

Java Runtime class

Java Runtime class is used to interact with java runtime environment. Java Runtime class provides methods to execute a process, invoke GC, get total and free memory etc. There is only one instance of java.lang.Runtime class is available for one java application.

The **Runtime.getRuntime()** method returns the singleton instance of Runtime class.

Important methods of Java Runtime class

No.	Method	Description
1)	public static Runtime getRuntime()	returns the instance of Runtime class.
2)	public void exit(int status)	terminates the current virtual machine.
3)	public void addShutdownHook(Thread hook)	registers new hook thread.
4)	public Process exec(String command)throws IOException	executes given command in a separate process.
5)	public int availableProcessors()	returns no. of available processors.
6)	public long freeMemory()	returns amount of free memory in JVM.
7)	public long totalMemory()	returns amount of total memory in JVM.

Java Runtime exec() method

```
public class Runtime1{
  public static void main(String args[])throws Exception{
   Runtime.getRuntime().exec("notepad");//will open a new notepad
}
}
```

How to shutdown system in Java

You can use *shutdown -s* command to shutdown system. For windows OS, you need to provide full path of shutdown command e.g. c:\\Windows\\System32\\shutdown.

Here you can use -s switch to shutdown system, -r switch to restart system and -t switch to specify time delay.

```
public class Runtime2{
  public static void main(String args[])throws Exception{
  Runtime.getRuntime().exec("shutdown -s -t 0");
  }
}
```

How to shutdown windows system in Java

```
public class Runtime2{
  public static void main(String args[])throws Exception{
   Runtime.getRuntime().exec("c:\\Windows\\System32\\shutdown -s -t 0");
}
```

How to restart system in Java

```
public class Runtime3{
  public static void main(String args[])throws Exception{
   Runtime.getRuntime().exec("shutdown -r -t 0");
}
}
```

Java Runtime availableProcessors()

```
public class Runtime4{
  public static void main(String args[])throws Exception{
    System.out.println(Runtime.getRuntime().availableProcessors());
}
}
```

Java Runtime freeMemory() and totalMemory() method

In the given program, after creating 10000 instance, free memory will be less than the previous free memory. But after gc() call, you will get more free memory.

```
public class MemoryTest{
```

```
public static void main(String args[])throws Exception{
   Runtime r=Runtime.getRuntime();
   System.out.println("Total Memory: "+r.totalMemory());
   System.out.println("Free Memory: "+r.freeMemory());

for(int i=0;i<10000;i++){
   new MemoryTest();
  }
   System.out.println("After creating 10000 instance, Free Memory: "+r.freeMemory());
   System.gc();
   System.out.println("After gc(), Free Memory: "+r.freeMemory());
}
</pre>
```

```
Total Memory: 100139008

Free Memory: 99474824

After creating 10000 instance, Free Memory: 99310552

After gc(), Free Memory: 100182832
```

← Prev

 $Next \rightarrow$

Youtube For Videos Join Our Youtube Channel: Join Now

Feedback

Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

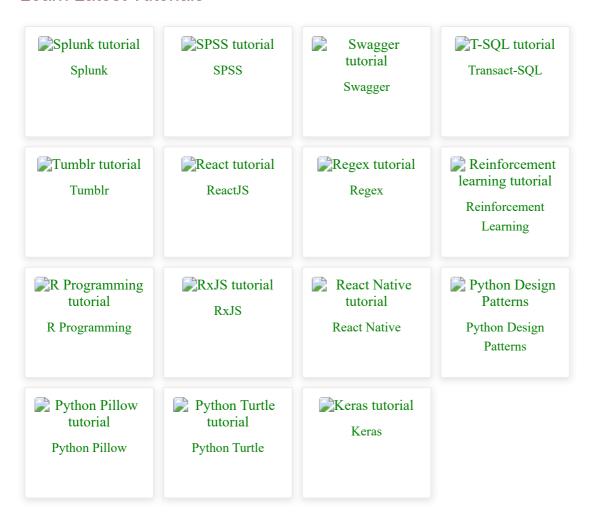








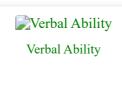
Learn Latest Tutorials



Preparation



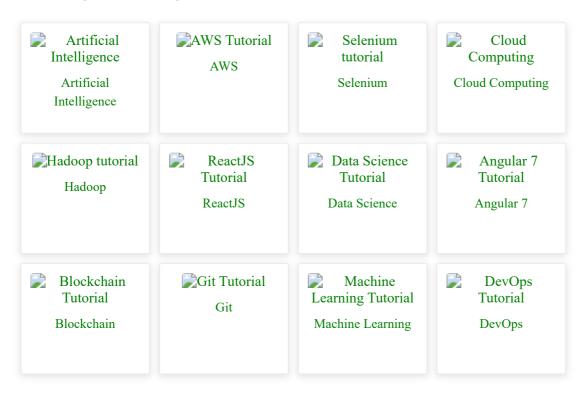








Trending Technologies



B.Tech / MCA



