**Exception handling in Java**

## **What is an exception?**

* An Exception is an unwanted event that **interrupts the normal flow of the program**.
* When an exception occurs program execution gets terminated. In such cases we get a system generated error message.
* The good thing about exceptions is that java developer can handle these exception in such a way so that the program doesn’t get terminated abruptly and the user get a meaningful error message.

**For example:** You are writing a program for division and both the numbers are entered by user. In the following example, user can enter any number, if user enters the second number (divisor) as 0 then the program will terminate and throw an exception because dividing a number by zero gives undefined result. To get the user input, we are using [Scanner class](https://beginnersbook.com/2022/08/java-scanner-class-with-examples/). **Notice the output of the program**.

import java.util.Scanner;

public class JavaExample {

public static void main(String[] args) {

int num1, num2;

Scanner scan = new Scanner(System.in);

System.out.print("Enter first number(dividend): ");

num1 = scan.nextInt();

System.out.print("Enter second number(divisor): ");

num2 = scan.nextInt();

int div = num1/num2;

System.out.println("Quotient: "+div);

}

}

## **Exception Handling in Java**

import java.util.Scanner;

public class JavaExample {

public static void main(String[] args) {

int num1, num2;

Scanner scan = new Scanner(System.in);

System.out.print("Enter first number(dividend): ");

num1 = scan.nextInt();

System.out.print("Enter second number(divisor): ");

num2 = scan.nextInt();

try {

int div = num1 / num2;

System.out.println("Quotient: "+div);

}catch(ArithmeticException e){

System.out.println("Do not enter divisor as zero.");

System.out.println("Error Message: "+e);

}

}

}

If an exception occurs, which has not been handled by programmer then program execution gets terminated and a system generated error message is shown to the user.

These system generated messages are **not user friendly** so a user will not be able to understand what went wrong. In order to let them know the reason in simple language, we handle exceptions. We handle such exceptions and then prints a user friendly warning message to user, which lets them correct the error as most of the time **exception occurs due to bad data provided by user**.

## **Why we handle the exceptions?**

Exception handling ensures that the flow of the program doesn’t break when an exception occurs. For example, if a program has bunch of statements and an exception occurs mid-way after executing certain statements then the statements that occur after the statement that caused the exception will not execute and the program will terminate abruptly. By handling we make sure that all the statements execute and the flow of execution of program doesn’t break.

## **Why an exception occurs?**

There can be several reasons that can cause a program to throw exception. For example: Opening a non-existing file in your program, Network connection problem, bad input data provided by user etc. Let’s see few scenarios:

**1. ArithmeticException:**  
We have already seen this exception in our example above. This exception occurs when we divide a number by zero. If we divide any number by zero.

int num = 25/0;//ArithmeticException

**2. NullPointerException:**  
When a variable contains null value and you are performing an operation on the variable. For example, if a string variable contains null and you are comparing with another string. Another example is when you are trying to print the length of the string that contains null.

String str = null;

//NullPointerException

System.out.println(str.length());

**3. NumberFormatException:**  
This exception occurs where there is a type mismatch. Let’s say you are trying to perform an arithmetic operator on a string variable.

String str = "beginnersbook.com";

//NumberFormatException

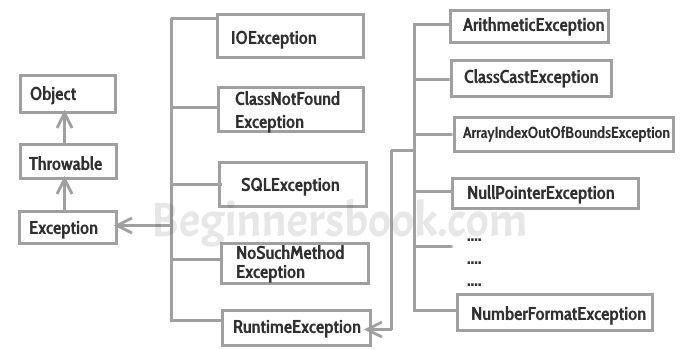
int num=Integer.parseInt(str);

**4. ArrayIndexOutOfBoundsException:**  
When you are trying to access the array index which is beyond the size of array. Here, we are trying to access the index 8 (9th element) but the size of the array is only 3. This exception occurs when you are accessing index which doesn’t exist.

int arr[]=new int[3];

//ArrayIndexOutOfBoundsException

arr[8]=100;



## Difference between error and exception

**Errors** indicate that something went wrong which is not in the scope of a programmer to handle. You cannot handle an error. Also, the error doesn’t occur due to bad data entered by user rather it indicates a system failure, disk crash or resource unavailability.

**Exceptions** are events that occurs during runtime due to bad data entered by user or an error in programming logic. A programmer can handle such conditions and take necessary corrective actions.

### Example of Multiple catch blocks

class Example2{

public static void main(String args[]){

try{

int a[]=new int[7];

a[4]=30/0;

System.out.println("First print statement in try block");

}

catch(ArithmeticException e){

System.out.println("Warning: ArithmeticException");

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("Warning: ArrayIndexOutOfBoundsException");

}

catch(Exception e){

System.out.println("Warning: Some Other exception");

}

System.out.println("Out of try-catch block...");

}

}

**Now lets change the code a little bit and see the change in output:**

class Example{

public static void main(String args[]){

try{

int arr[]=new int[7];

arr[10]=10/5;

System.out.println("Last Statement of try block");

}

catch(ArithmeticException e){

System.out.println("You should not divide a number by zero");

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("Accessing array elements outside of the limit");

}

catch(Exception e){

System.out.println("Some Other Exception");

}

System.out.println("Out of the try-catch block");

}

}

**Lets change the code again and see the output:**

class Example{

public static void main(String args[]){

try{

int arr[]=new int[7];

arr[10]=10/5;

System.out.println("Last Statement of try block");

}

catch(Exception e){

System.out.println("Some Other Exception");

}

catch(ArithmeticException e){

System.out.println("You should not divide a number by zero");

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("Accessing array elements outside of the limit");

}

System.out.println("Out of the try-catch block");

}

}

**Output:**

Compile time error: Exception in thread "main" java.lang.Error:

Unresolved compilation problems: Unreachable catch block for ArithmeticException.

It is already handled by the catch block for Exception Unreachable catch block

for ArrayIndexOutOfBoundsException. It is already handled by the catch block for

Exception at Example.main(Example1.java:11)