# JVM Shutdown Hook in Java

- Difficulty Level : Medium
- Last Updated : 12 Nov, 2021

- Read
- Discuss

Shutdown Hooks are a special construct that allows developers to plug in a piece of code to be executed when the JVM is shutting down. This comes in handy in cases where we need to do special clean up operations in case the VM is shutting down.
Handling this using the general constructs such as making sure that we call a special procedure before the application exits (calling System.exit(0) ) will not work for situations where the VM is shutting down due to an external reason (ex. kill request from O/S), or due to a resource problem (out of memory). As we will see soon, shutdown hooks solve this problem easily, by allowing us to provide an arbitrary code block, which will be called by the JVM when it is shutting down.
From the surface, using a shutdown hook is downright straightforward. All we have to do is simply write a class that extends the java.lang.Thread class, and provide the logic that we want to perform when the VM is shutting down, inside the public void run() method. Then we register an instance of this class as a shutdown hook to the VM by calling Runtime.getRuntime().addShutdownHook(Thread) method. If you need to remove a previously registered shutdown hook, the Runtime class provides the removeShutdownHook(Thread) method as well.
**Example 1 (Anonymous inner class):**

```
public class ShutDownHook
{
  public static void main(String[] args)
  {

    Runtime.getRuntime().addShutdownHook(new Thread()
    {
      public void run()
      {
        System.out.println("Shutdown Hook is running !");
      }
    });
    System.out.println("Application Terminating ...");
  }
}
```

When we run the above code, you will see that the shutdown hook is getting called by the JVM when it finishes the execution of the main method.
Output:

```
Application Terminating ...
Shutdown Hook is running !
```

## Example 2

```
class ThreadChild extends Thread {

    public void run() {

        System.out.println("In clean up code");
        System.out.println("In shutdown hook");
    }
}

class Demo {

    public static void main(String[] args) {

        Runtime current = Runtime.getRuntime();
        current.addShutdownHook(new ThreadChild());

        for(int i = 1; i <= 10; i++)
            System.out.println("2 X " + i + " = " + 2 * i);
    }
}
```

Output:

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
In clean up code
In shutdown hook
```

Simple right? Yes, it is.
While it is pretty simple to write a shutdown hook, one needs to know the internals behind the shutdown hooks to make use of those properly. Therefore, in this article, we will be exploring some of the ‘gotchas’ behind the shutdown hook design.

**1. Shutdown Hooks may not be executed in some cases!**

The first thing to keep in mind is that it is not guaranteed that shutdown hooks will always run. If the JVM crashes due to some internal error, then it might crash down without having a chance to execute a single instruction. Also, if the O/S gives a SIGKILL (http://en.wikipedia.org/wiki/SIGKILL) signal (kill -9 in Unix/Linux) or TerminateProcess (Windows), then the application is required to terminate immediately without doing even waiting for any cleanup activities. In addition to the above, it is also possible to terminate the JVM without allowing the shutdown hooks to run by calling Runtime.halt() method.

Shutdown hooks are called when the application terminates normally (when all threads finish, or when System.exit(0) is called). Also, when the JVM is shutting down due to external causes such as a user requesting a termination (Ctrl+C), a SIGTERM being issued by O/S (normal kill command, without -9), or when the operating system is shutting down.

**2. Once started, Shutdown Hooks can be forcibly stopped before completion.**

This is actually a special case of the case explained before. Although the hook starts execution, it is possible to be terminated before it completes, in cases such as operating system shutdowns. In this type of case, the O/S waits for a process to terminate for a specified amount of time once the SIGTERM is given. If the process does not terminate within this time limit, then the O/S terminates the process forcibly by issuing a SIGTERM (or the counterparts in Windows). So it is possible that this happens when the shutdown hook is halfway through its execution.

Therefore, it is advised to make sure that the Shutdown Hooks are written cautiously, ensuring that they finish quickly, and do not cause situations such as deadlocks. Also, the JavaDoc [1] specifically mentions that one should not perform long calculations or wait for User I/O operations in a shutdown hook.

**3. We can have more than one Shutdown Hooks, but their execution order is not guaranteed.**

As you might have correctly guessed by the method name of the addShutdownHook method (instead of setShutdownHook), you can register more than one shutdown hook. But the execution order of these multiple hooks is not guaranteed by the JVM. The JVM can execute shutdown hooks in any arbitrary order. Moreover, the JVM might execute all these hooks concurrently.

**4. We cannot register/unregister Shutdown Hooks within Shutdown Hooks**

Once the shutdown sequence is initiated by the JVM, it is not allowed to add more or remove any existing shutdown hooks. If this is attempted, the JVM throws IllegalStateException.

**5. Once shutdown sequence starts, it can be stopped by Runtime.halt() only.**

Once the shutdown sequence starts, only Runtime.halt() (which forcefully terminates the JVM) can stop the execution of the shutdown sequence (except for external influences such as SIGKILL). This means that calling System.exit() within a Shutdown Hook will not work. Actually, if you call System.exit() within a Shutdown Hook, the VM may get stuck, and we may have to terminate the process forcefully.

**6. Using shutdown hooks require security permissions.**
If we are using Java Security Managers, then the code which performs adding/removing of shutdown hooks need to have the shutdown hooks permission at runtime. If we invoke this method without permission in a secure environment, then it will result in SecurityException.

**References :**
http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/Runtime.html#addShutdownHook(java.lang.Thread)
This article is contributed by **Saket Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@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.

Article Contributed By :

GeeksforGeeks

A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

# JVM Shutdown Hook in Java

About Odysseas Mourtzoukos

Mourtzoukos Odysseas is studying to become a software engineer, at Harokopio University of Athens. Along with his studies, he is getting involved with different projects on gaming development and web applications. He is looking forward to sharing his knowledge and experience with the world.

# JVM Shutdown Hook in Java

Posted by: Odysseas Mourtzoukos in Core Java March 15th, 2021 0 511 Views

This article is a tutorial on implementing a simple JVM Shutdown Hook in Java. In this example, we will take a look at different ways we can terminate a JVM application.

## 1. Introduction

Shutdown Hooks allow developers to plug in the desired code to be executed when the JVM is shutting down. This is useful when we need to do special clean-up operations before the VM finishes terminating. Shutdown hooks are basically initialized but unstarted threads. When the JVM begins its shutdown process, it will start all registered hooks in an unspecified order. After running all hooks, the JVM will halt.

Also, Shutdown Hooks are also executed even when the VM is shutting down due to an external reason like a kill request from the O/S. The general methods such as System.exit(0) will only work if the VM is terminated by a controlled process.

## 2. Technologies used

- Java SE 14

## 3. Adding Hooks

In order to add a shutdown hook, we can use the `Runtime.getRuntime().addShutdownHook()` method:*Adding Hook*

```
Runtime.getRuntime().addShutdownHook(new Thread(){
public void run(){
    System.out.println("Shutdown Hook running");
    }
});
```

The way it works is that a created thread with the desired code to be executed is defined, and then we initialize it by adding it to the current runtime.

## 4. JVM Shutdown Hook Simple Example

*Simple Example*

```
01
02class MyShutdownHookThread extends Thread {
03    public void run() {
04        System.out.println("Main terminated. We are in Shutdown Hook.\nBye!");
05    }
06}
07public class ShutdownHook {
08    public static void main(String[] args) throws Exception {
09
10        Thread shutdownHook = new MyShutdownHookThread();
11        Runtime.getRuntime().addShutdownHook(shutdownHook);
12
13        System.out.println("Now main is running.");
14
15    }
16}
```

- line 1: we create a new thread class that will actually be out Shutdown Hook.
- line 2: inside the method run(), we write the code we want to execute
- line 11: inside our main code, we initialize our hook bypassing the thread class we created as a variable in the `addShutdownHook()` method

Fig. 1. Output of ShutdownHook class.

# 5. Removing a Shutdown Hook

Java also gives as the ability to remove a shutdown hook if it is already defined by using the method*Simple Example with Removing Hook*

```
01
02class RemovedShutdownHookThread extends Thread {
03    public void run() {
04        System.out.println("Main terminated. We are in Shutdown Hook.\nBye!");
      }
05}
06
07public class RemoveShutdownHook {
08    public static void main(String[] args) throws Exception {
09
10        Thread shutdownHook = new RemovedShutdownHookThread();
11        Runtime.getRuntime().addShutdownHook(shutdownHook);
12
13        System.out.println("Now main is running.");
14
15        Runtime.getRuntime().removeShutdownHook(shutdownHook);
      }
16}
17
```

- line 15: we are removing the hook bypassing the name in `removeShutdownHook()` method

Fig. 2. Output of ShutdownHook class with remove method

# 6. Summary

In these examples, we demonstrated how we can use the Shutdown Hook methods that Java provides us. As we noticed, even though we initialized the Shutdown Hook before the main code of our program, it ran after the program terminated. This is very helpful as it gives us flexibility and a safe way to run a specific part of code e.g memory clean up.

## 7. Download the source code

This was an example of how to create and remove a JVM Shutdown Hook in Java.

**Download**
You can download the full source code of this example here: **JVM Shutdown Hook in Java**

Tagged with: Java JVM Shutdown Hook

### About Java Code Geeks

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

### Disclaimer

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.