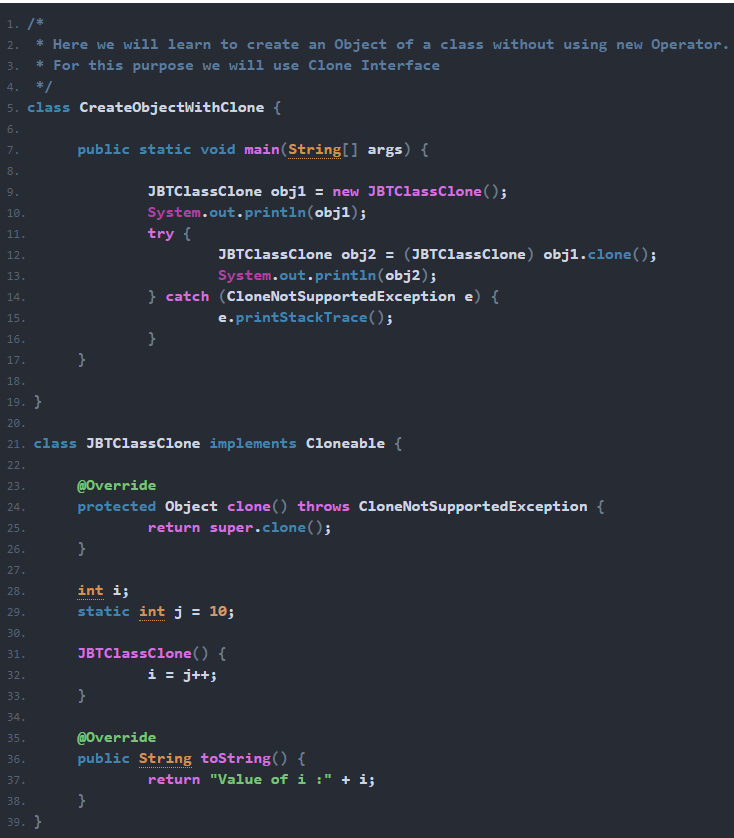
Or

Employee emp2 = Employee.class.newInstance();



## Using Clone

We can also use Clone() method to create a copy of an existing object.



**Additional notes on Clone**

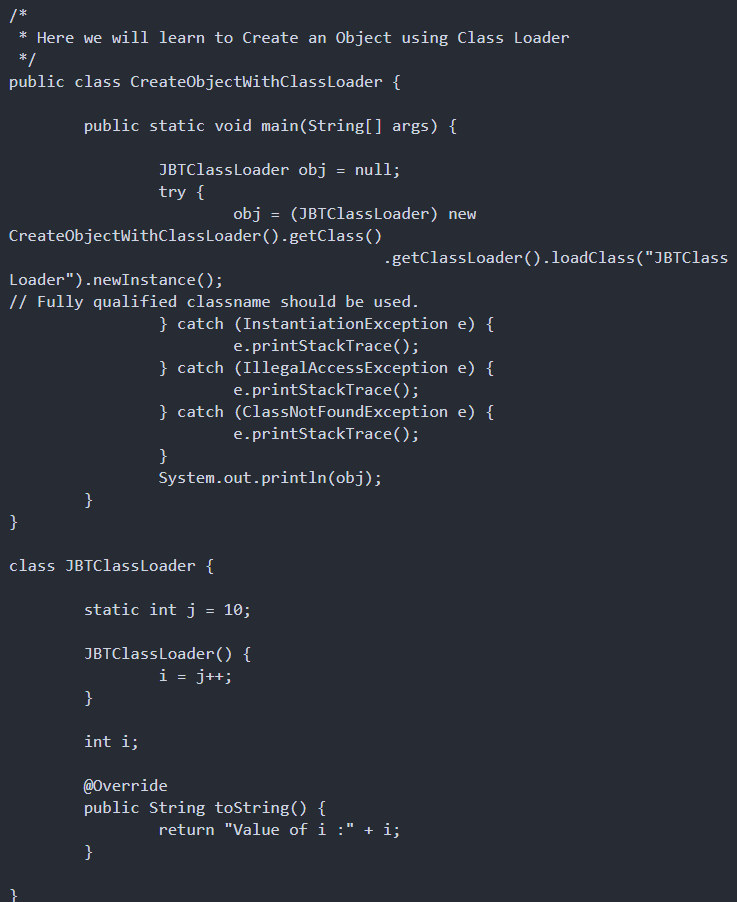
* Here we are creating the clone of an existing Object and not any new Object.
* The clone method is declared protected in Object class. So it can be accessed only in a subclass or in the same package. That is the reason why it has been overridden here in class.
* A class needs to implement the Cloneable interface otherwise it will throw CloneNotSupportedException.

## Using Object Deserialization

Object deserialization can also be used to create an Object. It produces the opposite of serializing an Object.

## Using ClassLoader

We can also use ClassLoader to create the object of a class. This way is much the same as***Class.forName*** option.



**Using newInstance() method of Constructor class :** This is similar to the newInstance() method of a class. There is one newInstance() method in the **java.lang.reflect.Constructor** class which we can use to create objects. It 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 internally uses newInstance() method of Constructor class.

Constructor<Employee> constructor = Employee.class.getConstructor();

Employee emp3 = constructor.newInstance();

public class ReflectionExample

{

    private String name;

    ReflectionExample()

    {

    }

    public void setName(String name)

    {

        this.name = name;

    }

    public static void main(String[] args)

    {

        try

        {

            Constructor<ReflectionExample> constructor

                = ReflectionExample.class.getDeclaredConstructor();

            ReflectionExample r = constructor.newInstance();

            r.setName("GeeksForGeeks");

            System.out.println(r.name);

        }

        catch (Exception e)

        {

            e.printStackTrace();

        }

    }

}