EMPOWERING THROUGH INNOVATIONS
MLRS

MARRI LAXMAN REDDY
Institute of Technology and Management
(Approved By AICTE, New Delhi & Affillated to JNTU Hyderabad)
Dundigal,Medchal(M),Hyderabad,Telangana, India – 500 043.
(UGC - AUTONOMOUS)

Unit-III-PART-I

Exception Handling in Java

An exception in java programming is an abnormal situation that is araised during the program execution.

In simple words, an exception is a problem that arises at the time of program execution.

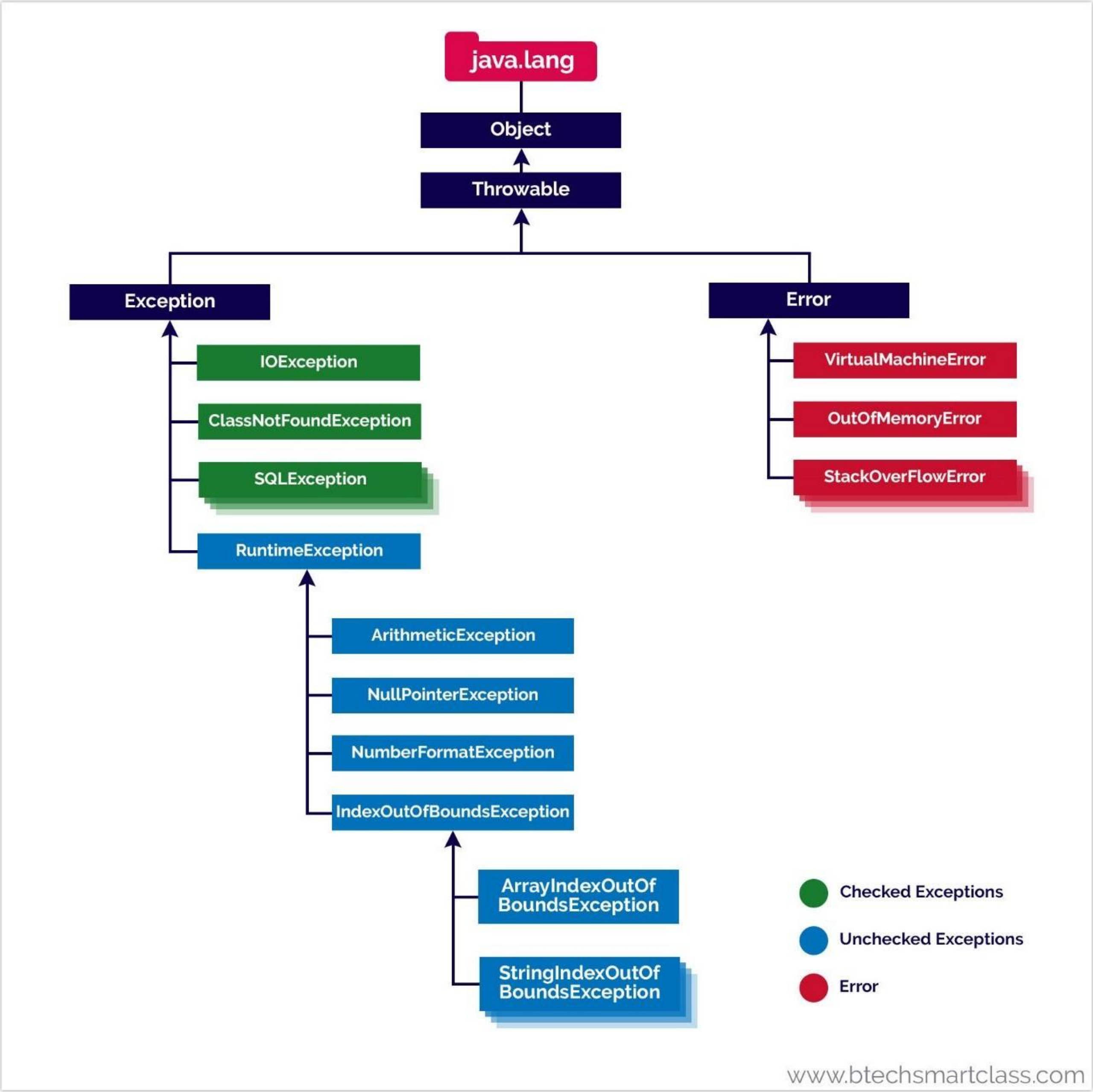
When an exception occurs, it disrupts the program execution flow.

When an exception occurs, the program execution gets terminated, and the systemgenerates an error.

We use the exception handling mechanism to avoid abnormal termination of programexecution.

Java programming language has a very powerful and efficient exception handling mechanism with a large number of built-in classes to handle most of the exceptionsautomatically.

java programming language has the following class hierarchy to support the exception handling mechanism.


java.lang
Object
Throwable
Exception
Error
IOException
ClassNotFoundException
SQLException
RuntimeException
ArithmeticException
NullPointerException
NumberFormatException
IndexOutOfBoundsException
ArrayIndexOutOf BoundsException
StringIndexOutOf BoundsException
VirtualMachineError
OutOfMemoryError
StackOverFlowError
Checked Exceptions
Unchecked Exceptions
Error
www.btechsmartclass.com

Reasons for Exception Occurrence

Several reasons lead to the occurrence of an exception. A few of them are as follows.

- When we try to open a file that does not exist may lead to an exception.
- When the user enters invalid input data, it may lead to an exception.
- When a network connection has lost during the program execution may leadto an exception.
- When we try to access the memory beyond the allocated range may lead toan exception.

- The physical device problems may also lead to an exception.

Types of Exception

In java, exceptions have categorized into two types, and they are as follows.

- **Checked Exception** - An exception that is checked by the compiler at the timeof compilation is called a checked exception.
- **Unchecked Exception** - An exception that can not be caught by the compilerbut occurrs at the time of program execution is called an unchecked exception.

How exceptions handled in Java?

In java, the exception handling mechanism uses five keywords namely ***try***, ***catch***, ***finally***, ***throw***, and ***throws***.

Exception Types in Java

In java, exceptions are mainly categorized into two types, and they are as follows.

- **Checked Exceptions**
- **Unchecked Exceptions**

Checked Exceptions

The checked exception is an exception that is checked by the compiler during the compilation process to confirm whether the exception is handled by the programmer or not. If it is not handled, the compiler displays a compilation error using built-in classes.

The checked exceptions are generally caused by faults outside of the code itself like missing resources, networking errors, and problems with threads come to mind.

The following are a few built-in classes used to handle checked exceptions in java.

- IOException
- FileNotFoundException
- ClassNotFoundException
- SQLException
- DataAccessException
- InstantiationException
- UnknownHostException

The checked exception is also known as a compile-time exception.

Unchecked Exceptions

The unchecked exception is an exception that occurs at the time of program execution. The unchecked exceptions are not caught by the compiler at the time of compilation.

The unchecked exceptions are generally caused due to bugs such as logic errors, improper use of resources, etc.

The following are a few built-in classes used to handle unchecked exceptions in java.

- ArithmeticException
- NullPointerException
- NumberFormatException
- ArrayIndexOutOfBoundsException
- StringIndexOutOfBoundsException

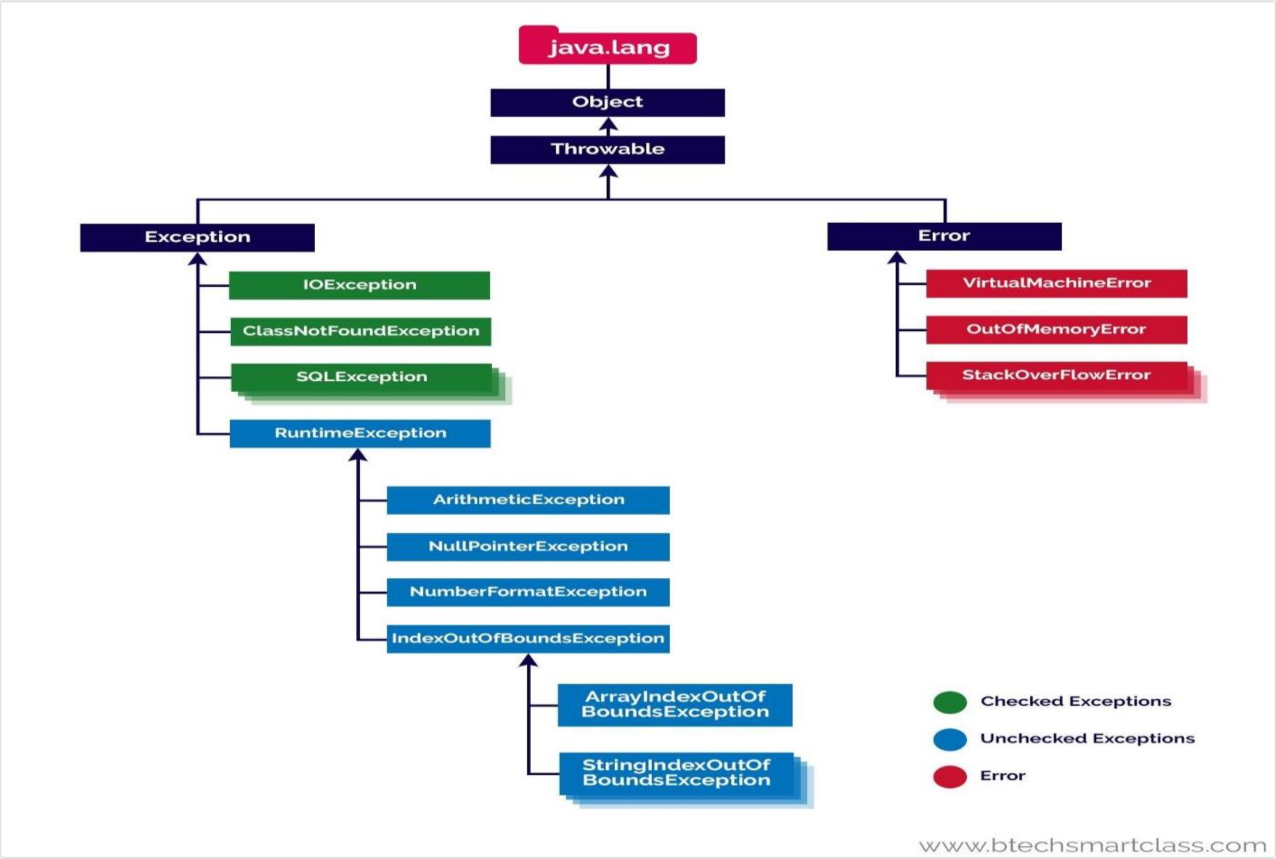
The unchecked exception is also known as a runtime exception.

Example - Unchecked Exceptions

```
public class UncheckedException {

          public static void main(String[] args) {

                    int list[] = {10, 20, 30, 40, 50};

                    System.out.println(list[6]);
          //ArrayIndexOutOfBoundsException

                    String msg=null;
                    System.out.println(msg.length()); //NullPointerException

                    String name="abc";
                    int i=Integer.parseInt(name);
          //NumberFormatException

          }

}
```

Exception class hierarchy

In java, the built-in classes used to handle exceptions have the following classhierarchy.



Exceptions in Java-try and catch

In java, the **try**try and **catch**, both are the keywords used for exception handling.

The keyword try is used to define a block of code that will be tests the occurence ofan exception.

The keyword catch is used to define a block of code that handles the exceptionoccured in the respective try block.

The uncaught exceptions are the exceptions that are not caught by the compiler butautomatically caught and handled by the Java built-in exception handler.

Both try and catch are used as a pair.

Every try block must have one or more catch blocks.

We can not use try without atleast one catch, and catch alone can be used (catchwithout try is not allowed).

The following is the syntax of try and catch blocks.

Syntax

```
try{
    ...
    code to be tested
    ...
}
catch(ExceptionType object){
    ...
    code for handling the exception
    ...
}
import java.util.Scanner;

public class TryCatchExample {
    public static void main(String[] args) {

        Scanner read = new Scanner(System.in);
        System.out.println("Enter the a and b values: ");
        try {
            int a = read.nextInt();
            int b = read.nextInt();
            int c = a / b;
            System.out.println(a + "/" + b +" = " + c);
        }
        catch(ArithmeticException ae) {
            System.out.println("Problem info: Value of divisor can not be ZERO");
        }
    }
}
```

Multiple catch clauses

In java programming language, a try block may has one or more number of catchblocks.

That means a single try statement can have multiple catch clauses.

When a try block has more than one catch block, each catch block must contain adifferent exception type to be handled.

The multipe catch clauses are defined when the try block contains the code that may leadto different type of exceptions.

```
public  class  TryCatchExample  {

    public  static  void  main(String[]  args)  {

        try  {
            int  list[]  =  new  int[5];
            list[2]  =  10;

            list[4]  =  2;
            list[10]  =  list[2]  /  list[4];
        }
        catch(ArithmeticException  ae)  {
            System.out.println("Problem info: Value of divisor can not be ZERO.");
        }
        catch(ArrayIndexOutOfBoundsException  aie)  {
            System.out.println("Problem info: ArrayIndexOutOfBoundsException
has occured.");
        }
        catch(Exception  e)  {
            System.out.println("Problem info: Unknown exception has occured.");
        }
    }
}
```

Nested try statements

The java allows to write a try statement inside another try statement. A try block within another try block is known as nested try block.

When there are nested try blocks, each try block must have one or more separate catch blocks.

```
public  class  TryCatchExample  {

    public static void main(String[] args) {

        try {
            int list[] = new int[5];
            list[2]  =  10;
            list[4] = 2;
            list[0] = list[2] / list[4];
            try {
                list[10]  =  100;
```

throw, throws, and finally keywords in Java

In java, the keywords throw, throws, and finally are used in the exception handlingconcept. Let's look at each of these keywords.

throw keyword in Java

The throw keyword is used to throw an exception instance explicitly from a try blockto corresponding catch block.

That means it is used to transfer the control from try block to corresponding catchblock.

The throw keyword must be used inside the try block.

When JVM encounters the throw keyword, it stops the execution of try block andjump to the corresponding catch block.

The following is the general syntax for using throw keyword in a try block.

Syntax

```
throw instance;
```

Here the instace must be throwable instance and it can be created dynamically using newoperator.

```
import java.util.Scanner;

public class Sample {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int num1, num2, result;
        System.out.print("Enter any two numbers: ");
        num1 = input.nextInt();
        num2 = input.nextInt();
        try {
            if(num2 == 0)
                throw new ArithmeticException("Division by zero is not posible");
            result = num1 / num2;
            System.out.println(num1 + "/" + num2 + "=" + result);
        }
        catch(ArithmeticException ae) {
            System.out.println("Problem info: " + ae.getMessage());
        }
            System.out.println("End of the program");
    }
}
```

throws keyword in Java

The throws keyword specifies the exceptions that a method can throw to the defaulthandler and does not handle itself.

That means when we need a method to throw an exception automatically, we usethrows keyword followed by method declaration

```
import  java.util.Scanner;
 public  class  ThrowsExample
{
        int num1, num2, result;
        Scanner  input =  new  Scanner(System.in);
        void division() throws ArithmeticException
        { System.out.print("Enter any two numbers: ");
                num1 =  input.nextInt();
                num2 =  input.nextInt();
                result =  num1 /  num2;
                System.out.println(num1 +  "/"  +  num2 +  "="  +  result);
        }
        public  static  void  main(String[]  args)
          {

                try
                {       new  ThrowsExample().division();

                }
                catch(ArithmeticException  ae)  {
                        System.out.println("Problem info: "  +
                        ae.getMessage());
                }
                System.out.println("End of the program");
        }
}
```

finally keyword in Java

The finally keyword used to define a block that must be executed irrespective ofexception occurence.

The basic purpose of finally keyword is to cleanup resources allocated by try block,such as closing file, closing database connection, etc.

```
import java.util.Scanner;


public class FinallyExample {


    public static void main(String[] args)
        {int num1, num2, result;
        Scanner input = new
        Scanner(System.in);
        System.out.print("Enter any two numbers:
        "); num1 = input.nextInt();
        num2 = input.nextInt();
        try {
            if(num2 == 0)
                throw new ArithmeticException("Division by
                                        zero");
            result = num1 / num2;
            System.out.println(num1 + "/" + num2 + "=" + result);
        }
        catch(ArithmeticException ae) {
            System.out.println("Problem info: " +
            ae.getMessage());
        }
```

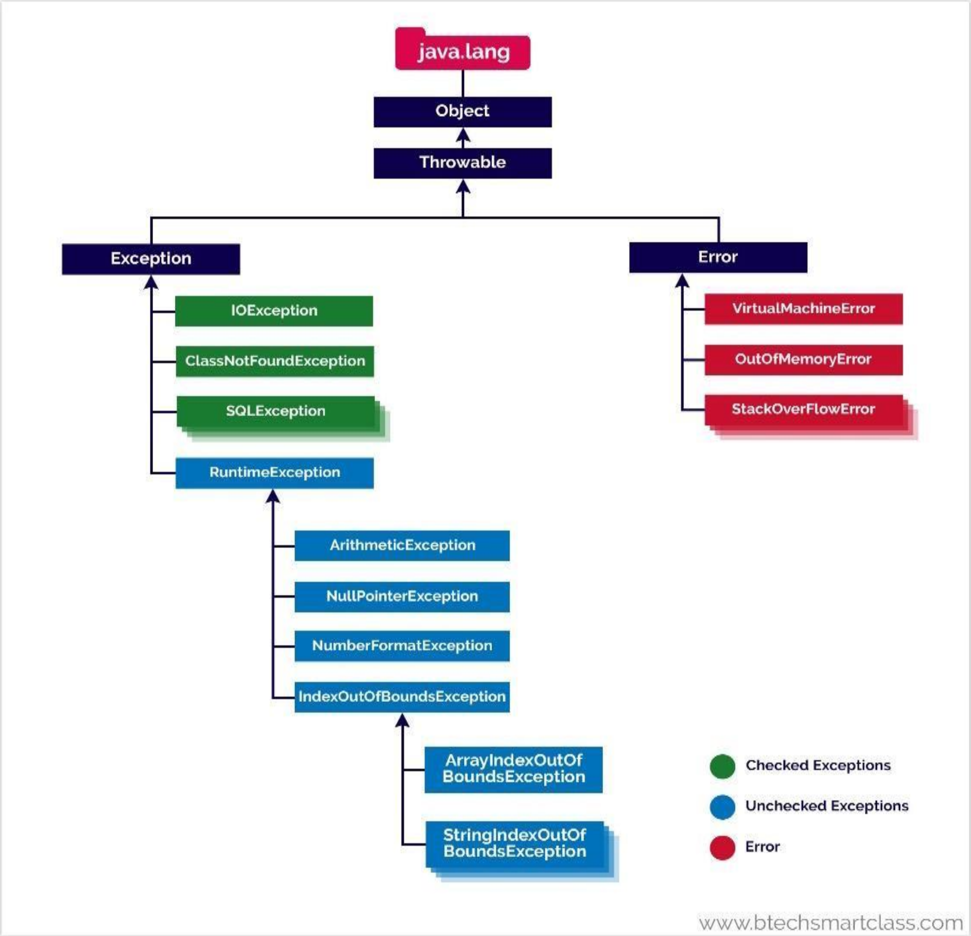
```java
        finally {
            System.out.println("The finally block executes always");
        }
        System.out.println("End of the program");
    }


}
```

Built-in Exceptions in Java

The Java programming language has several built-in exception class that support exception handling. Every exception class is suitable to explain certain error situationsat run time.

All the built-in exception classes in Java were defined a package **java.lang**.

Few built-in exceptions in Java are shown in the following image.



List of checked exceptions in Java

The following table shows the list of several checked exceptions.

S. No.	Exception Class with Description
1	**ClassNotFoundException** It is thrown when the Java Virtual Machine (JVM) tries to load a particular class and the sp
2	**CloneNotSupportedException** Used to indicate that the clone method in class Object has been called to clone an objectinterface.
3	**IllegalAccessException** It is thrown when one attempts to access a method or member that visibility qualifiers do
4	**InstantiationException** It is thrown when an application tries to create an instance of a class using the newInstancinstantiated because it is an interface or is an abstract class.
5	**InterruptedException** It is thrown when a thread that is sleeping, waiting, or is occupied is interrupted.
6	**NoSuchFieldException** It indicates that the class doesn't have a field of a specified name.
7	**NoSuchMethodException** It is thrown when some JAR file has a different version at runtime that it had at compile tito access a method that does not exist.

List of unchecked exceptions in Java

The following table shows the list of several unchecked exceptions.

S. No.	Exception Class with Description
1	**ArithmeticException** It handles the arithmetic exceptions like dividion by zero
2	**ArrayIndexOutOfBoundsException** It handles the situations like an array has been accessed with an illegal index. The index is
3	**ArrayStoreException** It handles the situations like when an attempt has been made to store the wrong type of
4	**AssertionError** It is used to indicate that an assertion has failed
5	**ClassCastException** It handles the situation when we try to improperly cast a class from one type to another.
6	**IllegalArgumentException** This exception is thrown in order to indicate that a method has been passed an illegal or
7	**IllegalMonitorStateException** This indicates that the calling thread has attempted to wait on an object's monitor, or has without owning the specified monitor.
8	**IllegalStateException** It signals that a method has been invoked at an illegal or inappropriate time.
9	**IllegalThreadStateException** It is thrown by the Java runtime environment, when the programmer is trying to modify th
10	**IndexOutOfBoundsException** It is thrown when attempting to access an invalid index within a collection, such

	as an arra
11	**NegativeArraySizeException**

It is thrown if an applet tries to create an array with negative size.

S. No.	Exception Class with Description
	It is thrown if an applet tries to create an array with negative size.
12	**NullPointerException** it is thrown when program attempts to use an object reference that has the null value.
13	**NumberFormatException** It is thrown when we try to convert a string into a numeric value such as float or integer, b
14	**SecurityException** It is thrown by the Java Card Virtual Machine to indicate a security violation.
15	**StringIndexOutOfBounds** It is thrown by the methods of the String class, in order to indicate that an index is either n

Creating Own Exceptions in Java

The Java programming language allow us to create our own exception classes whichare basically subclasses built-in class **Exception**.

To create our own exception class simply create a class as a subclass of built-inException class.

We may create constructor in the user-defined exception class and pass a string toException class constructor using **super()**.

We can use **getMessage()** method to access the string.