Java means DURGA SOFT..

CORE JAVA

Material

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Exception Handling

Information regarding Exception:-

- ❖ Dictionary meaning of the exception is abnormal termination.
- ❖ An expected event that disturbs or terminates normal flow of execution called exception.
- ❖ If the application contains exception then the program terminated abnormally the rest of the application is not executed.
- ❖ To overcome above limitation in order to execute the rest of the application must handle the exception.

In java we are having two approaches to handle the exceptions.

1) ***By using try-catch block.***
2) ***By using throws keyword.***

Exception Handling:-

- ✓ The main objective of exception handling is to get normal termination of the application in order to execute rest of the application code.
- ✓ Exception handling means just we are providing alternate code to continue the execution of remaining code and to get normal termination of the application.
- ✓ Every Exception is a predefined class present in different packages.

  java.lang.ArithmeticException
  java.io.IOException
  java.sql.SQLException
  javax.servlet.ServletException

The exception are occurred due to two reasons

a. Developer mistakes
b. End-user mistakes.
   i. While providing inputs to the application.
   ii. Whenever user is entered invalid data then Exception is occur.
   iii. A file that needs to be opened can't found then Exception is occurred.
   iv. Exception is occurred when the network has disconnected at the middle of the communication.

Types of Exceptions:-

As per the sun micro systems standards The Exceptions are divided into three types

1) ***Checked Exception***
2) ***Unchecked Exception***
3) ***Error***

checked Exception:-

- The Exceptions which are checked by the compiler at the time of compilation is called Checked Exceptions.

  IOException,SQLException,InterruptedException……..etc

- If the application contains checked exception the code is not compiled so must handle the checked Exception in two ways
  - *By using try-catch block.*
  - *By using throws keyword.*
- If the application contains checked Exception the compiler is able to check it and it will give intimation to developer regarding Exception in the form of compilation error.

There are two types of predefined methods

- ✓ Exceptional methods

  public static native void sleep(long)***throws java.lang.InterruptedException***
  publicbooleancreateNewFile() ***throws java.io.IOException***
  public abstract java.sql.StatementcreateStatement() ***throws java.sql.SQLException***

- ✓ Normal methods

  public long length();
  publicjava.lang.StringtoString();

In our application whenever we are using exceptional methods the code is not compiled because these methods throws checked exception hence must handle the exception by using try-catch or throws keywords.

Checked Exception scenario:-

Unchecked Exception:-

- The exceptions which are not checked by the compiler at the time of compilation are called unchecked Exception.

  ArithmeticException,ArrayIndexOutOfBoundsException,NumberFormatException….etc
- If the application contains un-checked Exception code is compiled but at runtime JVM(Default Exception handler) display exception message then program terminated abnormally.
- To overcome runtime problem must handle the exception in two ways.
  - *By using try-catch blocks.*
  - *By using throws keyword.*

Example :- different types of unchecked exceptions.

```
class Test
{       public static void main(String[] args)
        {       //java.lang.ArithmeticException: / by zero
                System.out.println(10/0);

                //java.lang.ArrayIndexOutOfBoundsException
                int[] a={10,20,30};
                System.out.println(a[5]);

                //java.lang.StringIndexOutOfBoundsException
                System.out.println("ratan".charAt(10));
        }
}
```

Note-1:-

If the application contains checked exception compiler generate information about exception so code is not compiled hence must handle that exception by using try-catch block or throws keyword it means for the checked exceptions try-catch blocks or throws keyword mandatory *but* ***if the application contains un-checked exception try-catch blocks or throws keyword is optional it means code is compiled but at runtime program is terminated abnormally.***

Note 2:-

In java whether it is a checked Exception or unchecked Exception must handle the Exception by using try-catch blocks or throws keyword to get normal termination of application.

<u>Note-3:-</u>

In java whether it is checked Exception or unchecked exceptions are occurred at runtime but not compile time.

<u>Error:-</u>

- ➔ Errors are caused due to lack of system resources like,
    - o *Heap memory full.*
    - o *Stack memory problem.*
    - o *AWT component problems…..etc*

  Ex: - StackOverFlowError, OutOfMemoryError, AssertionError…………etc
- ➔ Exceptions are caused due to developers mistakes or end user supplied inputs but errors are caused due to lack of system resources.
- ➔ *We are handle the exceptions by using try-catch blocks or throws keyword but we are unable to handle the errors.*

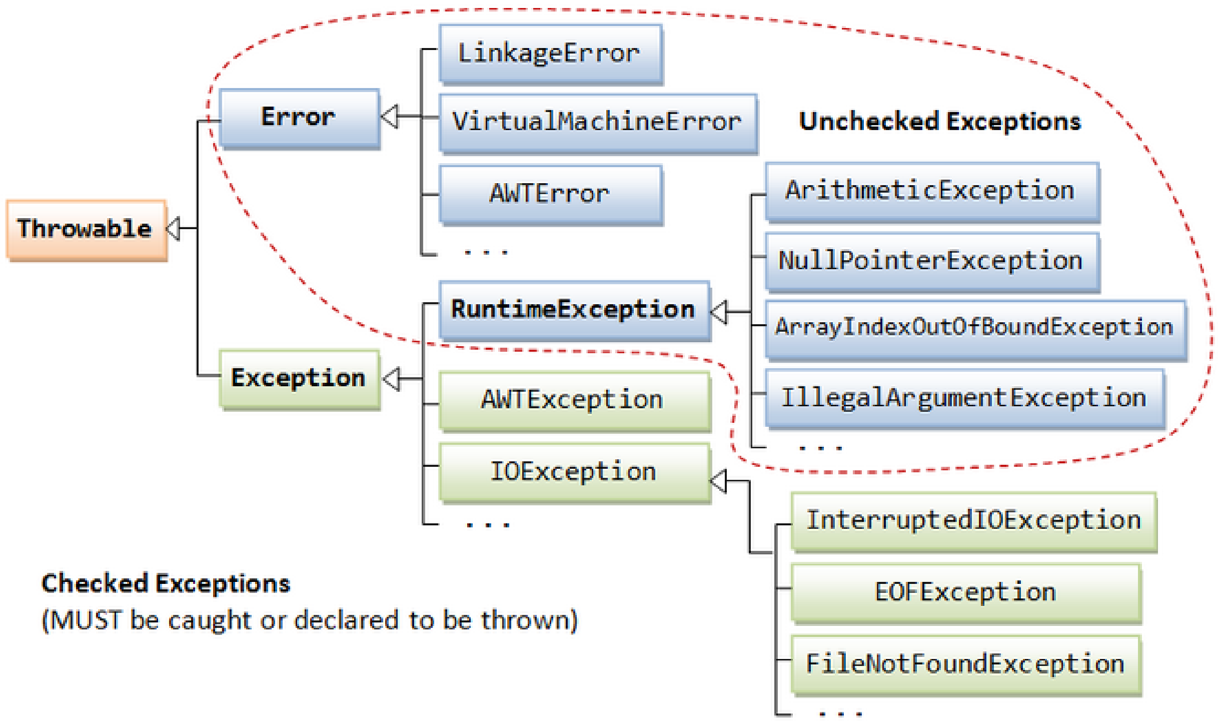
Example:-

```
class Test
{           public static void main(String[] args)
            {           Test[] t = new Test[100000000];
            }
};
```

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

<u>Exception Handling Tree Structure:-</u>

Root class of exception handling is Throwable class



- **In above tree Structure RuntimeException its child classes and Error its child classes are Unchecked remaining all exceptions are checked Exceptions.**

Exception handling key words:-

1) try
2) catch
3) finally
4) throw
5) throws

Exception Handling:-

In java wether it is a checked Excepiton or unchecked Exception must handle the Exception by using try-catch blocks or throws to get normal termination of application.

Exception handling by using Try –catch block:-

Syntax:-

```
try
{
        exceptional code;
}
catch (ExceptionNamereference_variable)
{
        Code to run if an exception is raised;
}
```

Example -1:-

Application without try-catch

```
class Test
{
        public static void main(String[] args)
        {
                System.out.println("rattan 1st class");
                System.out.println("rattan 2st class");
                System.out.println("rattan inter");
                System.out.println("rattan trainer");
                System.out.println("rattan weds anushka"+(10/0));
                System.out.println("rattan  kids");
        }
}
```

```
D:\>java Test
rattan 1st class
rattan 2st class
rattan inter
rattan trainer
```

$$\underbrace{\textbf{Exception in Thread “main”}}_{\textbf{Handled by JVM}}\ \underbrace{\textbf{java.lang.ArithmeticException:}}_{\textbf{type of the Exception}}\ \underbrace{\textbf{/ by zero}}_{\textbf{description}}$$

Application with try-catch blocks:-

1) Whenever the exception is raised in the try block JVM won’t terminate the program immidiatly it will search corresponding catch block.
    a. If the catch block is matched that will be executed then rest of the application executed and program is terminated normally.
    b. If the catch block is not matched program is terminated abnormally.

```
class Test
{
        public static void main(String[] args)
        {
                System.out.println("ratan 1st class");
                System.out.println("ratan 2st class");
                System.out.println("ratan inter");
                System.out.println("ratan trainer");
                try
                {       //Exceptional code
                        System.out.println("ratan weds anushka"+(10/0));
                }
                catch (ArithmeticExceptionae)
                {       //alternate code
                        System.out.println("ratan weds aruna");
                }
                System.out.println("ratan  kids");
        }
}
```

```
D:\>java Test
ratan 1st class
ratan 2st class
ratan inter
ratan trainer
ratan weds aruna
ratan  kids
```

Example :-

- If the exceptions raised in try block JVM will search for corresponding catch block,
    - If the catch block is matched corresponding catch is executed then rest of the application is executed & program terminated normally.
    - ***If the catch block is not matched program is terminated abnormally the rest of the application is not executed.***

```
class Test
{           public static void main(String[] args)
            {           try
                        {           System.out.println("sravya");
                                    System.out.println(10/0);
                        }
                        catch(NullPointerException e)
                        {           System.out.println(10/2);
                        }
                        System.out.println("rest of the app");
            }
}
```

```
E:\sravya>java Test
sravya
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

Example-3:-*If there is no exception in try block the catch blocks are not checked.*

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

```
                {           System.out.println("sravya");
                            System.out.println("anu");
                }
                catch(NullPointerException e)
                {           System.out.println(10/2);
                }
                System.out.println("rest of the app");
        }
}
```

```
E:\sravya>java Test
sravya
anu
rest of the app
```

Example:-

*in Exception handling independent try blocks are not allowed must declare **try-catch** or**try-finally** or **try-catch-finally.***

```
class Test
{           public static void main(String[] args)
            {           try
                        {           System.out.println("sravya");
                                    System.out.println("anu");
                        }
                        System.out.println("rest of the app");
            }
}
```

```
E:\sravya>javac Test.java
Test.java:4: 'try' without 'catch' or 'finally'
```

Example:-

In between try-catch blocks it is not possible to declare any statements must declare try with immediate catch block.

```
class Test
{	public static void main(String[] args)
	{	try
		{	System.out.println("sravya");
			System.out.println(10/0);
		}
		System.out.println("anu");
		catch(ArithmeticException e)
		{	System.out.println(10/2);
		}
		System.out.println("rest of the app");
	}
}
```

Example:-

- *If the exception raised in try block jvm will search corresponding catch block.*
- *If the exception raised other than try block it is always abnormal termination.*
- *In below example exception raised in catch block hence program is terminated abnormally.*

```
class Test
{	public static void main(String[] args)
	{	try
		{	System.out.println("sravya");
			System.out.println(10/0);
		}
		catch(ArithmeticException e)
		{	System.out.println(10/0);
		}
		System.out.println("rest of the app");
	}
}
E:\sravya>java Test
sravya
Exception in thread "main" java.lang.ArithmeticException: / by zero
atTest.main(Test.java:9)
```

Example:-

- If the exception raised in try block remaining code of try block won't be executed.

1) Once the control is out of the try block the control never entered into try block once again.

```
class Test
{           public static void main(String[] args)
            {           try
                        {           System.out.println(10/0);
                                    System.out.println("sravya");
                                    System.out.println("ratan");
                        }
                        catch(ArithmeticException e)
                        {           System.out.println(10/2);
                        }
                        System.out.println("rest of the app");
            }
}
E:\sravya>java Test
5
rest of the app
```

Example 8:-

The way of handling the exception is varied from exception to the exception hence it is recommended to provide try with multiple number of catch blocks.

```
importjava.util.*;
class Test
{           public static void main(String[] args)
            {           Scanner s=new Scanner(System.in);         //Scanner object used to take dynamic input
                        System.out.println("provide the division value");
                        int n=s.nextInt();
                        try
                        {           System.out.println(10/n);
                                    String str=null;
                                    System.out.println("u r name is  :"+str);
                                    System.out.println("u r name length is--->"+str.length());
                        }
                        catch (ArithmeticExceptionae)
                        {           System.out.println("good boy zero not allowed geting Exception"+ae);
                        }
                        catch (NullPointerException ne)
                        {           System.out.println("good girl getting Exception"+ne);
                        }
                        System.out.println("rest of the code");
            }
}
```

Output:- provide the division value: 5

Write the output

Output:- provide the division value: 0
Write the output

Example-9:- **By using Exceptional catch block we are able to hold any type of exceptions.**

```
importjava.util.*;
class Test
{	public static void main(String[] args)
	{	Scanner s=new Scanner(System.in);
		System.out.println("provide the division value");
		int n=s.nextInt();
		try
		{	System.out.println(10/n);
			String str=null;
			System.out.println("u r name is  :"+str);
			System.out.println("u r name length is--->"+str.length());
		}
		catch (Exception e)//this catch block is able to handle all  types  of Exceptions
	{System.out.println("I am an inexperienced or lazy programmer here="+e);
		}
		System.out.println("rest of the code");
	}
}
```

Example -10:-if we are declaring multiple catch blocks at that situation the catch block order should be child to parent shouldn't be parent to the child.

(No compilation error)

Child-parent

```
importjava.util.*;
class Test
{	public static void main(String[] args)
	{
Scanner s=new Scanner(System.in);
System.out.println("provide the division val");
int n=s.nextInt();
try
{
System.out.println(10/n);
String str=null;
System.out.println(str.length());
}
catch (ArithmeticExceptionae)
{
System.out.println("Exception"+ae);
}
catch (Exception ne)
{
System.out.println("Exception"+ne);
}
System.out.println("rest of the code");
	}
}
```

Compilation error

```
importjava.util.*;
class Test
{	public static void main(String[] args)
	{
	Scanner s=new Scanner(System.in);
System.out.println("provide the division val");
		int n=s.nextInt();
		try
		{
	System.out.println(10/n);
	String str=null;
	System.out.println(str.length());
		}
		catch (Exception ae)
		{
	System.out.println("Exception"+ae);
		}
		catch (ArithmeticException ne)
		{
	System.out.println("Exception"+ne);
```

```
                }
        System.out.println("rest of the code");
```

Possibilities of try-catch blocks:-

Possibility-1

```
try  {    }
catch ()  {   }
```

Possibility-2

```
try
{           }
catch ()
{           }
try
{           }
catch ()
{           }
```

Possibility-3

```
try
{           }
catch ()  { }
catch () { }
```

Possibility-4

```
try
{           try
            {           }
            catch ()
            {           }
}
catch ()    {     }
```

```
            }
}
```

Possibility-5

```
try
{           }
catch ()
{           try
            {           }
            catch ()
            {           }
}
```

Possibility-6

```
try
{           try
            {           }
            catch ()
            {           }
}
catch ()
{           try
            {           }
            catch ()
            {           }
}
```

Example 11:-

It is possible to combine two exceptions in single catch block the syntax is catch(ArithmeticException | StringIndexOutOfBoundsException a) .

```
importjava.util.Scanner;
public class Test
{   public static void main(String[] args)
  {   Scanner s = new Scanner(System.in);
            System.out.println("enter a number");
            int n = s.nextInt();
                    try {
                            System.out.println(10/n);
                            System.out.println("ratan".charAt(13));
                            }
                    catch(ArithmeticException | StringIndexOutOfBoundsException a)
                    {           System.out.println("ratansoft");
                    }
                    System.out.println("Rest of the application");
  }
}
```

```
D:\DP>java Test                          D:\DP>java Test
enter a number                           enter a number
0                                        2
ratansoft                                5
Rest of the application                  ratansoft
                                         Rest of the application
```

Finally block:-

1) Finally block is always executed irrespective of try and catch.
2) It is used to provide clean-up code
   a. Database connection closing. **Connection.close();**
   b. streams closing. **Scanner.close();**
   c. Object destruction . **Test t = new Test();t=null;**
3) It is not possible to write finally alone.

a. try-catch-finally	*--→valied*
b. try-catch	*--→valied*
c. catch-finally	*--→invalied*
d. try-catch-catch-finally	*--→valied*
e. try-finally	*--→valied*
f. catch-catch-finally	*--→invalied*
g. Try	*--→invalied*
h. Catch	*--→invalied*
i. Finally	*-→invalied*

Syntax:-

```
try
{   risky code;
}
catch (Exception obj)
```

```
{       handling code;
}
finally
{       Clean-up code;(database connection closing,streams closing……etc)
}
```

Example:-

if the exception raised in try block the JVM will search for corresponding catch block ,

- If the corresponding catch block is matched the catch block is executed then finally block is executed.
- If the corresponding catch block is not matched the program is terminated abnormally just before abnormal termination the finally block will be executed then program is terminated abnormally.

All possibilities of finally block execution :-

Case 1:-

```
try
{           System.out.println("try");
}
catch (ArithmeticExceptionae)
{           System.out.println("catch");
}
finally
{           System.out.println("finally");
}
```

Output:-

```
try
finally
```

case 2:-

```
try
{           System.out.println(10/0);
}
catch (ArithmeticExceptionae)
{           System.out.println("catch");
}
finally
{           System.out.println("finally");
}
```

Output:-

```
catch
finally
```

case 3:-

```
try
{           System.out.println(10/0);
}
catch (NullPointerExceptionae)
{           System.out.println("catch");
}
finally
```

```
        {       System.out.println("finally");
        }
```

Output:
```
finally
Exception in thread "main"
java.lang.ArithmeticException: / by zero
atTest.main(Test.java:4)
```

case 4:-
```
        try
        {       System.out.println(10/0);
        }
        catch (ArithmeticExceptionae)
        {       System.out.println(10/0);
        }
        finally
        {       System.out.println("finally");
        }
```
```
D:\morn11>java Test
finally
Exception in thread "main"
java.lang.ArithmeticException: / by zero
atTest.main(Test.java:7)
```

case 5:-
```
        try
        {       System.out.println("try");
        }
        catch(ArithmeticExceptionae)
        {       System.out.println("catch");
        }
        finally
        {       System.out.println(10/0);
        }
        System.out.println("rest of the code");
```
```
D:\>java Test
try
Exception in thread "main"
java.lang.ArithmeticException: / by zero
atTest.main(Test.java:15)
```

case 6:-it is possible to provide try-finally.
```
        try
        {       System.out.println("try");
        }
        finally
        {       System.out.println("finally");
        }
        System.out.println("rest of the code");
```
```
D:\>java Test
try
finally
rest of the code
```

Example:-in only two cases finally block won't be executed

Case 1:- whenever we are giving chance to try block then only finally block will be executed otherwise it is not executed.

```
class Test
{	public static void main(String[] args)
	{	System.out.println(10/0);
		try
		{	System.out.println("ratan");
		}
		finally
		{	System.out.println("finally block");
		}
		System.out.println("rest of the code");
	}
};
D:\>java Test
Exception in thread "main" java.lang.ArithmeticException: / by zero
atTest.main(Test.java:5)
```

Case 2:-In your program whenever we are using System.exit(0) the JVM will be shutdown hence the rest of the code won't be executed .

```
class Test
{	public static void main(String[] args)
	{	try
		{	System.out.println("ratan");
			System.exit(0);
		}
		finally
		{	System.out.println("finally block");
		}
		System.out.println("rest of the code");
	}
};
D:\>java Test
Ratan
```

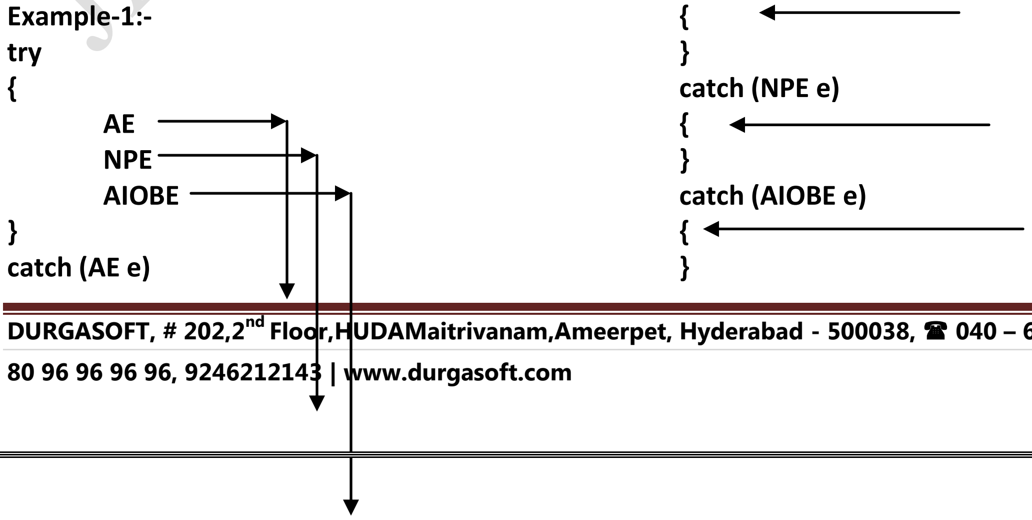
Methods to print Exception information:-

```
class Test1
{	void m1()
	{	m2();	}
	void m2()
	{	m3();	}
	void m3()
	{	try{
		System.out.println(10/0);}
```

```
        catch(ArithmeticExceptionae)
        {
        System.out.println(ae.toString());
        System.out.println(ae.getMessage());
        ae.printStackTrace();
        }
        }
        public static void main(String[] args)
        {       Test1 t = new Test1();
                t.m1();
        }
};
D:\DP>java Test1
java.lang.ArithmeticException: / by zero     //toString() method output
/ by zero                                    //getMessage() method output
java.lang.ArithmeticException: / by zero     //printStackTrace() method
at Test1.m3(Test1.java:8)
at Test1.m2(Test1.java:5)
at Test1.m1(Test1.java:3)
at Test1.main(Test1.java:17)
```

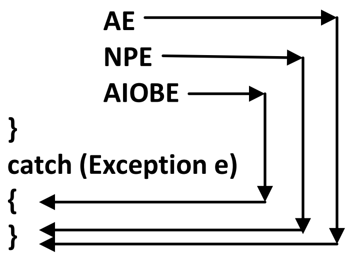
Possibilities of exception handling:-

Example-1:-



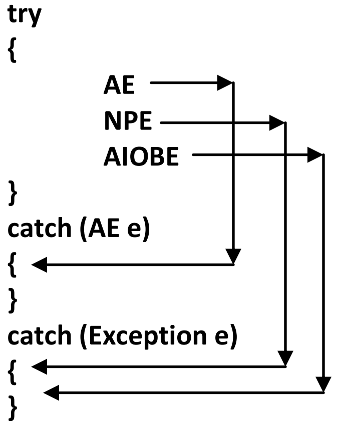
```
try                          {
{                            }
        AE                   catch (NPE e)
        NPE                  {
        AIOBE                }
}                            catch (AIOBE e)
catch (AE e)                 {
                             }
```

Example-2:-



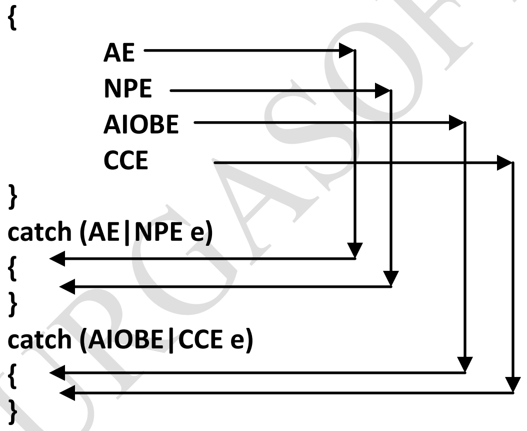
```
try
{
        AE
        NPE
        AIOBE
}
catch (Exception e)
{
}
```

Example-3:-



```
try
{
        AE
        NPE
        AIOBE
}
catch (AE e)
{
}
catch (Exception e)
{
}
```

Ex 4:-introduced in 1.7 version



```
try
{
        AE
        NPE
        AIOBE
        CCE
}
catch (AE|NPE e)
{
}
catch (AIOBE|CCE e)
{
}
```

Example :-

```
statement 1
statement 2
try
{       statement 3
        try
        {       statement 4
                statement 5
        }
        catch ()
        {       statement 6
                statement 7
        }
}
catch ()
{       statement 8
        statement 9
        try
        {       statement 10
                statement 11
        }
        catch ()
        {       statement 12
                statement 13
```

```
            }
}
Finally{
statement 14
statement 15
}
Statement -16
Statement -17
```

case 1:- **if there is no Exception in the above example**
1, 2, 3, 4, 5, 14, 15 Normal Termination

Case 2:- **if the exception is raised in statement 2**
1 ,Abnrmal Termination

Case 3:- **if the exception is raised in the statement 3 the corresponding catch block is matched.**
1,2,8,9,10,11,14,15 normal termination

Case 4:- **if the exception is raise in the statement-4 the corresponding catch block is not matched and outer catch block is not matched.**
1,2,3 abnormal termination.

Case 5:- **If the exception is raised in the statement 5 and corresponding catch block is not matched and outer catch block is matched.**
1,2,3,4,8,9,10,11,14,15 normal termination

Case 6:- **If the exception is raised in the statement 5 and the corresponding catch block is not matched and outer catch block is matched while executing outer catch inside the try block the exception is raised in the statement 10 and the corresponding catch is matched.**
1,2,3,4,8,9,12,13,14,15 normal termination.

Case 7:- **If the exception raised in statement 14.**
1,2,3,4,5 abnormal termination.

Case 8:- if the Exception raised in statement 17.

Throws :-

1) In the exception handling must handle the exception in two ways
   a. By using try-catch blocks.
   b. By using throws keyword.
2) Try-catch block is used to handle the exception but throws keyword is used to **delegate** the responsibilities of the exception handling to the caller method.
3) The main purpose of the throws keyword is **bypassing** the generated exception from present method to caller method.
4) Use throws keyword at method declaration level.
5) It is possible to throws any number of exceptions at a time based on the programmer requirement.
6) If main method is throws the exception then JVm is responsible to handle the exception.

By using try-catch blocks:-

```
import java.io.*;
class Student
{       voidstudentDetails()
        {
        try
        {
BufferedReaderbr=new BufferedReader(new
InputStreamReader(System.in));
System.out.println(" enter student name");
String sname=br.readLine();
System.out.println("u r name is:"+sname);
        }
        catch(IOException e)
        {
System.out.println(" getting Exception"+e);
        }
        }
        public static void main(String[] args)
        {
                Student s1=new Student();
                s1.studentDetails();
}}
```

Handling the exception by using throws keyword:-

Ex 1:-

```
import java.io.*;
class Student
{
voidstudentDetails()throws IOException
{
BufferedReaderbr=new BufferedReader(new
InputStreamReader(System.in));
System.out.println(" enter student name");
String sname=br.readLine();
System.out.println("u r name is:"+sname);
        }
public static void main(String[] args)throws
IOException
        {
                Student s1=new Student();
                s1.studentDetails();
        }
}
```

Ex ample:-

```
import java.io.*;
class Student
{
voidstudentDetails()throws IOException        //(delegating responsibilities to caller method principal())
        {       BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
                System.out.println("please enter student name");
                String sname=br.readLine();
                System.out.println("please enter student rollno");
                intsroll=Integer.parseInt(br.readLine());
                System.out.println("enter student address");
                String saddr=br.readLine();
                System.out.println("student name is:"+sname);
                System.out.println("student rollno is:"+sroll);
                System.out.println("student address is:"+saddr);
        }
void principal() throws IOException//(delegating responsibilities to caller method    officeBoy())
        {       studentDetails();
        }
voidofficeBoy()throws IOException//(delegating responsibilities to caller method    main())
        {       principal();
        }
public static void main(String[] args) throws IOException   ///(delegating responsibilities to  JVM)
        {       Student s1=new Student();
                s1.officeBoy();
        }
}
```

Throw:-

1) The main purpose of the throw keyword is to creation of Exception object explicitly either for predefined or user defined exception.
2) Throw keyword works like a try block. The difference is try block is automatically find the situation and creates an Exception object implicitly. Whereas throw keyword creates an Exception object explicitly.
3) Throws keyword is used to delegate the responsibilities to the caller method but throw is used to create the exception object.
4) If exception object created by JVM it will print predefined information **(/ by zero)** but if exception Object created by user then user defined information is printed.
5) We are using throws keyword at method declaration level but throw keyword used at method implementation (body) level.

throw keyword having two objectives

1. Handover the user created exception object JVM for predefined Exception.
2. Handover the user created exception object JVM for user defined Exception.

Ex:-Objective-1 of the throw keyword

throw keyword is used to create the exception object explicitly by the developer for predefined exceptions.

Step -1 :- create the Exception object explicitly by the developer.

newArithmeticException("ratan not eligible");

Step -2:- handover user created Exception object to jvm by using throw keyword.

throw newArithmeticException("ratan not eligible");

```
importjava.util.*;
class Test
{	static void validate(int age)
	{	if (age<18)
		{
		//creating Exception object by user & handover to Jvm
		throw new ArithmeticException("not eligible for vote");
		}
		else
		{	System.out.println("welcome to the voting");
		}
	}
	public static void main(String[] args)
	{	Scanner s=new Scanner(System.in);
		System.out.println("please enter your age ");
		int n=s.nextInt();
		validate(n);
		System.out.println("rest of the code");
	}
}
```

Objective-2 :- throw keyword is used to create the exception object explicitly by the developer for the user defined exceptions.

There are two types of exceptions present in the java language

1) Predefined Exceptions.
2) User defined Exceptions.

Predefined Exception:-

These exceptions are introduced by James Gosling comes along with software.

Ex:-ArithmeticException,IOException,NullPointerException…………..etc

User defined Exceptions:-

Exceptions created by user are called userdefined Exceptions.

Ex: InvalidAgeException,BombBlostException………..etc

Step-1: create user defined Exception.

```
classInvaliedAgeException extends Exception
{
};
```

Step-2:- create the object of user defined Exception.

```
newInvaliedAgeException();
```

step-3:- handover user defined Exception object to Jvm by using throw keyword.

```
thrownew InvaliedAgeException();
```

creation of user defined Exceptions:-(customization of Exceptions)
there are two types of user defined exceptions

i) User defined checked Exception (**these Exceptions are extends Exception class**)
ii) User defined un-checked Exception (**extends RuntimeException class**)

The naming conventions are every exception suffix must be the word Exception.

Creation of Userdefined checked Exception:-

There are two approaches to create Userdefined checked Exception

1. *Default constructor approach*
2. Parameterized constructor approach

Creation ofuserdefined checked Exception by using default constructor approach:-

Step-1:- create the user defined Exception

Normal java class will become Exception class whenever we are extends Exception class.

InvaliedAgeException.java:-

```
packagecom.tcs.userexceptions;
public class InvalidAgeExcepiton extends Exception
{        //default constructor
};
```

Note: - in this example we are creating user defind checked Exception hence must handle the Exception by using try-catch or throws keyword otherwise compiler generate compilation error "**unreportedException**"

Step-2:- use created Exception in our project.

Project.java

```
packagecom.tcs.project;
importcom.tcs.userexceptions.InvalidAgeExcepiton;
importjava.util.Scanner;
class Test
{        static void status(int age)throws InvalidAgeExcepiton
         {        if (age>25)
                  {System.out.println("eligible for mrg");
                  }
                  else
                  {        //using user created Exception
                           throw new InvalidAgeExcepiton();//default constructor executed
                  }
```

```
        }
        public static void main(String[] args)throws InvalidAgeExcepiton
        {       Scanner s = new Scanner(System.in);
                System.out.println("enter u r age");//23
                int age = s.nextInt();
                Test.status(age);
        }
}
```

```
D:\morn11>javac -d . InvalidAgeExcepiton.java
D:\morn11>javac -d . Test.java
D:\morn11>java com.tcs.project.Test
enter u r age
19
Exception in thread "main" com.tcs.userexceptions.InvalidAgeExcepiton
atcom.tcs.project.Test.status(Test.java:11)
atcom.tcs.project.Test.main(Test.java:18)
```

creation of userdefined checked exception by using parametarized constructor approach:-

step-1:- create the userdefined exception class.

InvalidAgeException.java

```
packagecom.tcs.userexceptions;
public class InvalidAgeExcepiton extends Exception
{       publicInvalidAgeExcepiton(String str)
        {//super constructor calling inorder to print your information
        super(str);
        }
};
```

Step-2:- use user created Exception in our project.

Project.java

```
packagecom.tcs.project;
importcom.tcs.userexceptions.InvalidAgeExcepiton;
importjava.util.Scanner;
class Test
{       static void status(int age)throws InvalidAgeExcepiton
```

```
        {       if (age>25)
                {System.out.println("eligible for mrg");
                }
                else
                {       //using user created Exception
                throw new InvalidAgeExcepiton("not eligible try after some time");
                }
        }
        public static void main(String[] args)throws InvalidAgeExcepiton
        {       Scanner s = new Scanner(System.in);
                System.out.println("enter u r age");
                int age = s.nextInt();
                Test.status(age);
        }
}
```

```
D:\morn11>javac -d . InvalidAgeExcepiton.java
D:\morn11>javac -d . Test.java
D:\morn11>java com.tcs.project.Test
enter u r age
28
eligible for mrg
D:\morn11>java com.tcs.project.Test
enter u r age
20
Exception in thread "main" com.tcs.userexceptions.InvalidAgeExcepiton: not eligible try after some
time
atcom.tcs.project.Test.status(Test.java:11)
atcom.tcs.project.Test.main(Test.java:18)
```

Ex:- ***creation of user defined un-checked exception by using default constructor approach:-***

Step-1:- create userdefined exception.

InvalidAgeException.java

```
//InvalidAgeException.java
packagecom.tcs.userexceptions;
public class InvalidAgeExcepiton extends RuntimeException
{
        //default constructor
};
```

Note: - in this example we are creating user defined unchecked exception so try-catch blocks and throws keywords are optional.

Step-2:- use user created Exception in our project.

Project.java

```
packagecom.tcs.project;
importcom.tcs.userexceptions.InvalidAgeExcepiton;
importjava.util.Scanner;
class Test
{       static void status(int age)
```

```
        {       if (age>25)
                {System.out.println("eligible for mrg");
                }
                else
                {//useing user created Exception
                        throw new InvalidAgeExcepiton();
                }
        }
        public static void main(String[] args)
        {       Scanner s = new Scanner(System.in);
                System.out.println("enter u r age");//23
                int age = s.nextInt();
                Test.status(age);
        }
}
```

Ex:- ***creation of user defined un-checked exception by using parameterized constructor approach:-***

InvalidAgeException.java

```
//InvalidAgeException.java
packagecom.tcs.userexceptions;
public class InvalidAgeExcepiton extends RuntimeException
{
        publicInvalidAgeExcepiton(String str)
        {super(str);
        }
};
```

Project.java

```
packagecom.tcs.project;
importcom.tcs.userexceptions.InvalidAgeExcepiton;
importjava.util.Scanner;
class Test
{       static void status(int age)
        {       if (age>25)
                {System.out.println("eligible for mrg");
                }
                else
                {//useing user created Exception
                        throw new InvalidAgeExcepiton("not eligible for mrg");
                }
        }
        public static void main(String[] args)
        {       Scanner s = new Scanner(System.in);
                System.out.println("enter u r age");//23
                int age = s.nextInt();
                Test.status(age);
        }
}
```

Different types of exceptions:-

<u>ArrayIndexOutOfBoundsException:-</u>

```
int[] a={10,20,30};
System.out.println(a[0]);//10
System.out.println(a[3]);//ArrayIndexOutOfBoundsException
```

<u>NumberFormatException:-</u>

```
String str="123";
int a=Integer.parseInt(str);
System.out.println(a);//conversion(string - int) is good
String str1="abc";
int b=Integer.parseInt(str1);
System.out.println(b);//NumberFormatException
```

<u>NullPointerException:-</u>

```
String str="rattaiah";
System.out.println(str.length());//8
String str1=null;
System.out.println(str1.length());//NullPointerException

Test t = new Test();
t.m1();  //output printed
t=null;
t.m1(); //NullPointerException
```

<u>ArithmeticException:-</u>

```
int b=10/0;
System.out.println(b);//ArithmeticExceptiom
```

<u>IllegalArgumentException:-</u>

```
Thread  priority range is 1-10
1---→low priority
10-→high priority
Thread t=new Thread();
t.setPriority(11);//IllegalArgumentException
```

<u>IllegalThreadStateException:-</u>

```
Thread t=new Thread();
```

```
                t.start();
                t.start();//IllegalThreadStateException
```

<u>StringIndexOutOfBoundsException:-</u>

```
                String str="rattaiah";
                System.out.println(str.charAt(3));//t
                System.out.println(str.charAt(13));//StringIndexOutOfBoundsException
```

<u>NegativeArraySizeException</u>:-

```
                int[] a1=new int[100];
                System.out.println(a1.length);//100
                int[] a=new int[-9];
                System.out.println(a.length);//NegativeArraySizeException
```

<u>InputMismatchException:-</u>

```
                Scanner s=new Scanner(System.in);
                System.out.println("enter first number");
                int a=s.nextInt();
```

```
D:\>java Test
enter first number
ratan
Exception in thread "main" java.util.InputMismatchException
```

<u>Different types of Errors:-</u>

<u>StackOverflowError:-</u>

```
class Test
{       void m1()
        {       m2();
                System.out.println("this is Rattaiah");
        }
        void m2()
        {       m1();
                System.out.println("from Sravyasoft");
        }
        public static void main(String[] args)
        {       Test t=new Test();
                t.m1();
        }
}
```

<u>OutOfMemoryError:-</u>

```
class Test
{       public static void main(String[] args)
        {       int[] a=new int[100000000];     //OutOfMemoryError
        }
}
```

<u>Different types of Exceptions in java:-</u>

Checked Exception	***Description***
ClassNotFoundException	If the loaded class is not available
CloneNotSupportedException	Attempt to clone an object that does not implement the Cloneable interface.
IllegalAccessException	Access to a class is denied.
InstantiationException	Attempt to create an object of an abstract class or interface.
InterruptedException	One thread has been interrupted by another thread.
NoSuchFieldException	A requested field does not exist.
NoSuchMethodException	If the requested method is not available.
UncheckedException	***Description***
ArithmeticException	Arithmetic error, such as divide-by-zero.
ArrayIndexOutOfBoundsException	Array index is out-of-bounds.(out of range)
InputMismatchException	If we are giving input is not matched for storing input.
ClassCastException	If the conversion is Invalid.
IllegalArgumentException	Illegal argument used to invoke a method.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	Array created with a negative size.
NullPointerException	Invalid use of a null reference.
NumberFormatException	Invalid conversion of a string to a numeric format.
StringIndexOutOfBoundsException	Attempt to index outside the bounds of a string.

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