Difference between final, finally and finalize in Java

02 July 2025 20:13

The **final, finally and finalize** are keywords in Java that are used in exception handling. Each of these keywords has a different functionality. The basic difference between **final, finally and finalize** is that **final is an access modifier, finally, is the block in exception handling**. And **finalize() is the method of the object class**.

public final class String extends Object
implements Serializable, Comparable<String>, CharSequence, Constable, ConstantDesc

From <https://docs.oracle.com/en/java/javase/24/docs/api//java.base/java/lang/String.html>

"final" Keyword: A final is a keyword used to restrict changes. A final variable cannot be reassigned. A final method cannot be overridden. And of final class. Can't be subclassed. It ensures immutability and is often used for constants.

Characteristics of final keyword

- **Used to declare constants:** A final variables value cannot be changed once assigned.
- **Prevents method overriding:** A final method cannot be overridden in a subclass.
- **Restricts inheritance:** A final class cannot be extended by any other class.

"final" keyword Example

```
10 public class FInalKeywordExample1
11 {
12     final int age = 18;
13
14     public void reassgnAge(int age){
15         this.age = age;
16     }
17 }
```

cannot assign a value to final variable age

```
10 public class FInalKeywordExample1 extends String
11 {
12     final int age = 18;
13
14     public void reassgnAge(int age){
15         //this.age = age;
16     }
17 }
```

cannot inherit from final java.lang.String

"finally" Block: A finally is a block used with try catch statement to execute important code like closing resources. It always runs whether an exception occurs or not, ensuring cleanup actions are performed.

Characteristics of finally:

- **Part of exception handling:** Used with try and catch blocks.
- **Always execute:** Runs regardless of whether an exception is thrown or caught.
- **Used for resource cleanup:** Like closing files, closing database connections etc.

Java finally block example:

```
package ExceptionHandling;
```

```
import java.util.*;


/**
 * Write a description of class FInalKeywordExample1 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class FinallyKeywordExample1
{
   public static void main(String args[]){
      int x, y, res;
      Scanner sc = new Scanner(System.in);
      System.out.print("\f");
      try{
         System.out.print("Enter any integer value: ");
         x = sc.nextInt();
         System.out.print("Enter any integer value: ");
         y = sc.nextInt();
         res = x / y;
         System.out.print("\nResult of division: " + res);
      }
      catch(ArithmeticException ae){
         System.out.print("\nDivisor can't be zero(0).");
         System.out.print("\n"+ ae.getMessage());
      }
      finally{
         System.out.print("\nfinally block is always executed.");
      }
      System.out.print("\nRest of the code...");
   }
}
```

catching object.

returning one string object that contains a message for the user.

Outputs:

```
Enter any integer value: 88
Enter any integer value: 0

Divisor can't be zero(0).
/ by zero
finally block is always executed.
Rest of the code...
```

```
Enter any integer value: 81
Enter any integer value: 14

Result of division: 5
finally block is always executed.
Rest of the code...
```

finalize() Method

A finalize () is a method called by garbage collector before an object is destroyed. It's used to perform cleanup operations like releasing resources. But it is deprecated and not recommended in modern Java.

Characteristics of finalize () method

- **Defined in java.lang.Object class:** Can be overridden by any class.
- **Called by the garbage collector:** Just before destroying an object.
- **Used for cleanup:** Freeing resources like file handles or network sockets. Or database connections.
- **Not reliable:** No guarantee it will be executed timely or at all.

`protected void`	`finalize()`	Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.

```
package ExceptionHandling;


/**
 * Write a description of class FinalizeMethodDemo here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class FinalizeMethodDemo
{
   @Override
   protected void finalize() throws Throwable{
      System.out.print("\nfinalize() method called before object is garbage collected.");
   }

   public static void main(String args[]){
      FinalizeMethodDemo ob = new FinalizeMethodDemo();
      ob = null; //Eligible for garbage collection.
      //Request JVM to run garbage collector.
      System.gc();
      System.out.print("\nMain method completed.");
   }
}
```

Output:

```
Main method completed.
finalize() method called before object is garbage collected.
```

First main method is completed, then garbage collector method has executed.

final vs finally Vs finalize() in Java

Aspect	final	finally	finalize()
Definition	final is the keyword and access modifier that is used to apply restrictions on a class, method or variable.	finally is the block in Java. Exception handling to execute the important code whether the exception occurs or not. So this is finally.	finalize() is the method in Java that is used to perform cleanup processing just before an object is garbage collected.
Applicable to	The final keyword is used with the classes, methods and variables.	finally block is always related to the try-catch block in exception handling.	The finalize() method is used with the objects.

Functionality	• Once declared, a final variable becomes a constant and cannot be modified. • The final method cannot be overridden by subclass. • The final class cannot be inherited.	• Finally block runs the important code even if an exception occurs or not. • The finally block cleans up all the resources used in a try block.	The finalize() method performs the cleaning activities with respect to the object before its destruction.
Execution	The final method is executed only when we call it.	Finally block is executed as soon as try catch block is executed. Its execution is not dependent on the exception.	The finalize () method is executed just before the object is destroyed.
Inheriting/Overriding	Prevents method overriding and class inheritance.	Not related to inheritance or overriding.	Can be overridden from the Object class.
Use case	To create constants, prevent subclassing or method overriding.	To ensure resource cleanup like closing files, streams etc. in exception handling.	To perform clean-up activities like memory or resource release before the object is collected.
Control	Gives compile-time control to avoid unintended changes.	Provides runtime control to handle cleanup post exception or normal flow.	Invoked by JVM (Java virtual machine) The programmer cannot call it directly for cleanup.