Errors V/s Exceptions In Java:

In java, both Errors and Exceptions are the subclasses of java.lang.Throwable class. Error refers to an illegal operation performed by the user which results in the abnormal working of the program. Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus errors should be removed before compiling and executing. It is of three types:

Compile-time

Run-time

Logical

Whereas exceptions in java refer to 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.



Now let us discuss various types of errors in order to get a better understanding over arrays. As discussed in the header an error indicates serious problems that a reasonable application should not try to catch. Errors are conditions that cannot get recovered by any handling techniques. It surely causes termination of the program abnormally. Errors belong to unchecked type and mostly occur at runtime. Some of the examples of errors are Out of memory errors or System crash errors.

Example 1 Run-time Error

// Java Program to Illustrate Error

// Stack overflow error via infinite recursion

// Class 1

class StackOverflow {

// method of this class

public static void test(int i)

{

// No correct as base condition leads to

// non-stop recursion.

if (i == 0)

return;

else {

test(i++);

}

}

}

// Class 2

// Main class

public class GFG {

// Main driver method

public static void main(String[] args)

{

// Testing for error by passing

// custom integer as an argument

StackOverflow.test(5);

}

}

// Java Program to Illustrate Run-time Errors

// Main class

class GFG {

// Main driver method

public static void main(String args[])

{

// Declaring and initializing numbers

int a = 2, b = 8, c = 6;

if (a > b && a > c)

System.out.println(a

+ " is the largest Number");

else if (b > a && b > c)

System.out.println(b

+ " is the smallest Number");

// The correct message should have been

// System.out.println

// (b+" is the largest Number"); to make logic

else

System.out.println(c

+ " is the largest Number");

}

}

Output

8 is the smallest Number

Now let us dwell onto Exceptions which indicates conditions that a reasonable application might want to catch. Exceptions are the conditions that occur at runtime and may cause the termination of the program. But they are recoverable using try, catch and throw keywords. Exceptions are divided into two categories:

Checked exceptions

Unchecked exceptions

Checked exceptions like IOException known to the compiler at compile time while unchecked exceptions like ArrayIndexOutOfBoundException known to the compiler at runtime. It is mostly caused by the program written by the programmer.

Example Exception

// Java program illustrating exception thrown

// by Arithmetic Exception class

// Main class

class GFG {

// main driver method

public static void main(String[] args)

{

int a = 5, b = 0;

// Try-catch block to check and handle exceptions

try {

// Attempting to divide by zero

int c = a / b;

}

catch (ArithmeticException e) {

// Displaying line number where exception occurred

// using printStackTrace() method

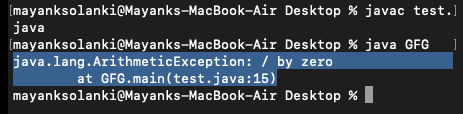
e.printStackTrace();

}

}

}

Output:



Finally now wrapping-off the article by plotting the differences out between them in a tabular format as provided below as follows:

Errors Exceptions

Recovering from Error is not possible. We can recover from exceptions by either using try-catch block or throwing exceptions back to the caller.

All errors in java are unchecked type. Exceptions include both checked as well as unchecked type.

Errors are mostly caused by the environment in which program is running. Program itself is responsible for causing exceptions.

Errors can occur at compile time as well as run time. Compile Time: eg Syntax Error

Run Time: Logical Error.

All exceptions occurs at runtime but checked exceptions are known to the compiler while unchecked are not.

They are defined in java.lang.Error package. They are defined in java.lang.Exception package

Examples : java.lang.StackOverflowError, java.lang.OutOfMemoryError Examples : Checked Exceptions : SQLException, IOException Unchecked Exceptions : ArrayIndexOutOfBoundException, NullPointerException, ArithmeticException.

*References:*

<https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/>