Difference Between Final, Finally and Finalize in Java

The words ‘final, finally, and finalize’ lies within the context of Java. Each one of them is provided a different functioning. The basic difference between final, finally, and finalize is that **final** is an access modifier, **finally** is a block and **finalize** is a method of an object class.

### **Comparison Chart**

| **BASIS FOR COMPARISON** | **FINAL** | **FINALLY** | **FINALIZE** |
| --- | --- | --- | --- |
| Basic | Final is a "Keyword" and "access modifier" in Java. | Finally is a "block" in Java. | Finalize is a "method" in Java. |
| Applicable | Final is a keyword applicable to classes, variables and methods. | Finally is a block that is always associated with try and catch block. | finalize() is a method applicable to objects. |
| Working | (1) Final variable becomes constant, and it can't be reassigned. (2) A final method can't be overridden by the child class. (3) Final Class can not be extended. | A "finally" block, clean up the resources used in "try" block. | Finalize method performs cleans up activities related to the object before its destruction. |
| Execution | Final method is executed upon its call. | "Finally" block executes just after the execution of" try-catch" block. | finalize() method executes just before the destruction of the object. |

### **Definition of Final**

“Final” is a keyword in Java. It is an access modifier. The “final” keyword is applicable to the classes, methods and variables. Let us see how it works with each of them.

#### **Final Variable**

* When a final keyword is applied to the variable, it can not be further modified.
* A final variable must be initialized when it is declared.
* In a common coding convention, final variables are declared in UPPERCASE.
* The final variable does not occupy memory on a per-instance basis.

final int FILE\_OPEN = 2;

#### **Final Methods**

* When a method in class, is declared as final, it can not be overridden by its subclass.
* Small methods that are declared final can be made “inline” by the compiler which will reduce the overhead of function calling and increase performance enhancements.
* Overridden methods call, are resolved dynamically, but when a method is declared as final, it can not be overridden. Hence, the function calling can be resolved at compile time.

class A {

final **void** meth() {

System.out.println("This is a final method.");

}

}

**class** B **extends** A {

**void** meth() { //Clas B can not inherit the method of class A.

System.out.println("Does not override");

}

}

#### **Final Class**

* When a class is declared as final, then it can not be inherited by any subclass.
* Declaring a class as final will automatically declare all its methods final.
* You can not declare a class as both “abstract” and “final”.

final **class** A {

// ...

}

**class** B **extends** A { // Class B can not inherit class A

// ...

}

### **Definition of Finally**

* In Java “finally” is a block that is always associated with the try/catch block.
* The “finally” block executes after the try/catch block and before the code following try/catch block.
* The “finally” block will execute whether the exception is thrown or not.
* When an exception is thrown, and no catch block matches the exception even then, the “finally” block is executed.
* When a method returns to the caller from inside the try/catch block via an uncaught exception or an explicit return statement, the “finally” block gets executed just before the method returns to the caller.
* The “finally” block is used to clean up the resources or free the memory used in “try” block.
* The “finally” block is optional, but it is a good practice to write finally block after the try/catch block.

class FinallyExample {

// This method throw an exception out of the method.

static **void** procA() {

**try** {

System.out.println("inside procA");

throw **new** RuntimeException("demo");

} finally {

System.out.println("procA's finally");

}

}

// Return from within a try block.

static **void** procB() {

**try** {

System.out.println("inside procB");

**return**;

} finally {

System.out.println("procB's finally");

}

}

// Execute a try block normally.

static **void** procC() {

**try** {

System.out.println("inside procC");

} finally {

System.out.println("procC's finally");

}

}

public static **void** main(String args[]) {

**try** {

procA();

} **catch** (Exception e) {

System.out.println("Exception caught");

}

procB();

procC();

}

}

//output

inside procA

procA’s finally

Exception caught

inside procB

procB’s finally

inside procC

procC’s finally

### **Definition of Finalize**

* Finalize is a method in an object class.
* An object may be holding some non-java resources such as file handle; then it must be freed before it is destroyed.
* This method is invoked by the garbage collector before it destroys the object completely.
* This method performs cleanup activities for the object before it is being destroyed.

The general form of the method finalize() is as follow.

protected **void** finalize( )

{

// finalization code here

}

The finalize method is declared as protected so that it can not be accessed from outside the class.

This method is always called before the garbage collection.

## **Key Differences Between Final, Finally, and Finalize**

1. The keyword final is an access modifier, finally is a block and finalize is a method.
2. The keyword final is applicable to the classes, variables and methods of the classes finally is a block associated with the try catch block that is used to handle exceptions, finalize is a method that operates only on objects.
3. The variable once declared as final becomes constant and can’t be reassigned again, a method declared as final can’t be overridden, and class once declared as final can never be inherited. The finally block is used to clean up the resources utilised by try and catch block. The finalize method is used to clean up the resources used by an object before the object is destroyed.

### **Conclusion**

Final, finally and finalize has a different effect when applied on a method.