Java Exception Handling

Exception is a condition that occurs during program execution and lead to program termination abnormally. Exception Handling is a mechanism to handle exception at runtime. There can be several reasons that can lead to exceptions, including programmer error, hardware failures, files that need to be opened cannot be found, resource exhaustion etc.

Suppose we run a program to read data from a file and if the file is not available then the program will stop execution and terminate the program by reporting the exception message.

The problem with the exception is, it terminates the program and skip rest of the execution that means if a program has 100 lines of code and at line 10 an exception occurs then program will terminate immediately by skipping execution of rest 90 lines of code.

To handle this problem, we use exception handling that avoid program termination and continue the execution by skipping exception code.

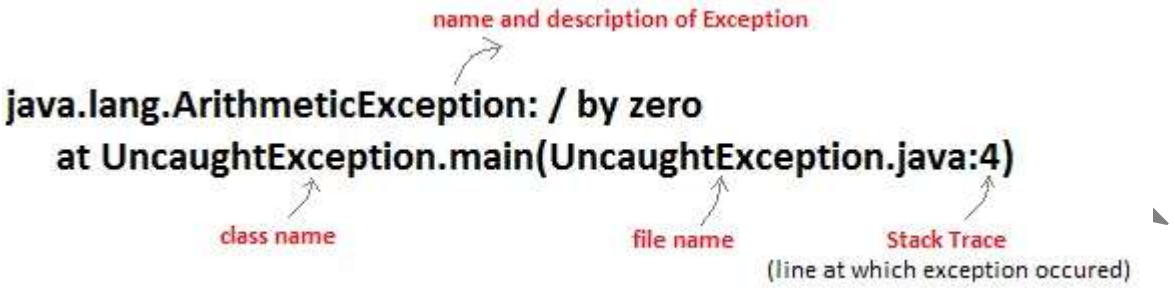
Java exception handling provides a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user.

Uncaught Exceptions

Let’s understand exception with an example. When we don't handle the exceptions, it leads to unexpected program termination. In this program, an ArithmeticException will be throw due to divide by zero.

```
class UncaughtException {
    public static void main (String args[])
    {
        int a = 0;
        int b = 7/a;
    }
}
```

This will lead to an exception at runtime; hence the Java run-time system will construct an exception and then throw it. As we don't have any mechanism for handling exception in the above program, hence the default handler (JVM) will handle the exception and will print the details of the exception on the terminal.



Java Exception

A Java Exception is an object that describes the exception that occurs in a program. When an exceptional event occurs in java, an exception is said to be thrown. The code that is responsible for doing something about the exception is called an exception handler

How to Handle Exception

Java provides controls to handle exception in the program. These controls are listed below.

try: It is used to enclose the suspected code.

catch: It acts as exception handler.

throw: It throws the exception explicitly.

throws: It informs for the possible exception.

finally: It is used to execute necessary code.

Types of Exceptions

In Java, exceptions can be broadly categorized into checked exception, unchecked exception and error based on the nature of exception.

1) Checked Exception

The exception that can be predicted by the JVM at the compile time,

For example: File that needs to be opened is not found, SQLException etc. These types of exceptions must be checked at compile time.

2) Unchecked Exception

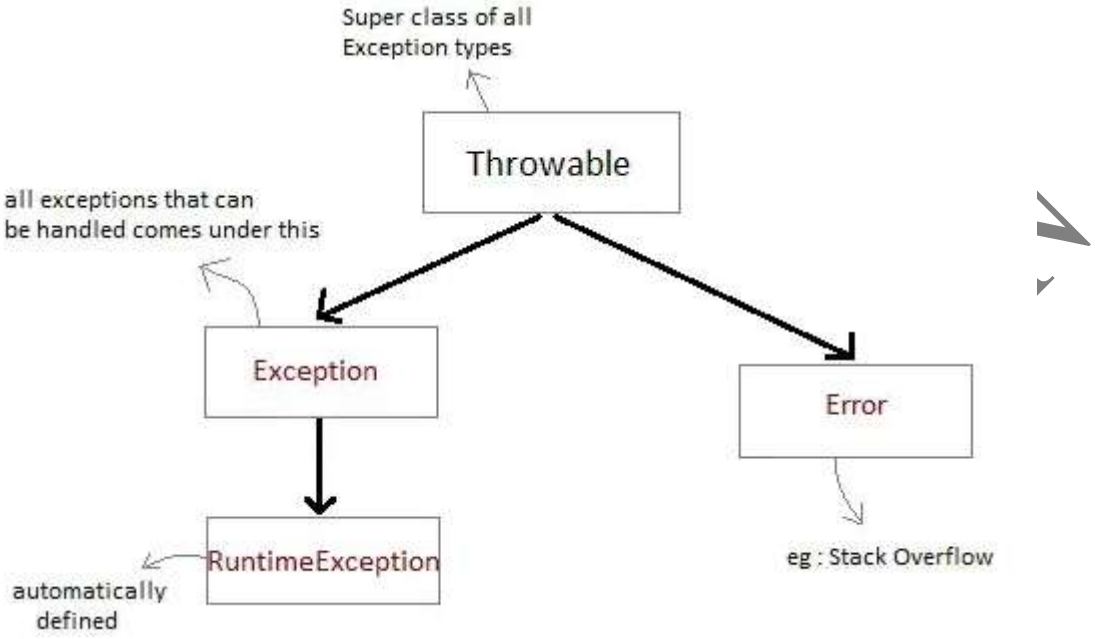
Unchecked exceptions are the class that extends RuntimeException class. Unchecked exception is ignored at compile time and checked at runtime. For example: ArithmeticException, NullPointerException, Array Index out of Bound exception. Unchecked exceptions are checked at runtime.

3) Error

Errors are typically ignored in code because you can rarely do anything about an error. For example, if stack overflow occurs, an error will arise. This type of error cannot be handled in the code.

Java Exception class Hierarchy

All exception types are subclasses of class Throwable, which is at the top of exception class hierarchy.



1) Exception class is for exceptional conditions that program should catch. This class is extended to create user specific exception classes.

2) RuntimeException is a subclass of Exception. Exceptions under this class are automatically defined for programs.

3) Exceptions of type Error are used by the Java run-time system to indicate errors having to do with the run-time environment, itself. Stack overflow is an example of such an error.