



SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai

Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &

Accredited by NBA (B.E - CSE, EEE, ECE, Mech & B.Tech.IT)

COIMBATORE-641 035, TAMIL NADU



UNIT IV - EXCEPTION AND MULTITHREADING

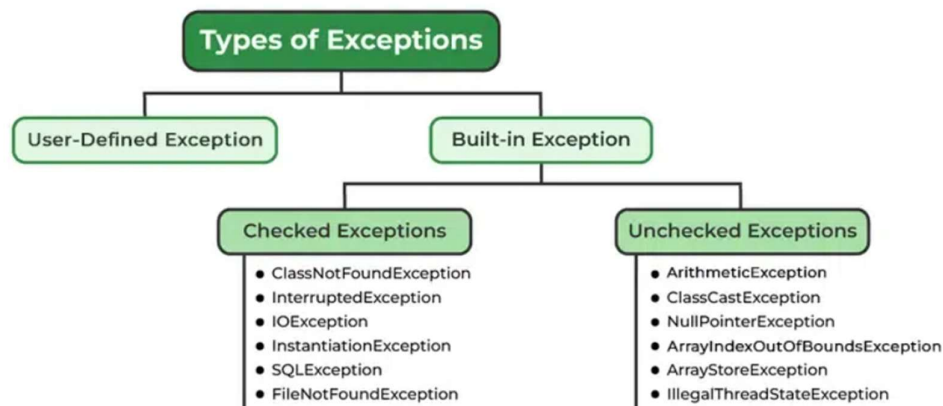
Exception handling -Exception types - try catch and finally block, throws -Runtime exception - Introduction to Multithreading - Thread Creation - Thread control and priorities - Thread synchronization, Inter-thread communication.

Runtime Exception

Exceptions Types

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception.

- Checked Exception
- Unchecked Exception
- Error



Examples

A. Arithmetic exception

```
class ArithmeticException_Demo
{
    public static void main(String args[])
    {
        try {
            int a = 30, b = 0;
            int c = a/b; // cannot divide by zero
            System.out.println ("Result = " + c);
        }
    }
}
```

```

    }
    catch(ArithmeticException e) {
        System.out.println ("Can't divide a number by 0");
    }
}
}

```

Output - Can't divide a number by 0

B. NullPointerException

```

class NullPointerException_Demo
{
    public static void main(String args[])
    {
        try {
            String a = null; //null value
            System.out.println(a.charAt(0));
        } catch(NullPointerException e) {
            System.out.println("NullPointerException..");
        }
    }
}

```

Output - NullPointerException..

C. StringIndexOutOfBoundsException

```

class StringIndexOutOfBoundsException_Demo
{
    public static void main(String args[])
    {
        try {
            String a = "This is like chipping "; // length is 22
            char c = a.charAt(24); // accessing 25th element
            System.out.println(c);
        }
        catch(StringIndexOutOfBoundsException e) {
            System.out.println("StringIndexOutOfBoundsException");
        }
    }
}

```

Output - StringIndexOutOfBoundsException

D. ArrayIndexOutOfBoundsException

```

class ArrayIndexOutOfBoundsException_Demo
{
    public static void main(String args[])
    {
        try{
            int a[] = new int[5];
            a[6] = 9; // accessing 7th element in an array of
                // size 5
        }
        catch(ArrayIndexOutOfBoundsException e){

```

```

        System.out.println ("Array Index is Out Of Bounds");
    }
}

```

Output - Array Index is Out Of Bounds

E. IllegalArgumentException:

```

/*package whatever //do not write package name here */

import java.io.*;
class GFG {
    public static void print(int a)
    {
        if(a>=18){
            System.out.println("Eligible for Voting");
        }
        else{
            throw new IllegalArgumentException("Not Eligible for Voting");
        }
    }
    public static void main(String[] args) {
        GFG.print(14);
    }
}

```

Output :

```

Exception in thread "main" java.lang.IllegalArgumentException: Not Eligible for
Voting
at GFG.print(File.java:13)
at GFG.main(File.java:19)

```

F. IllegalStateException:

```

/*package whatever //do not write package name here */
import java.io.*;
class GFG {
    public static void print(int a,int b)
    {
        System.out.println("Addition of Positive Integers :"+(a+b));
    }
    public static void main(String[] args) {
        int n1=7;
        int n2=-3;
        if(n1>=0 && n2>=0)
        {
            GFG.print(n1,n2);
        }
        else
        {
            throw new IllegalStateException("Either one or two numbers are not Positive
Integer");
        }
    }
}

```

Output :

```
Exception in thread "main" java.lang.IllegalStateException: Either one or two
numbers are not Positive Integer
at GFG.main(File.java:20)
```

User-Defined Exceptions

- Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, the user can also create exceptions which are called 'user-defined Exceptions'.
- can create our own exceptions that are derived classes of the Exception class. Creating our own Exception is known as custom exception or user-defined exception.

The following steps are followed for the creation of a user-defined Exception.

The user should create an exception class as a subclass of the Exception class.

Since all the exceptions are subclasses of the Exception class, the user should also make his class a subclass of it.

This is done as:

```
class MyException extends Exception
```

We can write a default constructor in his own exception class.

```
MyException()
{
}
```

We can also create a parameterized constructor with a string as a parameter.

We can use this to store exception details. We can call the superclass(Exception) constructor from this and send the string there.

```
MyException(String str)
{
    super(str);
}
```

To raise an exception of a user-defined type, we need to create an object to his exception class and throw it using the throw clause, as:

```
MyException me = new MyException("Exception details");
throw me;
```

Example

```
// class representing custom exception
```

```

class InvalidAgeException extends Exception
{
public InvalidAgeException (String str)
{
// calling the constructor of parent Exception
super(str);
}
}
// class that uses custom exception InvalidAgeException
public class TestCustomException1
{
// method to check the age
static void validate (int age) throws InvalidAgeException{
if(age < 18){
// throw an object of user defined exception
throw new InvalidAgeException("age is not valid to vote");
}
else {
System.out.println("welcome to vote");
}
}
// main method
public static void main(String args[])
{
try
{
// calling the method
validate(13);
}
catch (InvalidAgeException ex)
{
System.out.println("Caught the exception");
// printing the message from InvalidAgeException object
System.out.println("Exception occurred: " + ex);
}
System.out.println("rest of the code...");
}
}

```