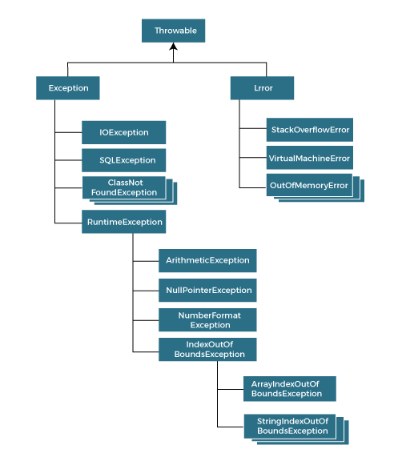
***What is Exception in Java?***

*In Java, an exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. These exceptions can occur for various reasons, such as invalid user input, file not found, or division by zero. When an exception occurs, it is typically represented by an object of a subclass of the java.lang.Exception class.*

## *What is Exception Handling?*

*Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.*



## *Types of Java Exceptions*

*In Java, exceptions are categorized into two main types: checked exceptions and unchecked exceptions. Additionally, there is a third category known as errors. Let's delve into each of these types:*

1. *Checked Exception*
2. *Unchecked Exception*
3. *Error*

## *1. Checked Exceptions*

*Checked exceptions are the exceptions that are checked at compile-time. This means that the compiler verifies that the code handles these exceptions either by catching them or declaring them in the method signature using the throws keyword. Examples of checked exceptions include:*

***IOException:****An exception is thrown when an input/output operation fails, such as when reading from or writing to a file*.

***SQLException****: It is thrown when an error occurs while accessing a database.*

***ParseException****: Indicates a problem while parsing a string into another data type, such as parsing a date.*

***ClassNotFoundException****: It is thrown when an application tries to load a class through its string name using methods like Class.forName(), but the class with the specified name cannot be found in the classpath.*

## *2. Unchecked Exceptions (Runtime Exceptions)*

*Unchecked exceptions, also known as runtime exceptions, are not checked at compile-time. These exceptions usually occur due to programming errors, such as logic errors or incorrect assumptions in the code. They do not need to be declared in the method signature using the throws keyword, making it optional to handle them. Examples of unchecked exceptions include:*

***NullPointerException:****It is thrown when trying to access or call a method on an object reference that is null.*

***ArrayIndexOutOfBoundsException:****It occurs when we try to access an array element with an invalid index.*

***ArithmeticException:****It is thrown when an arithmetic operation fails, such as division by zero.*

***IllegalArgumentException:****It indicates that a method has been passed an illegal or inappropriate argument*.

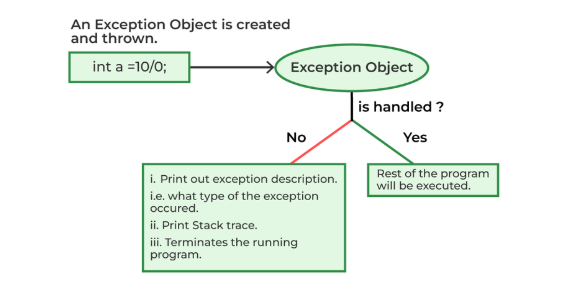
## *3. Errors*

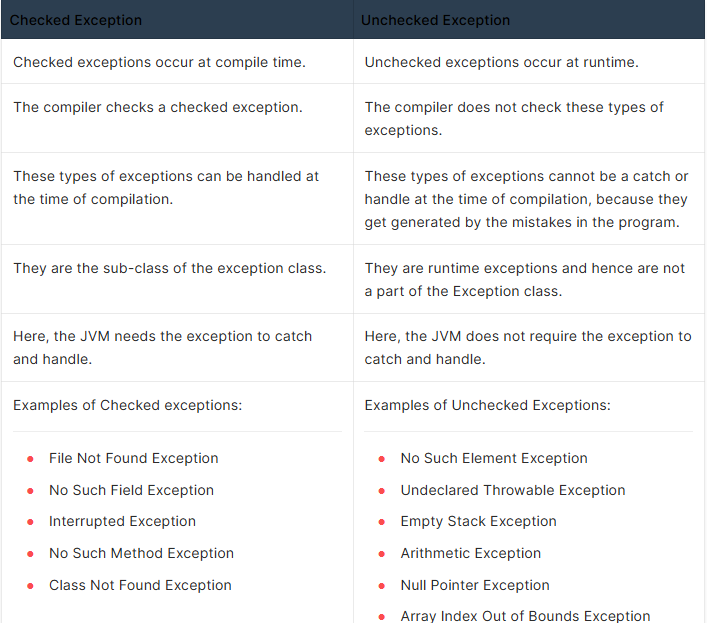
*Errors represent exceptional conditions that are not expected to be caught under normal circumstances. They are typically caused by issues outside the control of the application, such as system failures or resource exhaustion. Errors are not meant to be caught or handled by application code. Examples of errors include:*

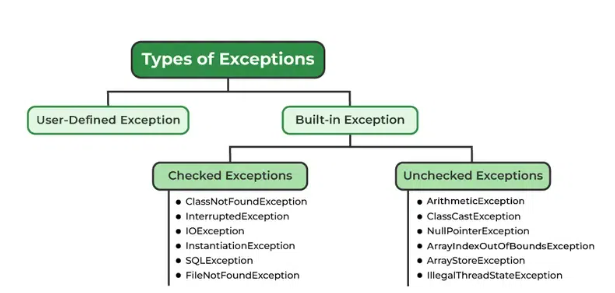
***OutOfMemoryError:****It occurs when the Java Virtual Machine (JVM) cannot allocate enough memory for the application.*

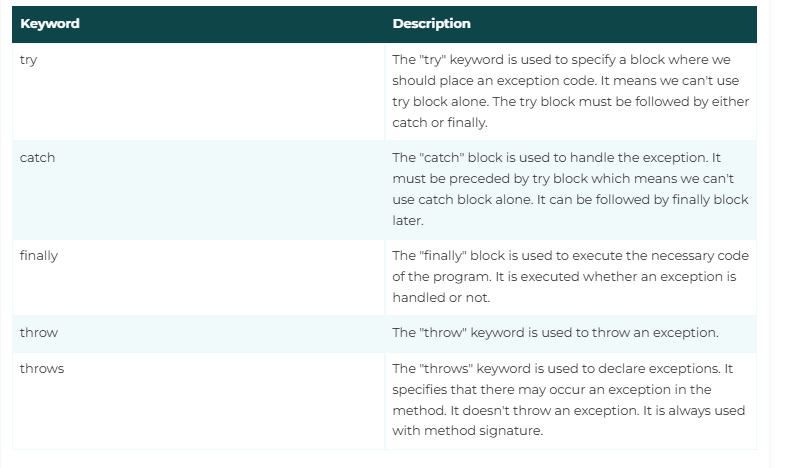
***StackOverflowError****: It is thrown when the stack memory is exhausted due to excessive recursion.*

***NoClassDefFoundError****: It indicates that the JVM cannot find the definition of a class that was available at compile-time.*

**

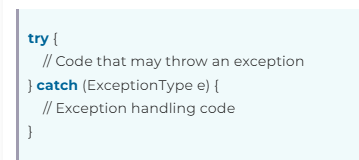
******

**

**

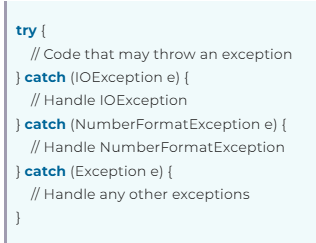
## *The try-catch Block*

*The try block contains the code that may throw an exception, and the catch block is used to handle the exception if it occurs.*



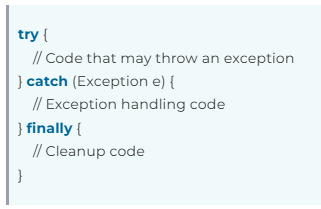
## *Handling Multiple Exceptions*

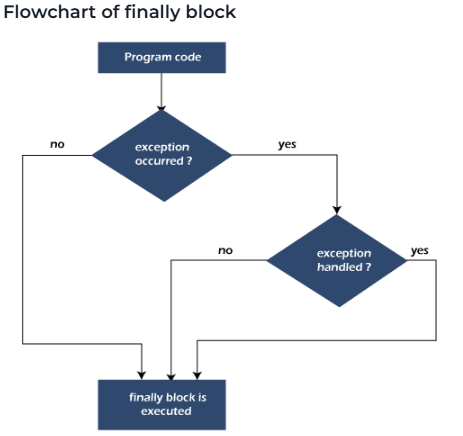
*You can handle multiple types of exceptions by providing multiple catch blocks, each catching a different type of exception*

**

## *The finally Block*

*Java also provides a finally block, which allows you to execute cleanup code, such as closing resources, regardless of whether an exception occurs or not. The finally block is typically used to release resources that were acquired in the try block.*

**

******

***Example 1***

***public class JavaExceptionExample{***

***public static void main(String args[]){***

***try{***

***//code that may raise exception***

***int data=100/0;***

***}catch(ArithmeticException e){System.out.println(e);}***

***//rest code of the program***

***System.out.println("rest of the code...");***

***}***

***}***

***O/p-> java.lang.ArithmeticException: / by zero***

***rest of the code..****.*

***Example 2***

***public class NullPointerExceptionExample {***

***public static void main(String[] args) {***

***String str = null; // Initializing a String variable to null***

***try {***

***int length = str.length(); // Attempting to call a method on a null reference***

***System.out.println("Length of the string: " + length);***

***} catch (NullPointerException e) {***

***System.out.println("Error: Null reference encountered.");***

***// Additional error handling code can be added here***

***}***

***}***

***}***

**O*/p-> Error: Null reference encountered.***

***Example 3:***

***public class NumberFormatExceptionExample {***

***public static void main(String[] args) {***

***String str = "abc"; // Initializing a String with non-numeric characters***

***try {***

***int num = Integer.parseInt(str); // Attempting to parse a non-numeric string to an integer***

***System.out.println("Parsed number: " + num);***

***} catch (NumberFormatException e) {***

***System.out.println("Error: Unable to parse the string as an integer.");***

***// Additional error handling code can be added here***

***}***

***}***

***}***

***O/P***

*Error: Unable to parse the string as an integer.*

***Example 4***

***public class ArrayIndexOutOfBoundsExceptionExample {***

***public static void main(String[] args) {***

***int[] numbers = {1, 2, 3, 4, 5}; // Initializing an array with 5 elements***

***try {***

***int index = 10; // Accessing an index that is out of bounds***

***int value = numbers[index]; // Attempting to access an element at an invalid index***

***System.out.println("Value at index " + index + ": " + value);***

***} catch (ArrayIndexOutOfBoundsException e) {***

***System.out.println("Error: Index is out of bounds.");***

***// Additional error handling code can be added here***

***}***

***}***

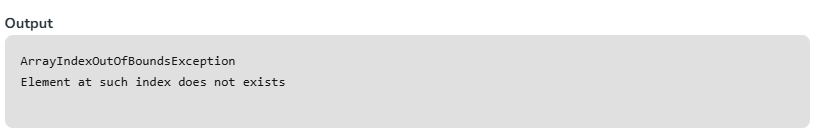
***}***

***O/P***

*Error: Index is out of bounds.*

***Example Nested try catch***

******

******

## *Exception Propogation*

*An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method. If not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This is called exception propagation.*

***class TestExceptionPropagation1{***

***void m(){***

***int data=50/0;***

***}***

***void n(){***

***m();***

***}***

***void p(){***

***try{***

***n();***

***}catch(Exception e){System.out.println("exception handled");}***

***}***

***public static void main(String args[]){***

***TestExceptionPropagation1 obj=new TestExceptionPropagation1();***

***obj.p();***

***System.out.println("normal flow...");***

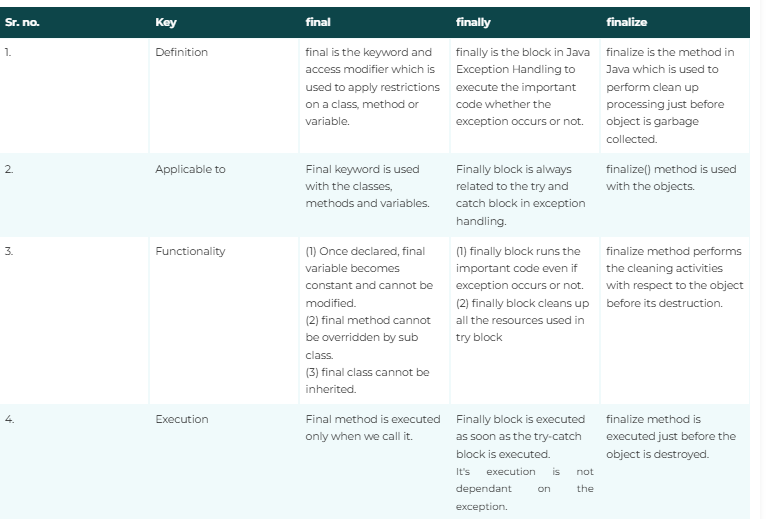
***}***

***}***

***Output***

*exception handled*

*normal flow...*

**

## *Java Custom Exception*

*In Java, we 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. Basically, Java custom exceptions are used to customize the exception according to user need.*

***// 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***

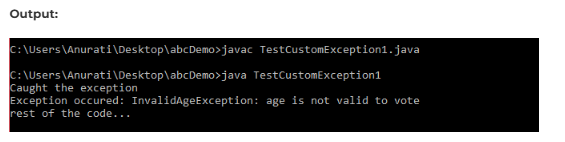
***System.out.println("Exception occured: " + ex);***

***}***

***System.out.println("rest of the code...");***

***}***

***}***

******