News Knowledge Base Deals About Search...

Java Code Geeks
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

ANDROID | CORE JAVA | DESKTOP JAVA | ENTERPRISE JAVA | JAVA BASICS | JVM LANGUAGES | SOFTWARE DEVELOPMENT | DEVOPS

Home » Desktop Java » swing » How to Close an Application Properly

ABOUT JYOTI JHA

Jyoti is a tech enthusiast and is an avid programmer. She holds a post graduation degree in (M.Tech) Computer Science Engineering from Thapar Univeristy, Patiala, India. Post her graduate studies, she has worked in Software companies such as SLK Software and Aricent, India as Software Engineer in various projects primarily in the field of Requirement analysis and design, implementing new algorithms in C++ and JAVA used in ISDN network and designing databases and. She is inquisitive about socio economic reforms as well as advancement in technical fronts and keep herself informed with TED talks and various blogs.

# How to Close an Application Properly

Posted by: Jyoti Jha in swing October 5th, 2016 1 Comment 7434 Views

In order to close Java program we need to consider which kind of Java application is it?, because termination of Java application varies between normal core java program to swing GUI application. In general all Java programs terminate automatically once all user threads created by program finishes its execution, including main thread.

## 1. Introduction

JVM doesn't wait for daemon thread so as soon as last user thread finished, Java program will terminate. If you want to close or terminate your java application before this, your only option is to use

```
System.exit(int status)
```

or

```
Runtime.getRuntime().exit()
```

.

This cause JVM to abandon all threads and exit immediately. Shutdown hooks are get called to allow some last minute clearing before JVM actually terminates.

```
System.exit()
```

also accept an int status parameter where a non zero value denote abnormal execute and the result returned by java command to caller. In this java tutorial we will see example of closing both Java program and Java Swing application.
Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier

```
Abstract Window Toolkit (AWT)
```

. JAVA provides a rich set of libraries to create Graphical User Interface in platform independent way.

## 2. Java Swing

Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API

## 2.1 MVC Architecture

Swing API architecture follows loosely based MVC architecture in the following manner.

- A Model represents component's data.
- View represents visual representation of the component's data.
- Controller takes the input from the user on the view and reflects the changes in Component's data.
- Swing component have Model as a seperate element and View and Controller part are clubbed in User Interface elements. Using this way, Swing has pluggable look-and-feel architecture.

Every user interface considers the following three main aspects:

- UI elements : These are the core visual elements the user eventually sees and interacts with. GWT provides a huge list of widely used and common elements varying from basic to complex.
- Layouts: They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface).
- Behavior: These are events which occur when the user interacts with UI elements.

## 2.2 Swing Features

Light Weight – Swing component are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.

- Rich controls – Swing provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, table controls.
- Highly Customizable – Swing controls can be customized in very easy way as visual apperance is independent of internal representation.
- Pluggable look-and-feel– SWING based GUI Application look and feel can be changed at run time based on available values.

## 2.3 Setup

Popular Java Editors:
To write your java programs you will need a text editor. There are even more sophisticated IDE available in the market. But for now, you can consider one of the following:

- Notepad: On Windows machine you can use any simple text editor like Notepad TextPad.
- NetBeans: is a Java IDE that is open source and free which can be downloaded from http://www.netbeans.org/index.html.
- Eclipse: is also a java IDE developed by the eclipse open source community and can be downloaded from http://www.eclipse.org

**Prerequisite**
This example is developed on Eclipse therefore a compatible Eclipse IDE is required to be installed on the system.
We also need WindowBuilder tool to be installed on Eclipse IDE for the easiness of the work.

# 3. Example of Closing Java program using System.exit()

Here is a code example of closing Java program by calling System.exit() method. Remember non zero argument to exit() method like exit(1) denotes abnormal termination of Java application.

*ApplicationExit.java*

```
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *Java program which terminates itself by using System.exit() method , non zero call to exit() method denote
 */
public class JavaCloseExample {

    public static void main(String args[]) throws InterruptedException {

       Thread t = new Thread(){
            @Override
           public void run(){
               while(true){
                   System.out.println("User thread is running");
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException ex) {
                        Logger.getLogger(JavaCloseExample.class.getName()).log(Level.SEVERE, null, ex);
                    }
               }
           }
       };

       t.start();
       Thread.sleep(200);
       System.out.println("terminating or closing java program");
       System.exit(1); //non zero value to exit says abnormal termination of JVM
    }
}

Output:
User thread is running
User thread is running
terminating or closing java program
Java Result: 1  //1 is what we passed to exit() method
```

This Java program first creates a Thread  in main method and start it,  which prints "User thread is running" and then main thread sleeps for 200 Milli second. Till then, the other user thread is running and printing but once the main thread wakes up, it terminates the program by calling

```
exit()
```

method of

```
java.lang.System
```

class.

## 3.2 How to close Java swing application from program

Swing application mostly uses

```
JFrame
```

as top level container which provides two option to close swing GUI application from code. First option which is default is EXIT_ON_CLOSE which terminates Java swing GUI program when you click close button on

```
JFrame
```

window. Another option is

```
DISPOSE_ON_CLOSE
```

which terminates JVM if last displayable window is disposed off.

Difference between

```
EXIT_ON_CLOSE
```

and

```
DISPOSE_ON_CLOSE
```

is that if you have a non daemon thread running it will not be closed in case of

```
DISPOSE_ON_CLOSE
```

, while

```
EXIT_ON_CLOSE
```

terminate JVM even if user thread is running. Run the below example by un comment

```
DISPOSE_ON_CLOSE
```

in your IDE and you can see user thread running even after clicking on close button. Here is a complete code example of closing Swing application in Java.

*ApplicationExit.java*

```java
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;

/**
 * Java swing program which terminates itself by calling  EXIT_ON_CLOSE and DISPOSE_ON_CLOSE
 */
public class CloseSwingExample {

    public static void main(String args[]) throws InterruptedException {

        JFrame frame = new JFrame("Sample");
        //frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); won't terminate JVM if user thread runnin
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 200);
        frame.setVisible(true);

        Thread t = new Thread() {

            @Override
            public void run() {
                while (true) {
                    System.out.println("User thread is running");
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException ex) {
                        Logger.getLogger(CloseSwingExample.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }
        };

        t.start();

    }
}
```

### 3.3 Important Points for closing or terminating a Java application

-
  ```
  System.exit()
  ```
  actually calls
  ```
  Runtime.getRuntime().exit()
  ```
  method.
- Non zero argument to exit() denotes abnormal termination of Java program.
- Shutdown hooks are executed before Java program actually terminates.
- There are two options to close Java Swing application one is
  ```
  EXIT_ON_CLOSE
  ```
  and other is
  ```
  DISPOSE_ON_CLOSE
  ```
  .

-
  ```
  DISPOSE_ON_CLOSE
  ```
  doesn’t terminate JVM if any user thread is running.
- You can also implement window listener to implement your closing mechanism by using
  ```
  System.exit()
  ```
  in Swing application.

That’s all on how to close or terminate Java program. We have also seen example of closing Swing application in Java and difference between

```
EXIT_ON_CLOSE
```

and

```
DISPOSE_ON_CLOSE
```

.

## 4. Download The Source Code

This was an example of closing a Java application properly.

**Download**
You can download the full source code of this example here: **ApplicationExit**

(No Ratings Yet) 1 Comment 7434 Views Tweet it!

Sign up

## LIKE THIS ARTICLE? READ MORE FROM JAVA CODE GEEKS

1

## Leave a Reply

Join the discussion...

1 0 0

1

Subscribe

newest oldest most voted

marc0s

Guest

Why can you provide the java code, i cannot use the class

+ 0 − Reply

5 months ago

KNOWLEDGE BASE

- Courses
- Minibooks
- News
- Resources
- Tutorials

THE CODE GEEKS NETWORK

- .NET Code Geeks
- Java Code Geeks
- System Code Geeks
- Web Code Geeks

HALL OF FAME

- Android Alert Dialog Example
- Android OnClickListener Example
- How to convert Character to String and a String to Character Array in Java
- Java Inheritance example
- Java write to File Example
- java.io.FileNotFoundException – How to solve File Not Found Exception
- java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception
- java.lang.NoClassDefFoundError – How to solve No Class Def Found Error
- JSON Example With Jersey + Jackson
- Spring JdbcTemplate Example

ABOUT JAVA CODE GEEKS

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

DISCLAIMER

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