


EXCEPTION HANDLING IN JAVA



- 
- ❑ Exception Handling in Java is one of the effective means to handle the runtime errors so that the regular flow of the application can be preserved.
 - ❑ Java Exception Handling is a mechanism to handle runtime errors such as `ClassNotFoundException`, `IOException`, `SQLException`, `RemoteException`, etc

Types Of Errors in Java

There are two types of errors in Java.

- ❑ **Compile-Time Error:** This is a common error that comes by the compiler. When curly brace, semi-colon or comma is not given in any syntax in the program, then this error occurs at compile-time.
- ❑ **Run-Time Error:** Here the program is successfully run. But some such internal errors come, which are given by the interpreter. This also closes the program.

Advantage of Exception Handling in Java

- ❑ Exception handling ensures that the flow of the program is maintained when an exception occurs.
- ❑ By this we can identify the types of errors.
- ❑ By this we can write the error-handling code separately from the normal code.

What is an Exception?

- ❑ An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.
- ❑ Exceptions can be caught and handled by the program. When an exception occurs within a method, it creates an object. This object is called the exception object.
- ❑ It contains information about the exception such as the name and description of the exception and the state of the program when the exception occurred.

Reasons of Exception occurrence

An exception can occur for many reasons. Some of them are:

- ☐ Invalid user input
- ☐ Device failure
- ☐ Loss of network connection
- ☐ Physical limitations (out of disk memory)
- ☐ Code errors
- ☐ Opening an unavailable file

What is an error?

- ❑ Errors represent irrecoverable conditions such as Java virtual machine (JVM) running out of memory, memory leaks, stack overflow errors, library incompatibility, infinite recursion, etc.
- ❑ Errors are usually beyond the control of the programmer and we should not try to handle errors.

Error vs Exception

- ❑ **Error:** An Error indicates a serious problem that a reasonable application should not try to catch.
- ❑ **Exception:** Exception indicates conditions that a reasonable application might try to catch.

Exception Models

- In java, there are two exception models. Java programming language has two models of exception handling. The exception models that java supports are as follows.
- **Termination Model**
- **Resumptive Model**

Termination Model

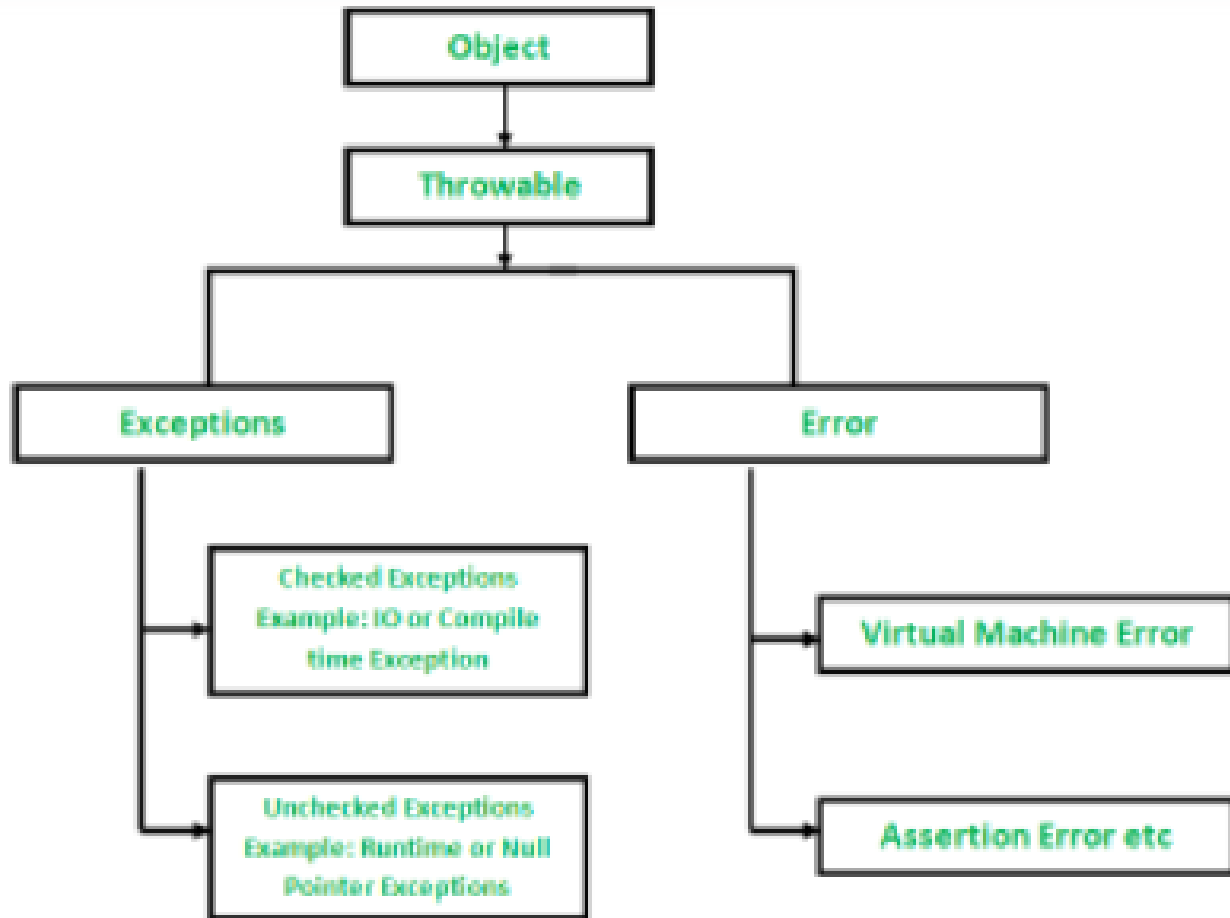
- ❑ In the termination model, when a method encounters an exception, further processing in that method is terminated and control is transferred to the nearest catch block that can handle the type of exception encountered.
- ❑ In other words we can say that in termination model the error is so critical there is no way to get back to where the exception occurred.

Resumptive Model

- ❑ The alternative of termination model is resumptive model. In resumptive model, the exception handler is expected to do something to stable the situation, and then the faulting method is retried. In resumptive model we hope to continue the execution after the exception is handled.
- ❑ In resumptive model we may use a method call that want resumption like behavior. We may also place the try block in a while loop that keeps re-entering the try block util the result is satisfactory.

Exception Hierarchy

- All exception and errors types are subclasses of class **Throwable**, which is the base class of the hierarchy. One branch is headed by **Exception**.
- This class is used for exceptional conditions that user programs should catch. `NullPointerException` is an example of such an exception.
- Another branch, **Error** is used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE). `StackOverflowError` is an example of such an error.

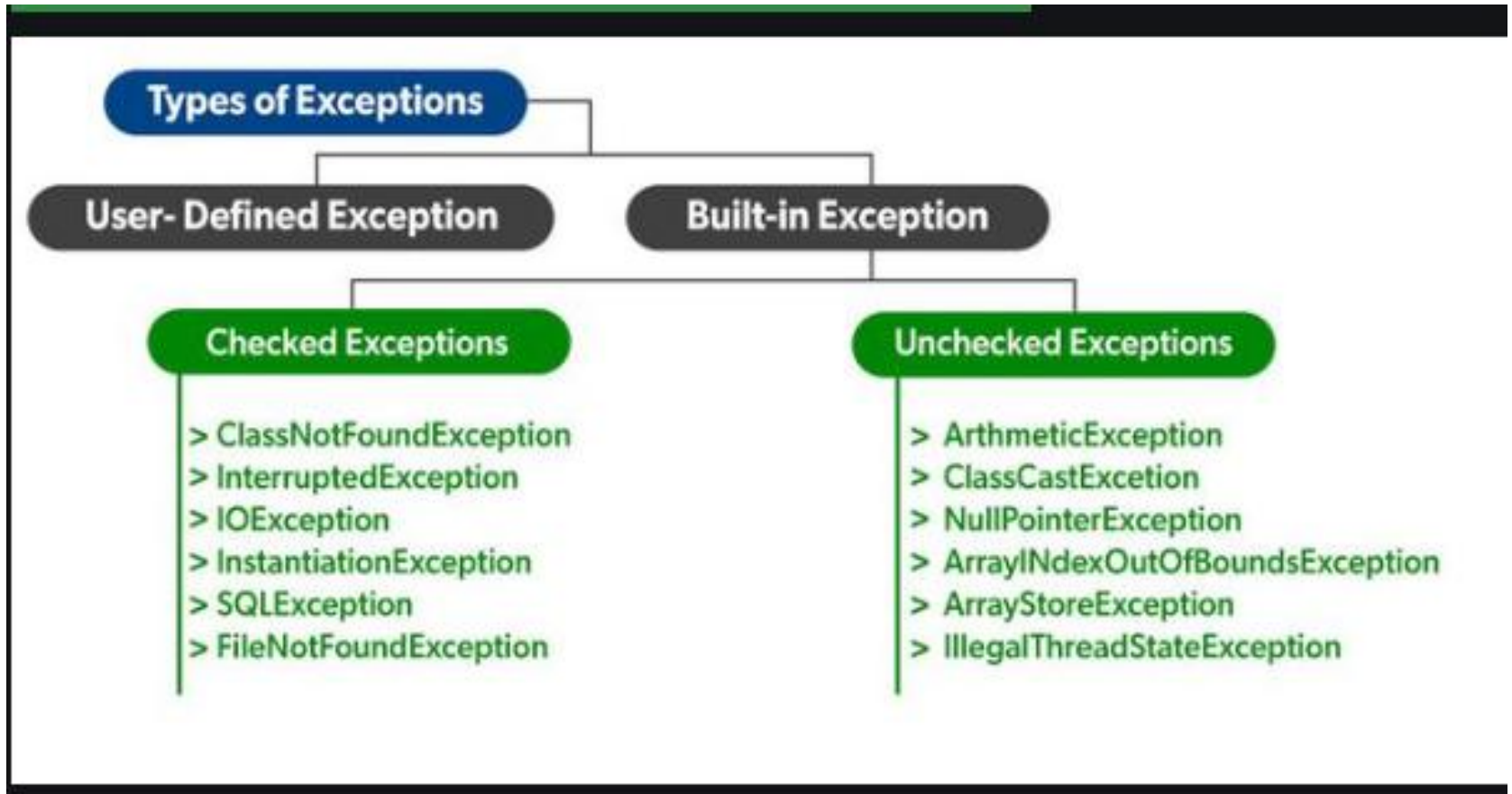


Types of Exceptions

- ❑ Java defines several types of exceptions that relate to its various class libraries.
- ❑ Java also allows users to define their own exceptions.

Categories of Exceptions

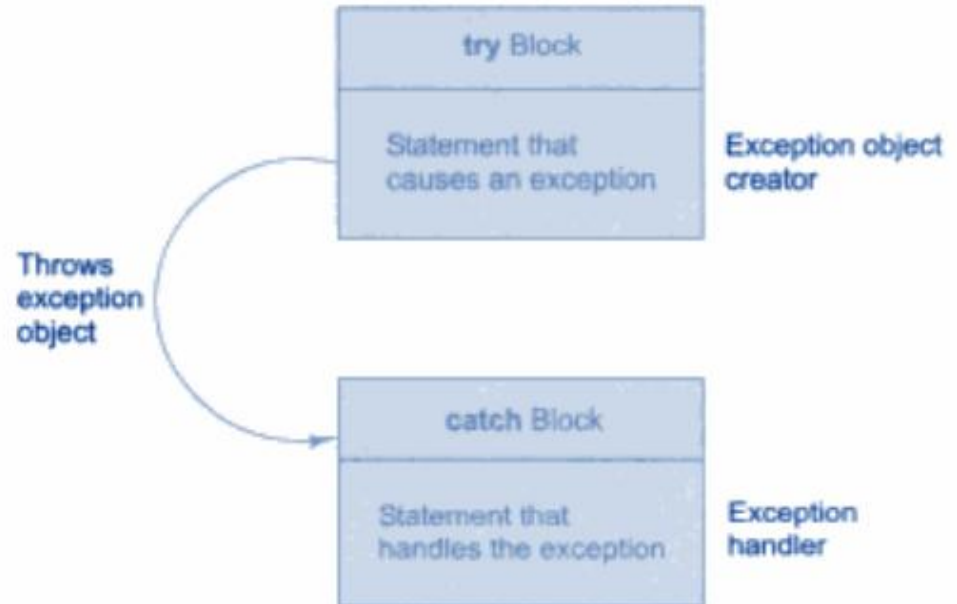
- ❑ **Exceptions can be Categorized into 2 Ways:**
- ❑ Built-in Exceptions
 - ▣ Checked Exception
 - ▣ Unchecked Exception
- ❑ User-Defined Exceptions



Exception class	Meaning
ArithmeticException	If we divide an integer number with zero
ArrayIndexOutOfBoundsException	If we are accessing array element by passing index value that is out of range
ClassNotFoundException	These exception is raised when we try to access a class whose definition is not found
FileNotFoundException	Raised when a file is not accessible or does not open
IOException	Thrown when an input/output operation failed or interrupted
InterruptedException	Thrown when an thread is waiting, sleeping or doing some processing and it is interrupted
NoSuchMethodException	Thrown when accessing a method which is not found
NullPointerException	Raised when we are accessing object members by using null reference variable
NumberFormatException	Raised when a method could not convert a String into a numeric format
RuntimeException	This represents any exception which occurs during runtime
NoSuchFieldException	Thrown when a class does not contain the field(variable) specified
NegativeArraySizeException	If we create an array by passing the array size as a negative number
StringIndexOutOfBoundsException	Thrown by string class methods to indicate that an index is out of range or greater than the size of the string

Syntax

```
...  
...  
try{  
    //statements to generate exception  
}  
catch(Exception-type object){  
    //statements to process exception  
}  
...  
...
```



Example

```
import java.io.*;

public class ExcepTest {
    public static void main(String args[]) {
        try { int a[] = new int[2];
            System.out.println("Access element three :" + a[3]); }
        catch ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception thrown :" + e); }
        System.out.println("Out of the block"); } }
```

OUTPUT:

Exception thrown :

java.lang.ArrayIndexOut
OfBoundsException: 3

Out of the block

Multiple Catch Blocks

```
try { // Protected code }  
catch (ExceptionType1 e1) { // Catch block }  
catch (ExceptionType2 e2) { // Catch block }  
catch (ExceptionType3 e3) { // Catch block }
```

The Finally Block

- ❑ The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.
- ❑ Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

Syntax of Finally block

```
try { // Protected code }  
catch (ExceptionType1 e1) { // Catch block }  
catch (ExceptionType2 e2) { // Catch block }  
catch (ExceptionType3 e3) { // Catch block }  
finally { // The finally block always executes }
```

Methods in Exception Handling

<code>public String getMessage()</code>	Get a detailed message about the exception that occurred.
<code>public Throwable getCause()</code>	Get the cause of the exception represented by a throwable object
<code>public void printStackTrace()</code>	Prints the result of <code>toString()</code> and the contents of stack trace to the error output stream, <code>System.err</code> .
<code>public StackTraceElement [] getStackTrace()</code>	Get each element in the stack trace in the form of an array.
<code>public Throwable fillInStackTrace()</code>	Fill the stack trace with the current stack trace.

Points to remember

- ❑ A catch clause cannot exist without a try statement.
- ❑ It is not compulsory to have finally clauses whenever a try/catch block is present.
- ❑ The try block cannot be present without either catch clause or finally clause.
- ❑ Any code cannot be present in between the try, catch, finally blocks.

User-defined Exceptions

- ❑ We can create our own exception in Java. To create your own exception class, the following things should be kept in mind.
- ❑ All exceptions that are there must be children of Throwable.
- ❑ If you want to write checked exception then you have to extend Exception class.
- ❑ If you want to write runtime exception then you will need to extend RuntimeException class.

Response	Percentage
Yes	75%
No	25%

```
class MyException extends Exception
{

}

```

How to Handle Exceptions?

Exception in Java is handled by 5 keywords.

- ❑ Try
- ❑ Catch
- ❑ Finally
- ❑ Throw
- ❑ Throws

Try

- ❑ The “try” keyword is used to specify a block where we keep the exception code.
- ❑ There must be a catch block or finally block at the end of this block. This means that we cannot use the try block alone.
- ❑ If it contains both catch and finally then their sequence should be try-catch-finally. Because if their order is wrong then compile-time error will come.
- ❑ The code inside the try block should always be inside curly braces. Otherwise a compile-time error will occur.

Catch

- ❑ The “catch” block is used to handle the exception.
- ❑ It should always be used after the try block. This means that we cannot use catch alone.
- ❑ This block takes an argument. This argument must be either of type Throwable or of its sub-class.
- ❑ Code inside the catch block should always be inside curly braces. Otherwise a compile-time error will occur.

Finally

- ❑ The “finally” block is used to execute the important code of the program. Such as – closing the database connection, closing the file resources etc.
- ❑ It is always used with try-catch block.
- ❑ There can be 2 combinations with finally. One try-finally and the other try-catch-finally.
- ❑ The finally block is always executed, whether the exception is handled or not.
- ❑ It executes after try and catch block.

Throw

- ❑ It is used to throw an exception.
- ❑ It is used with the exception of both checked and unchecked.
- ❑ Generally, the throw keyword is used to throw a user-defined exception.

Syntax:-

- ❑ `throw new Throwable_subclass;`

Throws

- ❑ The throws keyword is used to declare an exception.
- ❑ It does not throw exceptions.
- ❑ It is generally used to handle checked exceptions.
- ❑ If you do not want to handle the program with try and catch blocks, then you can handle it with throws.
- ❑ It is always used with method signature.

❑ Syntax:-

```
return_type  
    method_name() throws  
        exception_class_name  
{  
    //method code  
}
```


The Throw Keyword

“Throw” keyword is used to throw the exception, whereas the “throws” keyword is used to declare the exception.

```
public class Demothrow {  
    String content ;  
    public void driving(String c) {  
        this.content=c;  
        if (content.isEmpty()) {  
            throw new NullPointerException("content is empty");  
        }  
        System.out.println("content==" + content);  
    }  
    public static void main(String[] args) {  
        Demothrow dt=new Demothrow();  
        dt.driving("");  
    }  
}
```

The Throws Keyword

Throws are used to give information that this method throws this particular exception.

When you call that particular method, you need to handle that exception.

```
public class DemoThrows {  
    int i=2,j=0;  
    public void checkmethod () throws ArithmeticException{  
        System.out.println(i/j);}  
    public static void main(String[] args) {  
        DemoThrows d =new DemoThrows();  
        try {  
            d.checkmethod();}  
        catch (ArithmeticException ae) {  
            ae.printStackTrace();}  
        }  
    }
```

Key points to be noted:

- ❑ Exceptions are abnormal events that are occurred during the program execution and it will affect the execution flow.
- ❑ An error is different from exceptions. **Errors**
Example: Memory out of error.
- ❑ Checked exceptions are checked during compilation and it is mandatory to deal with this checked exception.
- ❑ An unchecked exception occurs during the runtime.

References

- E.Balaguruswamy, Programming with JAVA, A primer, 3e, TATA McGraw-Hill Company.
- <https://www.tutorialspoint.com/>
- <https://www.softwaretestinghelp.com/java/java-exceptions/>
- <https://www.programiz.com/java-programming/exceptions>