The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The try and catch keywords come in pairs:

try {

// *Block of code to try*

}

catch(Exception e) {

// *Block of code to handle errors*

}

## Finally

The finally statement lets you execute code, after try...catch, regardless of the result:

public class Main {

public static void main(String[] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]);

} catch (Exception e) {

System.out.println("Something went wrong."); // custom error

System.out.println(e); // Prins exact Exception msg.

} finally {

System.out.println("The 'try catch' is finished.");

}

}

}

The output will be:

Something went wrong.  
The 'try catch' is finished.

## throw keyword

The throw statement allows you to create a custom error.

The throw statement is used together with an **exception type**. There are many exception types available in Java: ArithmeticException, FileNotFoundException, ArrayIndexOutOfBoundsException,  SecurityException, etc:

### **Example**

Throw an exception if **age** is below 18 (print "Access denied"). If age is 18 or older, print "Access granted":

public class Main {

static void checkAge(int age) {

if (age < 18) {

throw new ArithmeticException("Access denied - You must be at least 18 years old.");

}

else {

System.out.println("Access granted - You are old enough!");

}

}

public static void main(String[] args) {

checkAge(15); // Set age to 15 (which is below 18...) }

}

The output will be:

Exception in thread "main" java.lang.ArithmeticException: Access denied - You must be at least 18 years old.  
        at Main.checkAge(Main.java:4)  
        at Main.main(Main.java:12)

**to maintain the normal flow of the application**.

The throw and throws is the concept of exception handling where the throw keyword throw the exception explicitly from a method or a block of code whereas the throws keyword is used in signature of the method.

**public** **static** **int** divideNum(**int** m, **int** n) **throws** ArithmeticException{ …}

// in method signature

|  |  |
| --- | --- |
| finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature |

1. import java.io.\*;
2. **class** M{
3. **void** method()**throws** IOException{
4. **throw** **new** IOException("device error");
5. }
6. }
7. **public** **class** Testthrows2{
8. **public** **static** **void** main(String args[]){
9. **try**{
10. M m=**new** M();
11. m.method();
12. }**catch**(Exception e){System.out.println("exception handled");}
14. System.out.println("normal flow...");
15. }
16. }

****Output:****

*exception handled*

*normal flow...*

# Java Exception Propagation

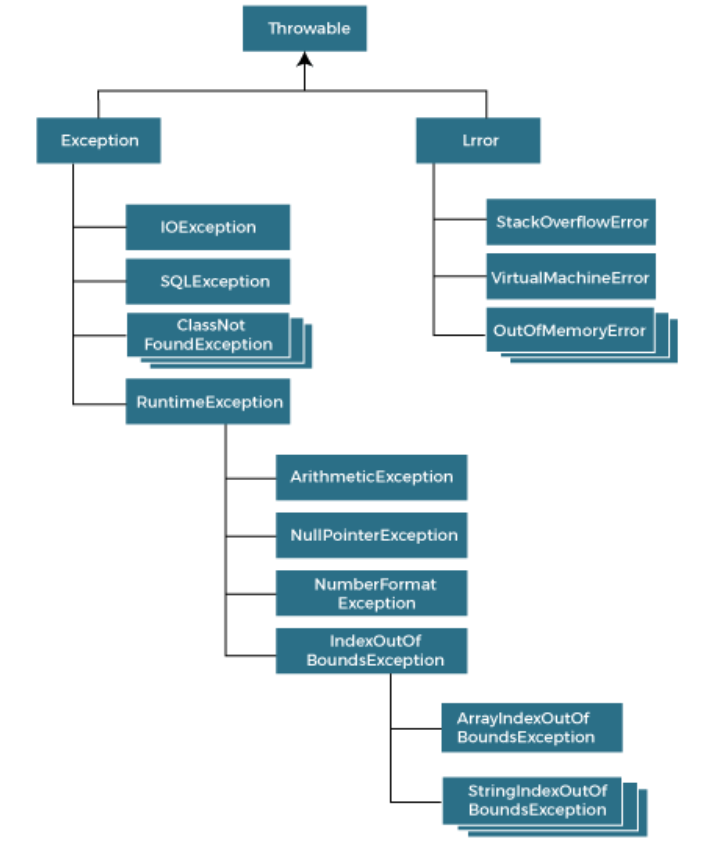
An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method. If not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This is called exception propagation.

Means if method calling another method has an Exception but the Exception is handled in calling method so this scenario is called Exception Propagation.

|  |  |  |
| --- | --- | --- |
| **final** | **finally** | **finalize** |
| final is the keyword and access modifier which is used to apply restrictions on a class, method or variable. | finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not. | finalize is the method in Java which is used to perform clean up processing just before object is garbage collected. |

## Hierarchy of Java Exception classes

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



## Difference between Checked and Unchecked Exceptions

### **1) Checked Exception**

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

### **2) Unchecked Exception**

The classes that inherit the RuntimeException are known as unchecked exceptions. For example,ArithmeticException,NullPointerException,ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

### **3) Error**

Error is irrecoverable.

String s=**null**;

System.out.println(s.length());//NullPointerException

* At a time only one exception occurs and at a time only one catch block is executed.
* All catch blocks must be ordered from most specific to most general, i.e. catch for ArithmeticException must come before catch for Exception.

1. Try vs Catch vs Finally vs Throw vs Throws? Code it
2. Exception propogation?
3. What is hierarchy of java exception class?