**Exception Handling:**

An unexpected or unwanted event that disturbs the normal flow of execution of program is called exception.

When we handle that exception by using a code then we called that process as Exception handling. Which terminate the program normally instead of abnormally.

There are two ways to handle an exception:

1. try-catch or try-catch-finally.

2. throws keyword.

Using try-catch block:

Try is a block which contains risky code(a code which we feel can cause an exception) and whenever an exception occurs at a particular line the control leave the try block at that point and search for the corresponding catch block.

Catch : It is also a block which has type of exception associated with it. Whenever an exception occurs and its type matches with any type of catch block then that catch block gets execute.

We can have multiple catch block with one try but we cannot have multiple try with one catch block.

If we get an exception and we don’t have corresponding catch block to handle it then our program will get terminate abnormally.

Example of Arithmetic Exception handling:

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

**int** k;

**try** {

**int** i = 10;

**int** j = 0;

k = i / j;

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

}

**catch**(ArithmeticException e)

{

System.***out***.println("Arithmetic catch is excuting");

}

System.***out***.println("After the exception handling");

}

}

Output:

Arithmetic catch is excuting

After the exception handling

Example 2 in which we cannot protect program from terminating abnormally:

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

**int** k;

**try** {

**int** i = 10;

**int** j = 0;

k = i / j;

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

}

**catch**(ArrayIndexOutOfBoundsException e)

{

System.***out***.println("Arithmetic catch is excuting");

}

**catch**(NullPointerException q)

{

System.***out***.println("Null pointer exception has arrived");

}

System.***out***.println("After the exception handling");

}

}

Output:

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

at exceptionhandlingdiscussion.TryCatch.main(TryCatch.java:13)

Note: Here no catch block is matching with the type of exception which comes into the try block hence the code gets terminate abnormally.

Example3: NullpointerException :

**public** **class** NullPointerExcepEx {

**static** NullPointerExcepEx *npe*;

**public** **void** m1()

{

System.***out***.println("m1 method is executing");

}

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

**try** {

*npe*.m1();

}

**catch**(NullPointerException e)

{

System.***out***.println("null pointer exception arrived");

}

System.***out***.println("After catch block");

}

}

Output:

null pointer exception arrived

After catch block

FileNotFoundException Example:

**public** **class** FileNotFoundExe {

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

**try**

{

System.***out***.println("before tracking the file");

FileInputStream fis = **new** FileInputStream("E:\\desktop\\Katraj\\15 Jan\\Constrdeewructor.docx");

}

**catch**(FileNotFoundException e)

{

System.***out***.println("file not found exception arrived and catch is executing");

}

System.***out***.println("after tracking the file");

}

Output:

before tracking the file

file not found exception arrived and catch is executing

after tracking the file

Note: In the above program if we are not able to locate the file in the system then we will get FileNotFoundException which is here handled by catch block.

**Categories of exception**:

There are 2 categories of exception:

a. checked

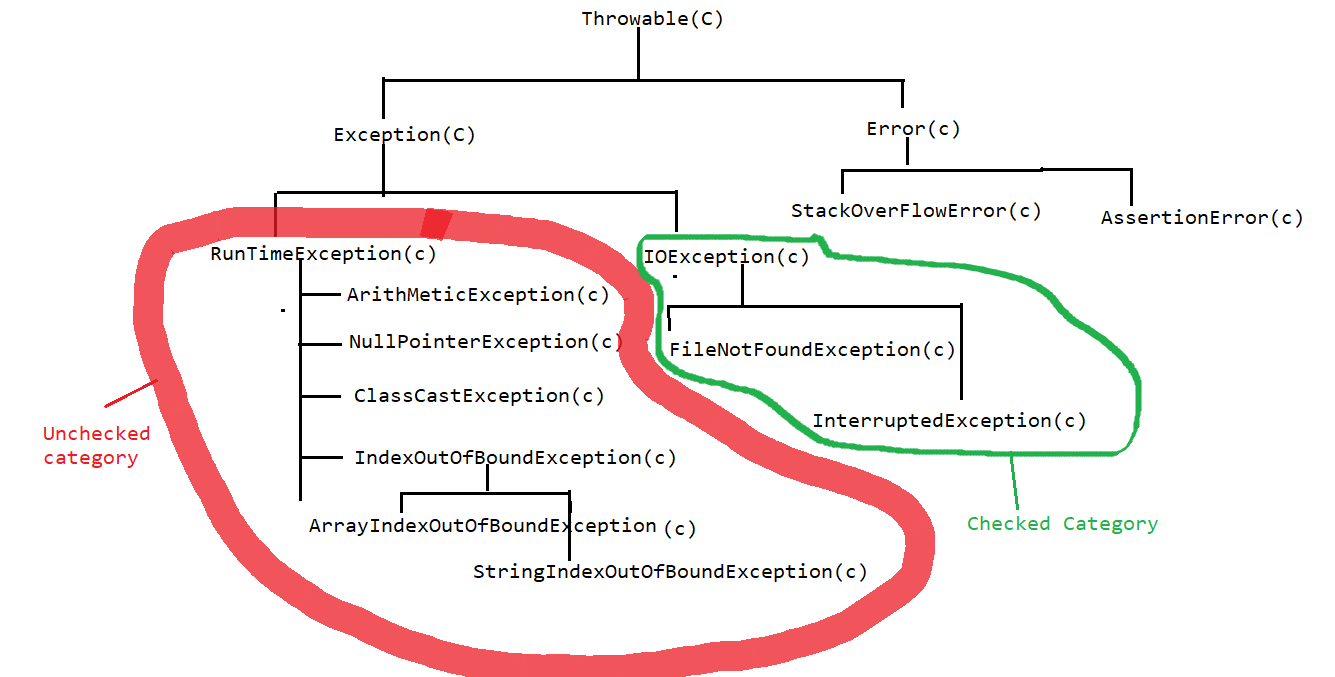
b. unchecked.

a. Checked category: A kind of exception whose code will suggest to handle the exception at the time of compilation of program is called checked category.

Example :FileNotFoundException, InterruptedException

a. UnChecked category: A kind of exception whose code will not suggest to handle the exception at the time of compilation of program and if any exception arrived we will see at the run time if we have not handled it properly is called Unchecked category.

Example :ArithmeticException, NullPointerException



Note:

We can have multiple catch block in that we must have Exception block to be present as a last catch block. If we have Exception block present in between the other catch block we will get error.

Example:

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

**int** k;

**try** {

**int** i = 10;

**int** j = 0;

k = i / j;

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

}

**catch**(ArrayIndexOutOfBoundsException e)

{

System.***out***.println("Arithmetic catch is excuting");

}

**catch**(NullPointerException q)

{

System.***out***.println("Null pointer exception has arrived");

}

**catch** (ClassCastException c) {

}

**catch**(Exception w)

{

System.***out***.println("Exception block ");

}

**catch**(Throwable e)

{

System.***out***.println("Throwable block ");

}

System.***out***.println("After the exception handling");

}

}

Checked Exception example:

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

**try**

{

System.***out***.println("before tracking the file");

FileInputStream fis = **new** FileInputStream("E:\\desktop\\Kfsfsdfatraj\\15 Jan\\Constructor.docx");

}

**catch**(FileNotFoundException e)

{

System.***out***.println("file not found exception arrived and catch is executing");

}

System.***out***.println("after tracking the file");

}

Example 2:

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

System.***out***.println("First line");

**try** {

Thread.*sleep*(2000);

}

**catch**(InterruptedException e)

{

System.***out***.println("Catch block");

}

System.***out***.println("last line");

}

Note: In both of the example if we don’t handle the exception then we will get compile time error which suggest to handle the exception.

**try –catch – finally:**

We can use the combination of try – catch – and finally in which the finally block should come after the catch in the combination of try-catch-finally or finally can come after try block in the combination of try-finally.

Finally : It is a block which get executes every time irrespective of an exception arrives into a program. It contains clean up activity related code.

We can use try-finally also but if the exception arrives into the try block then program will get terminate abnormally but finally will always gets execute.

Example: try – catch – finally:

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

**try** {

**int** i=10;

**int** j = 2;

**int** k = i /j;

}

**catch**(ArithmeticException e)

{

String message = e.getMessage();

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

System.***out***.println("Arithmetic exception block");

}

**finally**

{

System.***out***.println("finally block");

}

}

Output:

finally block

Example try –finally:

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

**try** {

System.***out***.println("inside the try block");

**int** i=10;

**int** j = 0;

**int** k = i /j;

}

**finally**

{

System.***out***.println("finally block");

}

}

}

Output:

inside the try block

finally block

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

at exceptionhandlingdiscussion.TryFinally.main(TryFinally.java:12)

**final vs finally vs finalize()**

**1. final:** It is an access modifier applicable to class, method and variable.

**2.** finally: It is a block which can be used after try-catch or try block. It always gets execute irrespective of exception arrived in the program.

3. **finalize**(): It is a method which is used to remove the unwanted or unnecessary object in the program. Garbage collector calls finalize method to clear the unwanted object.

**2. Throws keyword:** It is used to handle the exception but it cannot protect the program by terminating it abnormally.

Using throws keyword we can only resolve the compile time error but if an exception comes into the program then we will not be able to handle it. To handle that exception we must have try-catch block.

Example:

**public** **static** **void** main(String[] args) **throws** FileNotFoundException, InterruptedException, ArithmeticException

{

System.***out***.println("first line");

FileInputStream fis = **new** FileInputStream("E:\\desktop\\Kfsfsdfatraj\\15 Jan\\Constructor.docx");

System.***out***.println("LAst line");

}

Output:

first line

Exception in thread "main" java.io.FileNotFoundException: E:\desktop\Kfsfsdfatraj\15 Jan\Constructor.docx (The system cannot find the path specified)

at java.base/java.io.FileInputStream.open0(Native Method)