**Exception Handling**

Exception is an event raised while program execution & which disturbs normal execution flow we are calling as Exception.

The process of what we follow to handling the exception we are calling as exception handling process.

Majorly in exception handling we use 5 blocks

1) try

2) catch

3) finally

4) throw

5) throws

Compile time error:

The errors raised during compilation time are called as compiler time errors,

These are mainly syntax errors and often spelling mistakes.

Runtime errors:

The errors raised during runtime are called runtime errors or exceptions.

An exception is an abnormal condition that is raised in a code sequence at runtime.

When an exception is raised, an object representing that exception is created in the method and thrown, the exception object contains the exception information.

If an exception is not caught then the program will terminate abnormally by displaying the exception information.

Ex :below code throws ArithmeticException, but it is not caught/handled.

**package** pack1;

**public** **class** Ex1 {

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

// **TODO** Auto-generated method stub

**int** d=0;

**int** q=100/d;// RE

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

}

}

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

at pack1.Ex1.main(Ex1.java:9)

Every exception should be handled/processed so that the program should not stop at runtime.

The complete exception handling mechanism is managed by 5 keywords

**try, catch, throw, throws and finally.**

**try:** the code that should be monitored by the programmer for exceptions should be put in try-block.

**catch:** the handling code/ processing code should be placed inside catch-block.

**throw:** in general exceptions are thrown by java run-time system,but we can also throw exceptions manually, we can do this using throw keyword.

Ex: InsufficientFundsException

**throws:** if a method is throwing an exception but it is not handling it, then the method should specify this fact by including throws keyword in its declaration.

**finally:** any code that should be executed whether an exception is thrown or not is written inside finally block.

Try-block:

One try-block should be followed by at least one catch block or a finally block.

try

{

}

catch(ExType1 e)

{

}

try

{

}

catch(ExType1 e)

{

}

finally

{

}

try

{

}

finally

{

}

try

{

}

catch(ExType1 e)

{

}

catch(ExType2 e)

{

}

Once an exception is raised inside try-block then all the remaining lines are skipped.

When we write multiple catch blocks for a single try-block then

i>if there is no parent child relationship between exception types of catch-block then it is fine

ii>if there is parent child relationship between exception types then child exception class should come before parent exception class.

try

{

Stmnt1;

Stmnt2;//error raised at this line

These two statements are skipped, control will never come back to these statements even if exceptions are caught.

Stmnt3;

Stmnt4;

}

catch(ExType1 e)

{

}

Ex1:

**package** pack1;

**import** java.util.Scanner;

**public** **class** Ex1 {

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

// **TODO** Auto-generated method stub

*m1*();//method call

}

**static** **void** m1() **throws** ArithmeticException

{

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("please enter nr");

**int** nr=sc.nextInt();

System.***out***.println("please enter dr");

**int** dr=sc.nextInt();

**int** q=0;

**try**

{

q=nr/dr;

}

**catch**(ArithmeticException e)

{

System.***out***.println("**denominator should not be zero**");

}

System.***out***.println(q);

}

}

Note: if an exception is raised in a try-block then rest of the try-block is skipped even if the exception is handled.

**Multiple catch-blocks:**

One try-block can contain multiple catch-blocks.

Try

{

}

Catch(ExType1 e)

{

}

Catch(ExType2 e)

{

}

When an exception of ExType1 is throws from the above try-block then the first matching catch block is executed and remaining catch-blocks are skipped.

Ex:

**package** pack1;

**public** **class** Ex1 {

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

// **TODO** Auto-generated method stub

*m1*();//method call

}

**static** **void** m1()

{

**int** d=10;

**int** n=100;

**try**

{

**int** q=n/d;

String s=**null**;

s.length();

}

**catch**(ArithmeticException e)

{

System.***out***.println("denominator should not be zero");

}

**catch**(NullPointerException e)

{

System.***out***.println("string is not initialized ");

}

}

}

**Catch-block:**

Once a catch-block is matched then remaining catch-blocks are skipped.

catch(ExType e)

{

System.out.println(e);

OR

We can write our own message

OR

e.printStack();

}

Examples for multi catch blocks

**1.compiles fine**

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

{

// **TODO** Auto-generated method stub

**try**

{

}

**catch**(ArithmeticException e)

{

}

**catch**(NullPointerException e)

{

}

}

}

2.compiles fine

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

{

// **TODO** Auto-generated method stub

**try**

{

}

**catch**(NullPointerException e)

{

}

**catch**(ArithmeticException e)

{

}

}

}

3.gives CE

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

{

// **TODO** Auto-generated method stub

**try**

{

}

**catch**(Exception e)

{

}

**catch**(NullPointerException e)

{

}

}

}

4.compiles fine

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

{

// **TODO** Auto-generated method stub

**try**

{

}

**catch**(NullPointerException e)

{

}

**catch**(Exception e)

{

}

}

}

Nest try-catch statements:

We can write try-catch-finally statements inside try or inside catch or inside finally.

try

{

Stmnt1;

Stmnt2;//AE

Try

{

Stmnt3;//NPE

Stmnt4;//CCE

}

Catch(NPE e)

{

}

}

catch(Exception e)

{

}

finally

{

}

Ex1:

**package** pack1;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**public** **class** IOEx {

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

// **TODO** Auto-generated method stub

BufferedReader br=**null**;

PrintWriter pw=**null**;

String line=**null**;

**boolean** b=**false**;

**try**

{

br=**new** BufferedReader(**new** FileReader("C:\\Users\\ravilella\\Documents\\input1.txt"));

pw=**new** PrintWriter("C:\\Users\\ravilella\\Documents\\output.txt");

line=br.readLine();

**if**(line==**null**)

{

**throw** **new** Exception("no data");

}

**while**(line!=**null**)

{

pw.println(line);

line=br.readLine();

}

}

**catch**(Exception e)

{

System.***out***.println("please check data avilability in input file");

b=**true**;

}

**finally**

{

br.close();

pw.close();

}

pw.flush();

**if**(b==**false**)

System.***out***.println("pls check output text file");

}

}

Ex2:

**package** pack1;

**import** java.util.Scanner;

**public** **class** NestedTry {

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

// **TODO** Auto-generated method stub

Scanner sc=**new** Scanner(System.***in***);

System.***out***.print("enter nr :");

**int** nr=sc.nextInt();

System.***out***.println();

System.***out***.print("enter dr :");

**int** dr=sc.nextInt();

**int** q=0;

**try**

{

q=nr/dr;

System.***out***.println(q);

**try**

{

System.***out***.print("enter some string :");

String s=sc.next();

**int** l=s.length();

**if**(l<3)

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

}

**catch**(NullPointerException e)

{

System.***out***.println("length must be greater than 3");

}

}

**catch**(ArithmeticException e)

{

System.***out***.println("dr cannot be zero");

}

}

}

Every exception is a child class of Throwable class.

Throwable is the root class or parent class for every exception class.

Throwable

Exception

Error

RuntimeEx IOEx SQLEx ServletEx…

FileNotFoundEx

ArithEx EOFEx

NullPEx

ClassCastEx

IndexOutOfBoundsEx

ArrayIndexOutOfBoundsEx

StringIndexOutOfBoundsEx

Checked exception and unchecked exceptions:

the exceptions which are checked by the compiler for try-catch or for throws keyword at compile time are called as checked exceptions

ex:this program gives compilation error.

**package** pack1;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**public** **class** CheckedEx {

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

// **TODO** Auto-generated method stub

BufferedReader br=**new** BufferedReader(**new** FileReader("input.txt"));

}

}

We can resolve above problem by writing try-catch or by using throws keyword as shown below

1)

**package** pack1;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** CheckedEx {

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

// **TODO** Auto-generated method stub

**try**

{

BufferedReader br=**new** BufferedReader(**new** FileReader("input.txt"));

}

**catch**(IOException e)

{

}

}

}

2)

**package** pack1;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** CheckedEx {

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

// **TODO** Auto-generated method stub

BufferedReader br=**new** BufferedReader(**new** FileReader("input.txt"));

}

}

**Unchecked exceptions:**

The exceptions which are not checked by the compiler for try-catch or for throws keyword are called as unchecked exceptions.

Ex:this program will not give compile time error

**package** pack1;

**public** **class** UnCheckedEx {

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

// **TODO** Auto-generated method stub

**int** nr=10;

**int** dr=0;

**int** q=nr/dr;

}

}

Note:

Whether the exception is checked or unchecked it is raised at runtime only.

Throws keyword:

inside a method, if there is a chance of raising any checked exception then that method has to handle it using try-catch or it can pass it on to the caller using throws keyword.

If the method uses throws keyword then it is the responsibility of caller method to handle the exception. Now the caller method may choose to handle the exception using try-catch or using throws keyword.

Ex:below program gives CE.

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

// **TODO** Auto-generated method stub

*m1*();

}

**static** **void** m1()

{

**throw** **new** FileNotFoundException("demo");//CE

}

}

we can resolve the above CE in 2ways:

1.using try-catch

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

// **TODO** Auto-generated method stub

*m1*();

}

**static** **void** m1()

{

**try**

{

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

}

**catch**(Exception e)

{

}

}

}

2a)using throws keyword.

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

{

// **TODO** Auto-generated method stub

**try**

{

*m1*();

}

**catch**(Exception e)

{

}

}

**static** **void** m1()**throws** Exception

{

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

}

}

2b)using throws keyword

**package** pack1;

**import** java.io.FileNotFoundException;

**public** **class** ThrowsEx{

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

{

// **TODO** Auto-generated method stub

*m1*();

}

**static** **void** m1()**throws** Exception

{

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

}

}