

Finally Clause

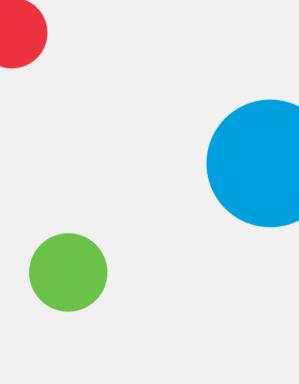
Agenda



Finally Clause

Sensitivity: Internal & Restricted

Finally Clause





Using finally

- When an exception occurs, the execution of the program takes a non-linear path, and could bypass certain statements
- A program establishes a connection with a database, and an exception occurs
- The program terminates, but the connection is still open
- To close the connection, **finally** block should be used
- The finally block is guaranteed to execute in all circumstances

Using finally

- When an exception occurs, the execution of the program takes a non-linear path, and could bypass certain statements
- A program establishes a connection with a database, and an exception occurs
- The program terminates, but the connection is still open
- To close the connection, **finally** block should be used
- The finally block is guaranteed to execute in all circumstances

```
import java.io.*;
class FinallyDemo{
static void funcA() throws FileNotFoundException
   try{
       System.out.println("inside funcA()");
       throw new FileNotFoundException();
   finally{
       System.out.println
       ("inside finally of funA()");
```

```
static void funcB() {
 try{
     System.out.println("inside funcB()");
 finally{
     System.out.println
      ("inside finally of funB()");
```

```
static void funcC() {
   try{
        System.out.println("inside funcC()");
   }
   finally{
        System.out.println
        ("inside finally of funcC()");
   }
}
```

```
public static void main(String args[]) {
 try{
     funcA();
 catch (Exception e) {
     System.out.println("Exception caught");
 funcB();
 funcC();
```

Sensitivity: Internal & Restricted

Significance of printStackTrace() method

- We can use the *printStackTrace()* method to print the program's execution stack
- This method is used for debugging

Example on printStackTrace() method

```
import java.io.*;
class PrintStackExample {
 public static void main(String args[]) {
   try {
      m1();
   catch(IOException e) {
      e.printStackTrace();
```

Sensitivity: Internal & Restricted

contd...

Example on printStackTrace() method

```
static void m1() throws IOException {
 m2();
static void m2() throws IOException {
 m3();
static void m3() throws IOException{
 throw new IOException();
                       Expected Output
```

java.io.IOException

at PrintStackExample.m3(PrintStackExample.java:24) at PrintStackExample.m2(PrintStackExample.java:20) at PrintStackExample.m1(PrintStackExample.java:16) at PrintStackExample.main(PrintStackExample.java:5)

Quiz

What will be the result, if we try to compile and execute the following code as java Ex2 A

```
class Ex2 {
    public static void main(String[] args) {
      try {
        int i= Integer.parseInt(args[0]);
        System.out.println(i);
      catch (NumberFormatException e) {
        System.out.println(e);
      System.out.println("Exception Caught");
      finally { }
```

It will throw compilation Error

Quiz(Contd.).

What will be the result, if we try to compile and execute the following code

```
public class Tester {
static void method() {
        throw new Exception();
public static void main(String[] args) {
        try {
                                          It will throw compilation Error;
        method();
                                          Why?
        } catch (Throwable e) {
                                          How to remove the same?
                 try {
                 throw new Exception();
                 } catch (Exception ex) {
                 System.out.print("exception");
                 } finally {
                 System.out.print("finally");
} }
```

Summary

In this session, you were able to:

- Learn about finally clause
- Learn about print stack trace method



Thank You

