Fast I/O in Java in Competitive Programming
Command Line arguments
Explore More...
Arrays
Arrays in Java
Default array values in Java
Compare two arrays
Final Arrays & Jagged Arrays
Array IndexOutofbounds Exception
Explore More...
Strings
String Class in Java
StringBuffer , StringTokenizer & StringJoiner
Initialize and Compare Strings
String vs StringBuilder vs StringBuffer
Integer to String & String to Integer
Search, Reverse and Split()
Explore More...
OOP in Java
Classes and Objects in Java
Different ways to create objects

Access Modifiers in Java
Object class in Java
Encapsulation & Inheritance
Method Overloading & Overriding
Explore More...
Constructors
Constructors & Constructor Chaining
Constructor Overloading
Private Constructors and Singleton Classes
Explore More...
Methods
Parameter Passing
Returning Multiple Values
Private and Final Methods
Default Methods
Explore More...
Exception Handling
Exceptions & Types of Exceptions
Flow control in try-catch & Multicatch
throw and throws

Explore More...

Multithreading

Multithreading

Lifecycle and States of a Thread

Main Thread

Synchronization

Inter-thread Communication & Java Concurrency

Explore More...

File Handling

File Class

File Permissions

Different ways of Reading a text file

Delete a File

Explore more...

Garbage Collection

Garbage Collection

Mark and Sweep

Explore more...

Java Packages

Packages

Java.io Package

Java.lang package

Different ways to create objects in Java

As you all know, in Java, a class provides the blueprint for objects, you create an object from a class. There are many different ways to create objects in Java.

Following are some ways in which you can create objects in Java:

1) Using new Keyword : Using new keyword is the most basic way to create an object. This is the most common way to create an object in java. Almost 99% of objects are created in this way. By using this method we can call any constructor we want to call (no argument or parameterized constructors).

```
// Java program to illustrate creation of Object
// using new keyword
public class NewKeywordExample
{
    String name = "GeeksForGeeks";
    public static void main(String[] args)
    {
        // Here we are creating Object of
        // NewKeywordExample using new keyword
        NewKeywordExample obj = new NewKeywordExample();
        System.out.println(obj.name);
    }
}
```

Run on IDE

Output:

```
GeeksForGeeks
```

2) Using New Instance : If we know the name of the class & if it has a public default constructor we can create an object –**Class.forName**. We can 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 of the Class you have to use the new Instance Method of the Class.

```
// Java program to illustrate creation of Object
// using new Instance
public class NewInstanceExample
{
    String name = "GeeksForGeeks";
    public static void main(String[] args)
    {
        try
        {
            Class cls = Class.forName("NewInstanceExample");
            NewInstanceExample obj =
                    (NewInstanceExample) cls.newInstance();
            System.out.println(obj.name);
        }
        catch (ClassNotFoundException e)
        {
            e.printStackTrace();
        }
        catch (InstantiationException e)
        {
            e.printStackTrace();
        }
        catch (IllegalAccessException e)
        {
            e.printStackTrace();
        }
    }
}
```

Run on IDE

Output:

```
GeeksForGeeks
```

3) Using clone() method: Whenever clone() is called on any object, the JVM actually creates a new object 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 program to illustrate creation of Object
// using clone() method
public class CloneExample implements Cloneable
{
    @Override
    protected Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }
    String name = "GeeksForGeeks";

    public static void main(String[] args)
    {
        CloneExample obj1 = new CloneExample();
        try
        {
            CloneExample obj2 = (CloneExample) obj1.clone();
            System.out.println(obj2.name);
        }
        catch (CloneNotSupportedException e)
        {
            e.printStackTrace();
        }
    }
}
```

Run on IDE

Output:

```
GeeksForGeeks
```

Note :

- Here we are creating the clone of an existing Object and not any new Object.
- Class need to implement Cloneable Interface otherwise it will throw **CloneNotSupportedException**.

4) Using deserialization : Whenever we serialize and then deserialize an object, JVM creates a separate object. In **deserialization**, JVM doesn't use any constructor to create the object. To deserialize an object we need to implement the Serializable interface in the class.

Serializing an Object :

```
// Java program to illustrate Serializing
// an Object.
import java.io.*;

class DeserializationExample implements Serializable
{
    private String name;
    DeserializationExample(String name)
    {
        this.name = name;
    }

    public static void main(String[] args)
    {
        try
        {
            DeserializationExample d =
                    new DeserializationExample("GeeksForGeeks");
            FileOutputStream f = new FileOutputStream("file.txt");
            ObjectOutputStream oos = new ObjectOutputStream(f);
            oos.writeObject(d);
            oos.close();
            f.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Run on IDE

Object of DeserializationExample class is serialized using writeObject() method and written to file.txt file.

Deserialization of Object :

```
// Java program to illustrate creation of Object
// using Deserialization.
import java.io.*;

public class DeserializationExample
{
    public static void main(String[] args)
    {
        try
        {
            DeserializationExample d;
            FileInputStream f = new FileInputStream("file.txt");
            ObjectOutputStream oos = new ObjectOutputStream(f);
            d = (DeserializationExample)oos.readObject();
```

```
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        System.out.println(d.name);
    }
}
```

Run on IDE

Output:

```
GeeksForGeeks
```

5) 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.

```
// Java program to illustrate creation of Object
// using newInstance() method of Constructor class
import java.lang.reflect.*;

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();
        }
```

```
    }
}
```

Run on IDE

Output:

```
GeeksForGeeks
```

This article is contributed by **Saket Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Practice Tags : Java

Article Tags : Java

Login to Improve this Article

Recommended Posts:

How to swap or exchange objects in Java?
Inheritance in Java
How are Java objects stored in memory?
Classes and Objects in Java
Encapsulation in Java
Android | Starting with first app/android project
Life Cycle of a Servlet
BigDecimal doubleValue() Method in Java

BigDecimal unscaledValue() in Java
BigDecimal ulp() Method in Java

(Login to Rate)

3.2 Average Difficulty : **3.2/5.0**
Based on **31** vote(s)

Basic | Easy | Medium | Hard | Expert

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!