

**Java means DURGA SOFT..**

# **CORE JAVA**

## **Material**



**India's No.1 Software Training Institute**

**DURGASOFT**

**[www.durgasoft.com](http://www.durgasoft.com) Ph: 9246212143,8096969696**

## Exception Handling

What is the difference between Error and Exception?

| Error  | Exception  |
|--|--|
| 1. Error is a problem at runtime, for which we are unable to provide solutions programmatically. | Exception is a problem, for which, we are able to provide solution programmatically. |
| Ex: JVM internal Problem<br>StackOverFlowError<br>InSufficientMainMemory                         | Ex: ArithmeticException<br>NullPointerException<br>ArrayIndexOutOfBoundsException    |

**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

# JAVA MEANS DURGASOFT

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
 SOFTWARE SOLUTIONS

**#202 2<sup>nd</sup> FLOOR**  
[www.durgasoft.com](http://www.durgasoft.com)

**040-64512786**  
**+91 9246212143**  
**+91 8096969696**

### Definition of Exception:

Exception is an Unexpected event occurred at runtime provided by the users while entering dynamic input to the Java program, provided by the database while executing sql queries in JDBC applications or provided by the remote machine while establish connection between local machine and remote machine in distributed applications causes abnormal termination to the Java programs.

In Java applications, exceptions may provide abnormal terminations to the Java programs, these abnormal terminations may effect operating system, network applications.....

To overcome the above problems, we have to provide smooth terminations to the Java applications, for this, we have to handle the exceptions properly, for this, we have to use "Exception Handling Mechanisms".

**Note: Terminating Java program in the middle is called as Abnormal Termination, terminating Java program at the end is called as Smooth Termination.**

**[www.durgasoftonlinetraining.com](http://www.durgasoftonlinetraining.com)**



**Online Training  
Pre Recorded Video  
Classes Training  
Corporate Training**

**Ph: +91-8885252627, 7207212427  
+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : [durgasoftonlinetraining@gmail.com](mailto:durgasoftonlinetraining@gmail.com)**

Java is Robust programming language, because,

1. Java is having very good memory management system in the form of Heap memory Management System, a dynamic memory management system, which allocated and deallocated memory for the objects at runtime as per the JVM requirement.

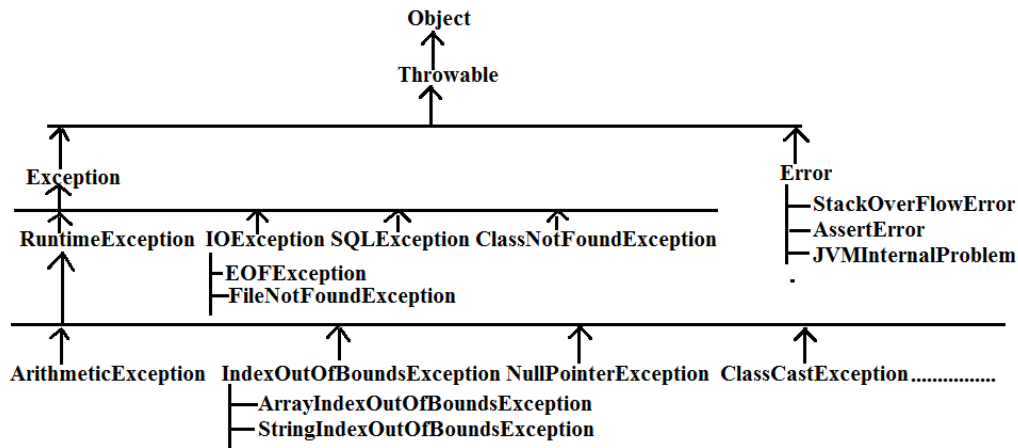
2. Java is having very good Exception handling mechanisms due to the availability of very good predefined library to represent each and every exceptional situation.

There are two types of exceptions in Java:

1. Predefined Exceptions
2. UserDefined Exceptions

1. Predefined Exceptions:

These Exceptions are defined by Java programming language along with Java predefined library.



There are two types of predefined exceptions:

1. Checked Exceptions
2. Unchecked Exceptions

**[www.durgasoftonlinetraining.com](http://www.durgasoftonlinetraining.com)**



**Online Training  
Pre Recorded Video  
Classes Training  
Corporate Training**

**Ph: +91-8885252627, 7207212427  
+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : [durgasoftonlinetraining@gmail.com](mailto:durgasoftonlinetraining@gmail.com)**

Q)What is the difference between checked Exception and Unchecked Exception?


- 1.Checked Exception is an exception identified by the compiler at compilation time[occurred at runtime, not at compilation time].



**NOTE:while Java program compilations, compiler will recognize same situations to get exceptions at runtime then that exceptions are called as Checked Exceptions.**

Unchecked exceptions are the exceptions recognized by the JVM at runtime, not a compiler at compilation time.

2.Runtime Exceptions and its subclasses, error and its subclasses are treated as Unchecked exceptions and the remaining exception classes are treated as checked exception.



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

# **JAVA MEANS DURGASOFT**

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
SOFTWARE SOLUTIONS

**#202 2<sup>nd</sup> FLOOR**  
**www.durgasoft.com**

**040-64512786**  
**+91 9246212143**  
**+91 8096969696**

There are two types of checked exceptions:

- a) pure checked exceptions
- b) partially checked exceptions.

Q)What is the difference between pure checked and partially checked exceptions?

If any checked exception is having only checked exceptions as child classes then that checked exceptions are called as "pure checked exceptions".

Ex: IOException

If any checked exception is having atleast one unchecked exception as a child class then that checked exception is called as "partially Unchecked exception"

Ex: Exception, Throwable

## Review of Predefined Exceptions:

---

### 1.Arithmetic Exception:

In Java applications,when we have a situation like number divided by zero,("num/0") then JVM will rise an exception like "Arithmetic Exception".

```
class Test{  
  
public static void main(String args[]){  
  
int i=100;  
  
int j=0;  
  
float f=i/j;  
  
System.out.println(f);  
  
}}
```



If we execute the above code then JVM will provide the following Exception message. Exception in thread "main" java.lang.ArithmeticException: /by zero at Test.main(Test.java7)

The above exception message is divided into the following three parts:

- 1.Exception Name : java.lang.ArithmeticException
- 2.Exception Description: / by zero
- 3.Exception location : Test.java:7

## 2.NullPointerException:

In Java applications, when we access any variable or method on a reference variable having 'null' value then JVM will rise an exception like "NullPointerException".



```
import java.util.*;

class Test{

public static void main(String args[]){

Date d=null;

System.out.println(d.toString());

}}
```

If we execute the above code then JVM will provide the following exception details.

Exception Name : java.lang.NullPointerException

Exception Description: No Description

Exception Location: Test.java: 7

### 3.ArrayIndexOutOfBoundsException:

In Java applications, when we access a value from an array whose index value is greater than or same as array size then JVM will rise "ArrayIndexOutOfBoundsException".

Ex:

```
class Test{  
    public static void main(String args[]){  
        int[] a={ 1,2,3,4};  
        System.out.println(a[4]);  
    }  
}
```

If we execute the above code then JVM will provide the following exception message:



Exception Name: java.lang.ArrayIndexOutOfBoundsException

Exception Description: 4



Exception Location: Test.java: 10

#### 4.FileNotFoundException:

In Java applications,when we are preparing FileInputStream or FileReader with a particular source file,where if the specified source file is not existed then JVM will rise "FileNotFoundException"

Ex:

```
import java.io.*;  
  
class Test{  
  
public static void main(String args[]) throws Exception{  
//FileInputStream fis=new FileInputStream("abc.txt");  
FileReader fr=new FileReader("abc.txt");  
}}
```



If we execute the above code then JVM will provide the following exception message:

Exception Name: java.io.FileNotFoundException

Exception Description: abc.txt(The System Cannot Find the file specified)

Exception Location: Test.java: 7

### 5. ClassNotFoundException:

In Java applications, when we try to load a particular class bytecode to the memory by using `Class.forName(-)` method, where if the specified class .class file is not available at current location, at java predefined library and at the locations referred by "classpath" environment variable then JVM will rise "ClassNotFoundException".

```
class Test{  
    public static void main(String args[]){  
        Class.forName("A");  
    }  
}
```



If we execute the above code JVM will provide the following exception details:

Exception Name: `java.lang.ClassNotFoundException`

Exception Description: A

Exception Location: `Test.java: 5`

### 6. InstantiationException:

In Java applications, when we are creating an object for the particular class by using `newInstance()` method from "java.lang.Class", where JVM will

search and execute 0-argument constructor, if the respective class is not having 0-argument constructor then JVM will rise "InstantiationException".

```
class A{  
    static{  
        System.out.println("Class Loading");  
    }  
    A(int i){  
        System.out.println("Object Creating");  
    }  
}  
class Test{  
    public static void main(String args[]) throws Exception{  
        Class c=Class.forName("A");  
        Object obj=c.newInstance();  
    }  
}
```



If we execute the above code then JVM will provide the following ExceptionDetails.

Exception Name : java.lang.InstantiationException

Exception Description: A

Exception Location: Test.java: 7

### 7.1 IllegalAccessException:

In Java applications, when we are trying to create object for a particular class by using "newInstance()" method where JVM will search and execute "non-private" constructor, where if the respective class is having only "private" constructor then JVM will rise "IllegalAccessException".

Ex:

```
class A{
    private A(){
        System.out.println("Object Creating");
    }
}
class Test{
    public static void main(String args[])throws Exception{
        Class c=Class.forName("A");
        Object obj=c.newInstance();
    }
}
```



If we execute the above code then JVM will provide the following exception details.



Exception Name: java.lang.IllegalAccessException

Exception Description: Class Test can not access a member of class A with modifiers "private".

Exception Location: Test.java: 10

### 8.1 IllegalArgumentException:

In general, in Java applications, we may provide methods with parameters, where method parameters will allow the values on the basis of the parameter dataTypes ranges. In some situations, methods may have restrictions about the parameter values irrespective of dataTypes provided ranges. In this context, if we provide any value which is not in the range defined by the methods then JVM will rise an exception like "IllegalArgumentException".

In Java, Threads are having priority values range that is from 1 to 10. If we pass any priority value which is not in between 1 and 10 as a parameter to setPriority() method then JVM will rise an exception like "IllegalArgumentException".



```
class Test{
    public static void main(String args[]){
        Thread t=new Thread();
        t.setPriority(15);
```

```
}}
```

If we execute the above code then JVM will provide the following exception details.

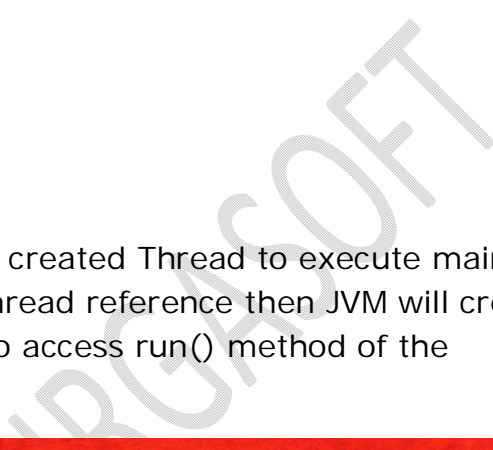
Exception Name: java.lang.IllegalArgumentException

Exception Description: No Description

Exception Location: Test.java: 10

### 9.1 IllegalStateException:

In MultiThreading, when Main Thread [JVM created Thread to execute main() method] access start() method over any thread reference then JVM will create an Thread and JVM will start that thread to access run() method of the respective thread class.



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**  
**JAVA MEANS DURGASOFT**  
**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

|  |  |   |
|--|--|---|
| <small>AN ISO 9001:2008 CERTIFIED</small><br><b>DURGA</b><br><small>SOFTWARE SOLUTIONS</small> | <b>#202 2<sup>nd</sup> FLOOR</b><br><b>www.durgasoft.com</b> | <b>040-64512786</b><br><b>+91 9246212143</b><br><b>+91 8096969696</b> |
|--|--|---|

In MultiThreading, if any Thread terminated along with Main Thread automatically then that thread is called as "Daemon Thread".

To make Daemon threads, we have to access setDaemon(true) method before starting that thread.

If we access setDaemon(true) method after starting the thread then JVM will rise an exception like "IllegalThreadStateException".

```
class MyThread extends Thread{
    public void run(){
        while(true){
            System.out.println("User Thread");
```

```
}}}  
class Test{  
    public static void main(String args[]){  
        MyThread mt=new MyThread();  
        mt.setDaemon(true);  
        mt.start();  
        for(int i=0;i<10;i++){  
            System.out.println("Main Thread");  
        }  
    }  
}}
```



If we run the above programme then JVM will provide the following exception details.

Exception Name: java.lang.IllegalThreadStateException

Exception Description: No Description

Exception Location: Test.java: 17

#### 10. ClassCastException:

In Java applications, it is possible to keep sub class object reference value in super class reference variable but it is not possible to keep super class object reference value in sub class reference variable.

In Java applications, when we are trying to keep super class object reference value in subclass reference variable in downcasting then JVM will rise an exception like "java.lang.ClassCastException".

```
class A{
}

class B extends A{
}

class Test{
    public static void main(String args[]){
        A a=new A();
        B b=(B)a;
    }
}
```

If we run the above code then JVM will provide the following exceptional details:

Exception Name: java.lang.ClassCastException

Exception Description: A cannot be cast to B

Exception Location: Test.java: 10



“throw” keyword:

throw is a JAVA keyword, it can be used to raise an exception intentionally as per the application requirement.

Syntax:

```
throw new Exception_Name([Param_List]);
```

**NOTE:** In general, we will utilize "throw" keyword in custom exceptions in order to raise an exception.



Ex:

```
import java.io.*;

class Test{

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

BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

System.out.println("Account Number :");

String accNo=br.readLine();

System.out.println("Account Name :");
```

```

String accName=br.readLine();

int p_num=Integer.parseInt(pin_number);

if(p_num > 1000 && p_num < 9999){

System.out.println("Account Details");

System.out.println("-----");

System.out.println("Account Number :"+accNo);

System.out.println("Account Name  :"+accName);

System.out.println("Pin Number    :"+XXXXXX);

System.out.println("Valid Pin Number");

}

else{

throw new RuntimeException("Invalid PIN Number");

}}}
```



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

**JAVA MEANS DURGASOFT**

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
SOFTWARE SOLUTIONS

**#202 2<sup>nd</sup> FLOOR**  
**www.durgasoft.com**

**040-64512786**  
**+91 9246212143**  
**+91 8096969696**

If any statement we have provided immediately after "throw" statement then compiler will rise an error like "unreachable statement".

```

class Test{

public static void main(String args[]){

System.out.println("Before Exception");
```

```
throw new ArithmeticException("My Arithmetic Exception");
System.out.println("After Exception");
}}
```

In Java, there are two ways to handle the exceptions:

1. By using "throws" keyword
2. By using try-catch-finally



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**  
**JAVA MEANS DURGASOFT**  
**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
 SOFTWARE SOLUTIONS

#202 2<sup>nd</sup> FLOOR  
[www.durgasoft.com](http://www.durgasoft.com)

040-64512786  
 +91 9246212143  
 +91 8096969696

### 1. "throws" keyword:

---

It is a Java keyword, it can be used to bypass the generated exception from the present method or constructor to the caller method (or) constructor.

In Java applications, "throws" keyword will be used in method declarations, not in method body.

In Java applications, "throws" keyword will allow an exception class name, it should be either same as the generated exception or super class to the generated exception. It should not be subclass to the generated Exception.

"throws" keyword will allow more than one exception in method prototypes.

In Java applications, "throws" keyword will be utilized mainly for checked exceptions.

Ex: void m1() throws RuntimeException{

Throw new ArithmeticException();

}

Status: Valid

Ex: void m1() throws FileNotFoundException{

Throw new IOException();

}

Status: Invalid

Ex:

Void m1() throws NullPointerException, ClassNotFoundException{

}

Status: Valid

Ex:

Void m1() throws IOException, FileNotFoundException

{

}



Status: Valid



If we specify any super exception class along with throws keyword, then it is not necessary to specify any of its child exception classes along with "throws" keyword.

NOTE: In any Java method, if we call some other method which is bypassing an exception by using "throws" keyword, then we must handle that exception either by using "throws" keyword in the present method prototype or by using "try-catch-finally" in the body of the present method.

Ex:

```
Void m1() throws Exception{
```

```
-----
```

```
-----
```

```
}
```

```
Void m2(){
```

```
try{
```

```
m1();
```

```
}
```

**www.durgasoftonlinetraining.com**



**Online Training**  
**Pre Recorded Video**  
**Classes Training**  
**Corporate Training**

**Ph: +91-8885252627, 7207212427**  
**+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : durgasoftonlinetraining@gmail.com**

Just Attend one Session & see difference in Quality.

# SPRING

**By Mr. Sriman**

(Realtime Expert & Brand of Frameworks)

**SPRING Means DURGASOFT**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
Software Solutions®  
[www.durgasoft.com](http://www.durgasoft.com)

# 202, 2nd Floor, HUDA Maitrivanam,  
Ameerpet, Hyd. Ph: 040-64512786,  
**9246212143, 8096969696**

**CORE JAVA with**  
**OCJP/SCJP**  
JAVA CERTIFICATION



**Mr. DURGA** M.Tech  
JAVA EXPERT  
Trained Thousands of Students



**One to One**  
**VIDEO**  
**CLASSES**

**EVERYTHING AT YOUR CONVENIENCE**

**At your convenient Time**

**With in your convenient duration**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
Software Solutions®  
[www.durgasoft.com](http://www.durgasoft.com)

# 202, 2nd Floor, HUDA Maitrivanam,  
Ameerpet, Hyd. Ph: 040-64512786,  
**9246212143, 8096969696**

```
catch(Exception e){  
e.printStackTrace();  
}}
```

```
void m1() throws Exception{
```

```
----
```

```
}
```

```
void m3() throws Exception{
```

```
    m1();
```

```
}
```

**Example program for “throws” keyword:**

```
Import java.io.*;
```

```
class A{
```

```
void add() throws Exception{
```

```
concat();
```

```
}
```

```
Void concat() throws IOException{
```

```
throw new IOException();
```

```
}
```



```
}
```

```

class Test{

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

A a=new A();

a.add();

}}

```

### Internal Flow:

If we compile the above program, then compiler will recognize throw keyword in concat() method at Line: 10

With this compiler will generate exception notification with 6 line as exception location, due to "throws" keyword in concat() method prototype the specified exception will be bypassed to concat() method call in add() method body at line: 6, due to "throws" keyword in add(-) method prototype, the specified exception is bypassed to add() method call in main() method that is at line: 18, due to "throws" keyword in main() method prototype, the specified exception will be bypassed to main() method call that is to JVM. In this context, compiler will not provide any exception notification, but when we execute the above program, Default Exception as part of JVM will take that exception and it will provide the exception details by including all the locations like 10<sup>th</sup> line, 6<sup>th</sup> line and 18<sup>th</sup> line.

**[www.durgasoftonlinelearning.com](http://www.durgasoftonlinelearning.com)**



**Online Training  
Pre Recorded Video  
Classes Training  
Corporate Training**

**Ph: +91-8885252627, 7207212427  
+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : [durgasoftonlinelearning@gmail.com](mailto:durgasoftonlinelearning@gmail.com)**



Q)What are the differences between "throw" and "throws" keywords?

Ans:

|  |   |
|--|---|
| 1."throw" keyword can be used to rise the exceptions intentionally as Per the application requirement. | "throws" keyword will by pass the exceptions from the present method to the caller method.              |
| 2."throw" keyword will be utilized in method body.   | "throws" keyword will be used in method declarations or in method prototype (or) in method header part. |
| 3."throw" keyword will allow only one exception class name.  | "throws" keyword will allow more than one exception class name.   |

try-catch-finally:

In Java application "throws" keyword is not really an exception handler,because "throws" keyword will bypass the exception handling responsibility from present method to the caller method.

If we want to handle the exceptions,the location where exceptions are generated then we have to use "try-catch-finally".

Syntax:

try{

**www.durgajobs.com**  
*Continuous Job Updates for every hour*

**Fresher Jobs**
**Govt Jobs**
**Bank Jobs**

**Walk-ins**
**Placement Papers**
**IT Jobs**

**Interview Experiences**

*Complete Job information across India*

```

}

catch(Exception_Name e){

}

finally{

}

```



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

# **JAVA MEANS DURGASOFT**

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
SOFTWARE SOLUTIONS

**#202 2<sup>nd</sup> FLOOR**  
[www.durgasoft.com](http://www.durgasoft.com)

**040-64512786**  
**+91 9246212143**  
**+91 8096969696**

where the purpose of try block is to include some java code where the chances of getting exceptions.

If JVM identify any exception inside "try" block then JVM will bypass flow of execution to "catch" block by skipping all the remaining instructions in try block and by passing the generated Exception object reference as parameter.

If no exception is identified in "try" block then JVM will execute completely "try" block, at the end of try block JVM will bypass flow of execution to "finally" block directly.

The main purpose of catch block is to catch the exception from try block and to display exception details on command prompt.

To display exception details on command prompt, we have to use the following three approaches.

1. e.printStackTrace()
2. System.out.println(e):

3.System.out.println(e.getMessage());

1.e.printStackTrace();

---

It will display the exception details like Exception Name,Exception Description andException Location.

2.System.out.println(e):

---

If we pass Exception object reference variable as parameter to System.out.println(-)method then JVM will access Exception class toString() method internally,it will displaythe exception details like Exception name,Exception description.

3.System.out.println(e.getMessage()):

---

Where getMessage() method will return a String contains the exception details like onlyDescription of the exception.



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**  
**JAVA MEANS DURGASOFT**  
**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
 SOFTWARE SOLUTIONS

#202 2<sup>nd</sup> FLOOR  
[www.durgasoft.com](http://www.durgasoft.com)

040-64512786  
 +91 9246212143  
 +91 8096969696

```
class Test{
    public static void main(String args[]){
        try{
            throw new ArithmeticException("My Arithmetic Exception");
        }
        catch(ArithmeticException e){
            e.printStackTrace();
        }
    }
}
```

```

System.out.println();
System.out.println(e);
System.out.println();
System.out.println(e.getMessage());
}
finally{
}
}}

```

Output:

```

java.lang.ArithmeticException: My Arithmetic Exception
at Test.main(Test.java:7)

```

```

java.lang.ArithmeticException: My Arithmetic Exception
My Arithmetic Exception

```

The main purpose of finally block is to include some Java code in order to execute irrespective of getting exception in "try" block and irrespective of executing "catch" block.



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

**JAVA MEANS DURGASOFT**

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
SOFTWARE SOLUTIONS

#202 2<sup>nd</sup> FLOOR  
[www.durgasoft.com](http://www.durgasoft.com)

040-64512786  
+91 9246212143  
+91 8096969696

Q)What is the difference between "final", "finally" and "finalize" in JAVA?

1. "final" is a keyword it can be used to declare constant expressions.



a)final variable: It will not allow modifications over its value.

b)final methods: It will not allow method overriding.

c)final class: It will not be extended.

2.finally block: It is a part in try-catch-finally syntax, it will include some instructions, which must be executed by JVM irrespective of getting exception from try block and irrespective of executing catch block.

3.finalize(): It is a method in java.lang.Object class, it will be executed before destroying objects in order to give final notification to the user about to destroy objects.



Q)Find the output from the following programs.

```
class Test{
    public static void main(String args[]){
        System.out.println("Before Try");
        try{
            System.out.println("Inside Try");
        }
        catch(Exception e){
            System.out.println("Inside Catch");
        }
    }
}
```



```

}
finally{
System.out.println("Inside Finally");
}
System.out.println("After Finally");
}}

```

Output:

Before try

Inside try

Inside finally

After finally



```

class Test{
public static void main(String args[]){
System.out.println("Before Try");
try{
System.out.println("Before Exception in try");
float f=100/0;

```

```
System.out.println("After Exception in try");  
}  
catch(Exception e){  
System.out.println("Inside Catch");  
}  
finally{  
System.out.println("Inside Finally");  
}  
System.out.println("After Finally");  
}}
```



**www.durgasoftonlinetraining.com**

**Online Training**  
**Pre Recorded Video**  
**Classes Training**  
**Corporate Training**

**Ph: +91-8885252627, 7207212427**  
**+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : durgasoftonlinetraining@gmail.com**

Output:

Before try

Before exception in try

Inside catch

Inside finally

After finally

```
class A{  
    int m1(){  
        try{  
            return 10;  
        }  
        catch(Exception e){  
            return 20;  
        }  
    }  
}
```



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**  
**JAVA MEANS DURGASOFT**  
**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

|  |  |   |
|--|--|---|
| <small>AN ISO 9001:2008 CERTIFIED</small><br><b>DURGA</b><br><small>SOFTWARE SOLUTIONS</small> | <b>#202 2<sup>nd</sup> FLOOR</b><br><b>www.durgasoft.com</b> | <b>040-64512786</b><br><b>+91 9246212143</b><br><b>+91 8096969696</b> |
|--|--|---|

```
finally{  
    return 30;  
}}}  
class Test{  
    public static void main(String args[]){  
        A a=new A();  
        int val=a.m1();  
        System.out.println(val);  
    }  
}
```

```
}}
```

Output:

30

NOTE: finally block provided return statement is the finally return statement for the method

Q) Is it possible to provide "try" block without "catch" block?

Ans: Yes, it is possible to provide try block with out catch block but by using "finally" Block.

```
try{
```

```
}
```

```
finally{
```

```
}
```

**[www.durgasoftonlinelearning.com](http://www.durgasoftonlinelearning.com)**



**Online Training**  
**Pre Recorded Video**  
**Classes Training**  
**Corporate Training**

**Ph: +91-8885252627, 7207212427**  
**+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : [durgasoftonlinelearning@gmail.com](mailto:durgasoftonlinelearning@gmail.com)**

```
class Test{
```

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

```
System.out.println("Before try");
```

```
try{
    System.out.println("Before Exception inside try");
    int i=100;
    int j=0;
    float f=i/j;
    System.out.println("After Exception inside try");
}
finally{
    System.out.println("Inside finally");
}
System.out.println("After Finally");
}}
```

Status:No Compilation Error.

Output:

Before try

Before exception inside try

Inside finally

Exception in thread "main" java.lang.ArithmeticException: /by zero

at Test.main(Test.java:11)

Reason: When JVM encounter exception in try block,JVM will search for catch block,if no catch block is identified,then JVM will terminate the program abnormally after executing finally block.

Q)Is it possible to provide "try" block with out "finally" block?



Ans: Yes, it is possible "try" block without using "finally" block but by providing "catch" block.

Ex:

```
try{
```

```
-----
```

```
-----
```

```
}
```

```
catch(Exception e){
```

```
-----
```

```
-----
```

```
}
```



Q) Is it possible to provide try-catch-finally

- a) inside try block,
- b) inside catch block and
- c) inside finally block

Ans: Yes, it is possible to provide try-catch-finally inside try block, inside catch block and inside finally block.

### Syntax-1:

---

```
try{  
    try{  
    }  
    catch(Exception e){  
    }  
    finally{  
    }  
}  
catch(Exception e){  
  
  
}  
finally{  
}
```



### Syntax-2:

---

```
try{
```

```
}  
catch(Exception e){  
    try{  
    }  
    catch(Exception e)  
    {  
    }  
    finally{  
    }  
}  
finally{  
}  
}
```

**www.durgasoftonlinetraining.com**



**Online Training  
Pre Recorded Video  
Classes Training  
Corporate Training**

**Ph: +91-8885252627, 7207212427  
+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : durgasoftonlinetraining@gmail.com**

Syntax-3:

---

```
try{  
}  
catch(Exception e){
```

```

}
finally{
try{
}
catch(Exception e){
}
finally{
}}

```



Q)Is it possible to provide more than one catch block for a single try block?

Ans: Yes, it is possible to provide more than one catch block for a single try block but with the following conditions.

1. If no inheritance relation existed between exception class names which are specified along with catch blocks then it is possible to provide all the catch blocks in any order. If inheritance relation is existed between exception class names then we have to arrange all the catch blocks as per Exception classes inheritance increasing order.

2. In general, specifying an exception class along with a catch block is not giving any guarantee to rise the same exception in the corresponding try block, but if we specify any pure checked exception along with any catch block then the corresponding "try" block must rise the same pure checked exception.

Ex1:

```
try{  
}  
catch(ArithmeticException e){  
}  
catch(ClassCastException e){  
}  
catch(NullPointerException e){  
}
```

Status: Valid Combination

Reason: No Inheritance Relation is existed between exception classes.



Ex2:

```
try{  
}  
catch(NullPointerException e){  
}
```



```
catch(ArithmeticException e){
}

catch(ClassCastException e){
}

status:Valid Combination
```

Ex3:

```
try{
}

catch(ArithmeticException e){
}

catch(RuntimeException e){
}

catch(Exception e){
}
```

Status:Valid

Reason: Inheritance Relationship

**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

# JAVA MEANS DURGASOFT

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
SOFTWARE SOLUTIONS

**#202 2<sup>nd</sup> FLOOR**  
[www.durgasoft.com](http://www.durgasoft.com)

**040-64512786**  
**+91 9246212143**  
**+91 8096969696**

Ex4:

```
try{  
}  
catch(Exception e){  
}  
catch(RuntimeException e){  
}  
catch(ArithmeticException e){  
}  
status:Invalid
```



Ex5:

```
try{  
throws new ArithmeticException("My Exception");  
}  
catch(ArithmeticException e){  
}
```

```

catch(IOException e){
}
catch(NullPointerException e){
}

```

Status: Invalid



**www.durgasoftonlinetraining.com**

**Online Training**  
**Pre Recorded Video**  
**Classes Training**  
**Corporate Training**

**Ph: +91-8885252627, 7207212427**  
**+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : durgasoftonlinetraining@gmail.com**

Ex6:

```

try{
throw new IOException("My Exception");
}
catch(ArithmeticException e){
}
catch(IOException e){
}

```

```
catch(NullPointerException e){
}

status:Valid
```

### JAVA7 Features in Exception Handling:

---

- 1.Multi Catch block
- 2.Try-with-Resources/Automatic Resources Management/Auto close able Resources

#### 1.Multi Catch block:

---

Consider the below syntax:

```
try{
}
catch(Exception e){
}
```



If we specify "Exception" class along with catch block then it able to catch and handle all the exceptions which are either same as Exception or child classes to Exception, this approach will not provide specific handling for the



specific exceptions, it will handle all the exceptions in the common way like Exception object.

If we want to handle the Exceptions separately then we have to use multiple catch blocks for a single try block.

```
try{
} catch(ArithmeticException e){
} catch(NullPointerException e){
} catch(ClassCastException e){
}
```

If we use this approach then no. of catch blocks are increased.

In Java applications, if we want to handle all the exceptions separately and by using a single catch block then we have to use "JAVA7" provided multi-catch block

#### Syntax:

---

```
try{
}
catch(Exception1 | Exception2 | .... | Exception-n e){
}
```

where Exception1, Exception2.... must not have inheritance relation otherwise Compilation error will be raised.

**www.durgasoftonlinetraining.com**



**Online Training  
Pre Recorded Video  
Classes Training  
Corporate Training**

**Ph: +91-8885252627, 7207212427  
+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : durgasoftonlinetraining@gmail.com**

**[www.durgasoftonlinelearning.com](http://www.durgasoftonlinelearning.com)**



**Online Training**  
**Pre Recorded Video**  
**Classes Training**  
**Corporate Training**

**Ph: +91-8885252627, 7207212427**  
**+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : [durgasoftonlinelearning@gmail.com](mailto:durgasoftonlinelearning@gmail.com)**

Example Programme on Multi catch block example program:

```

class Test{
public static void main(String args[]){
try{
/* int a=10;
    int b=0;
    float c=a/b;
    */
    /*java.util.Date d=null;
    System.out.println(d.toString());
    */
    int[] a={ 1,2,3,4,5};
    System.out.println(a[10]);
}
    
```

```

catch(ArithmeticException | NullPointerException |
ArrayIndexOutOfBoundsException e){

e.printStackTrace();

}}

```



**www.durgasoftonlinetraining.com**

**Online Training**  
**Pre Recorded Video**  
**Classes Training**  
**Corporate Training**

**Ph: +91-8885252627, 7207212427**  
**+91-7207212428**

**USA Ph : 4433326786**

**E-mail : durgasoftonlinetraining@gmail.com**

### Try-With-Resources/Auto Closeable Resources:

In general, in Java applications, we may use the resources like Files, Streams, Database Connections....as per the application requirements.

If we want to manage the resources along with try-catch-finally in Java applications then we have to use the following conventions.

1. Declare all the resources before "try" block.
2. Create the resources inside "try" block.
3. Close the resources inside finally block.

The main intention to declare the resources before "try" block is to make available resources variables to "catch" block and to "finally" block to use.

If we want to close the resources in "finally" block then we have to use close() methods, which are throwing some exceptions like

IOException,SQLException depending on the resource by using "throws" keyword, to handle these exceptions we have to use "try-catch-finally" inside "finally" block.

```
//Declare the resources
```

```
File f=null;
```

```
BufferedReader br=null;
```

```
Connection con=null;
```

```
try{
```

```
//create the resources
```

```
f=new File("abc.txt");
```

```
br=new BufferedReader(new InputStreamReader(System.in));
```

```
con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");
```

```
-----
```

```
-----
```

```
}
```



```
catch(Exception e){
```

```
}
```



```

finally{

//close the resources

try{

f.close();

br.close();

con.close();

}catch(Exception e){

e.printStackTrace();

}}

```



**LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...**

# **JAVA MEANS DURGASOFT**

**INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
SOFTWARE SOLUTIONS

**#202 2<sup>nd</sup> FLOOR**  
**www.durgasoft.com**

**040-64512786**  
**+91 9246212143**  
**+91 8096969696**

To manage the resources in Java applications, if we use the above convention then developers have to use close() methods explicitly. Developers have to provide try-catch-finally inside "finally" block, this convention will increase no. of instructions in Java applications.

To overcome all the above problems, JAVA 7 version has provided a new Feature in the form of "Try-With-Resources" or "Auto Closeable Resources".

In the case of "Try-With-Resources", just we, have to declare and create the resources along with "try" [not inside try block, not before try block] and no need to close these resources inside the finally block, why because, JVM will close all the resources automatically when flow of execution is coming out from "try" block.



In the case of "Try-With-Resources", it is not required to close the resources explicitly, it is not required to use close() methods in finally block explicitly, so that it is not required to use "finally" block in try-catch-finally syntax.

**[www.durgasoftonlinelearning.com](http://www.durgasoftonlinelearning.com)**



**Online Training  
Pre Recorded Video  
Classes Training  
Corporate Training**

**Ph: +91-8885252627, 7207212427  
+91-7207212428**

 **USA Ph : 4433326786**

**E-mail : [durgasoftonlinelearning@gmail.com](mailto:durgasoftonlinelearning@gmail.com)**

Syntax:

---

```
try(Resource1; Resource2; .....Resource-n){
-----
-----
}
catch(Exception e){
-----
-----
}
```

Where all the specified Resources classes or interfaces must implement or extend "java.io.AutoCloseable" interface.

Where if we declare resources as AutoCloseable resources along with "try" then the resourcesreference variables are converted as "final" variables.

Ex:

```
try(File f=new File("abc.txt");
```

```
BufferedReader br=new BufferedReader(new  
InputStreamReader(System.in));
```

```
Connection
```

```
con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");)
```

```
{
```

```
-----
```

```
-----
```

```
}
```



```
catch(Exception e){
```

```
e.printStackTrace(); }
```

**NOTE:** In Java, all the predefined Stream classes, File class, Connection interface are extends/implemented "java.io.AutoCloseable" interface predefinedly.

```
import java.io.*;
```

```
class Test{  
    public static void main(String args[]){  
        try(FileInputStream fis=new FileInputStream("ScannerEx.java")  
            FileOutputStream fos=new FileOutputStream("DynamicEx2.java");)  
        {  
            int size=fis.available();  
            byte[] b=new byte[size];  
            fis.read(b);  
            fos.write(b);  
            System.out.println("File Transferred From ScannerEx.java to  
            DynamicEx2.java");  
        }  
    }  
}
```



```
catch(Exception e){  
    e.printStackTrace();  
}}}
```

Custom Exceptions/User Defined Exceptions:

Custom Exceptions are the exceptions, which would be defined by the developers as per their application requirements.

If we want to define user defined exceptions then we have to use the following steps:

### 1. Define User defined Exception class:

---

To declare user-defined Exception class, we have to take an user-defined class, which must be extended from `java.lang.Exception` class.

Class `MyException` extends `Exception`

```
{
}
```



2. Declare a String parametrized constructor in User-Defined Exception class and access String parametrized super class constructor by using "super" keyword:

---

```
class MyException extends Exception{
    MyException(String err_Msg){
        super(err_Msg);
    }
}
```

```
}}
```

3. Create and Rise exception in Java application as per the application requirement:

---

```
try{
throw new MyException("My Custom Exception");
}
catch(MyException me){
me.printStackTrace();
}
```



```
class InsufficientFundsException extends Exception{
InsufficientFundsException(String err_Msg){
super(err_Msg);
}}

class Transaction{
String accNo;

String accName;
```



```

String accType;

int initial_Amt=10000;

Transaction(String accNo,String accName,String accType){

this.accNo=accNo;

this.accName=accName;

this.accType=accType;

}

```



```

public void withdraw(int wd_Amt){
try{
System.out.println("Transaction Details");
System.out.println("-----");
System.out.println("Transaction Id   :T123");
System.out.println("Account Number  :"+accNo);
System.out.println("Account Name   :"+accName);
System.out.println("Account Type   :"+accType);
System.out.println("Transaction Type : WITHDRAW");
System.out.println("Initial Amount  :"+initial_Amt);
}
}

```

```

System.out.println("Withdrawl Amount : "+wd_Amt);

int total_Amt=0;

if(wd_Amt<initial_Amt)

total_Amt=initial_Amt-wd_Amt;

initial_Amt=total_Amt;

System.out.println("Total Amount : "+total_Amt);

System.out.println("Transaction Status : SUCCESS");

}

else{

total_Amt=initial_Amt;

System.out.println("Total Amount : "+total_Amt);

System.out.println("Transaction Status:FAILURE");

throw new InsufficientFundsException("Amount is not sufficient in your
Account");

}

}

```



```

catch(InsufficientFundsException e){

```

```
System.out.println(e.getMessage());
}
System.out.println("*****ThankQ,Visit Again*****");
}
class Test{
public static void main(String args[]){
Transaction tx1=new Transaction("abc123","Durga","Savings");
tx1.withdraw(5000);
System.out.println();
Transaction tx2=new Transaction("xyz123","Anil","Savings");
tx2.withdraw(15000);
}}
```

**LEARN FROM EXPERTS ...**

**COMPLETE JAVA**  
CORE JAVA, ADV. JAVA, ORACLE, STRUTS, HIBERNATE, SPRING, WEB SERVICES,...

**COMPLETE .NET**  
C#.NET, ASP.NET, SQL SERVER, MVC 5 & WCF

**TESTING TOOLS**  
MANUAL + SELENIUM

**ORACLE | D2K**

**MSBI | SHARE POINT**

**HADOOP | ANDROID**

**C, C++ , DS, UNIX**

**CRT & APTITUDE TRAINING**

AN ISO 9001:2008 CERTIFIED  
**DURGA**  
Software Solutions®

# 202, 2nd Floor, HUDA Maitrivanam,  
Ameerpet, Hyd. Ph: 040-64512786,  
**9246212143, 8096969696**

**www.durgasoft.com**