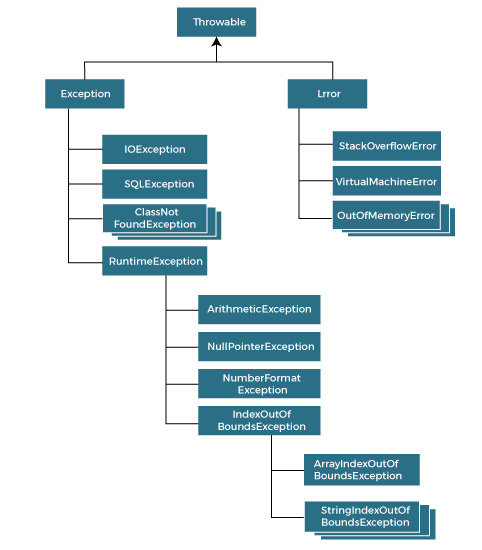
Java Exception handling

**Exception** is 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 and handling those errors so that normal flow of the application can be preserved is called exception handling such as class not found exception ,ioException etc.

Reasons for Exception Occurrence:

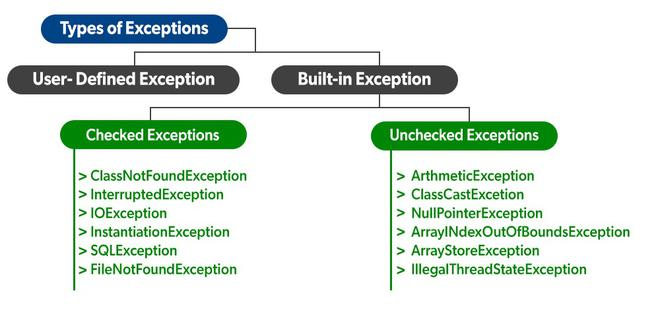
* Invalid user Input
* Device Failure
* Loss of network connection
* Physical Limitation (out of disk memory)
* Code errors
* Opening an Unavailable file

**Hierarchy of java Exception classes :**



# **Types of Exceptions in java**

1. Checked exception
2. Unchecked exception
3. Error



# **1. Exception:**

Exception indicates a condition that a reasonable application might try to catch.

## **1.1 Checked Exception:**

These Exceptions are the built in exceptions and it is compile time exceptions because these exceptions are checked at compile -time by the compiler.

## **1.1 UnChecked Exception:**

These Exceptions are the built in exceptions and they are opposite to checked exceptions. The compiler will not check these exceptions at compile time .

# **2. Error**

An Error indicated a serious problem that a reasonable application should not try to catch. Irrecoverable conditions such as java virtual machines running out of memory ,memory leaks ,stack overflow errors .

# **3 Try -catch Block:**

Java try catch is used to enclose the code that might throw an exception . if an exception occurs at a particular statement in the try block ,the rest of the code will not execute.so it is recommended not to keep the code the try block that will not throw an exception.

| /\*\* \* @author rohit.tamang \* @Created 10/17/2022 \*/ public class Test {  public static void main(String[] args) throws ParseException {  try {  int a = 10 / 0;  System.out.println("rest of the code will not execute");  } catch (Exception e) {  System.out.println(e);  }  } }  Output :java.lang.ArithmeticException: / by zero  //handle exception  public class Test {  public static void main(String[] args) throws ParseException {  try {  int a = 10 / 0;    } catch (ArithmeticException e) {  System.out.println(e);  }  System.out.println("rest of the code will not execute");  } }  Output :java.lang.ArithmeticException: / by zero  rest of the code will not execute |
| --- |

# **4 multiple Try -catch Block:**

Try blocks followed by one or more catch blocks each block must have a different Exception handler . so that different exceptions are caught from different tasks . at this point only one exception occurs and at a time only one catch block is executed and catch blocks must be ordered from most specific to most general i,e catch for Arithmetic Exception must come before catch for exception.

| **/\*\* \* @author rohit.tamang \* @Created 10/17/2022 \*/ public class Test {  public static void main(String[] args) throws ParseException {  try {  int a[] = new int[5];  a[5] = 30 / 0;  } catch (ArithmeticException e) {  System.out.println("Arithmetic Exception occurs");  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("ArrayIndexOutOfBounds Exception occurs");  } catch (Exception e) {  System.out.println("Parent Exception occurs");  }  System.out.println("rest of the code");  } }** |
| --- |

# **5 Nested Try -catch Block:**

Try catch block which has multiple try catch block inside it is refer as the Nested try catch block.Every statement that we enter a statement in try block ,context of that exception is pushed on to the stack.

**Q Why use nested try blocks?**

Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error. In such cases exception handlers have to be nested.

Syntax :

| main try catch.... try {  statment1.....  statmenet2....  try {  statment3.....  statmenet4....  try {  statment5.....  statmenet6....   } catch (Exception e3) {   }    } catch (Exception e2) {  } } catch (Exception e1) { } |
| --- |

Example :-

| /\*\* \* @author rohit.tamang \* @Created 10/17/2022 \*/ public class Test {  public static void main(String[] args) {  //outer try block  try {  //inner try block 1  try {  System.out.println("going to divide by 0");  int b = 39 / 0;  }  //catch block of inner try block 1  catch (ArithmeticException e) {  System.out.println(e);  }   //inner try block 2  try {  int a[] = new int[5];   //assigning the value out of array bounds  a[5] = 4;  }   //catch block of inner try block 2  catch (ArrayIndexOutOfBoundsException e) {  System.out.println(e);  }   System.out.println("other statement");  }  //catch block of outer try block  catch (Exception e) {  System.out.println("handled the exception (outer catch)");  }  System.out.println("normal flow..");  } }  OutPut :  going to divide by 0  java.lang.ArithmeticException: / by zero  java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5  other statement  normal flow.. |
| --- |

When any try catch block does not have a catch block for a particular exception then the catch block of the outer (parent ) try catch block is checked for that exception and if it matches the catch block of the outer try block is executed.

If none of the catch blocks specified in the code is unable to handle the exception then the java runtime exception will handle the exception. The is displays the system generated message for that exception.

# **6 java throws Keyword:**

It gives the programmers information that there may occur an exception . so the normal flow of the program can be maintained. It falls under checked exceptions.

Example :

| public class Test {  void m()throws IOException {  throw new IOException("device error");//checked exception  }  void n()throws IOException{  m();  }  void p(){  try{  n();  }catch(Exception e){System.out.println("exception handled");}  }  public static void main(String args[]){  Test obj=new Test();  obj.p();  System.out.println("normal flow...");  } }  Output :  exception handled  normal flow... |
| --- |

#### Rule: If we are calling a method that declares an exception, we must either caught or declare the exception.

Example 1 : handle Exception using try catch block

| class M{  void method()throws IOException{  throw new IOException("device error");  } } public class Testthrows2{  public static void main(String args[]){  try{  M m=new M();  m.method();  }catch(Exception e){System.out.println("exception handled");}   System.out.println("normal flow...");  } }  Output  exception handled  normal flow... |
| --- |

Example 1 : Declare Exception

In case we declare the exception and the exception occurs it will be thrown at runtime because throws does not handle the exception.

| class M{  void method()throws IOException{  System.out.println("device operation performed");  } } class Testthrows3{  public static void main(String args[])throws IOException{//declare exception  M m=new M();  m.method();   System.out.println("normal flow...");  } }  Output  device operation performed  normal flow... |
| --- |

# **7 java throw Keyword:**

The java throw keyword is used to throw an exception explicitly. We specify the exception object which is to be thrown . the exceptions may be related to user inputs ,server etc. we can throw either checked or unchecked exceptions in java by throw a keyword. It is mainly used to throw a custom exception.

Example : -

| Throw new exception\_class(“error message”)  Throw new ioException(“sory device error”) |
| --- |

Exception is the subclass of throwable and the user-defined exceptions usually extend the exception class.

**Example : throwing unchecked exception**

| **/\*\* \* @author rohit.tamang \* @Created 10/17/2022 \*/ public class TestThrow1 {  //function to check if person is eligible to vote or not  public static void validate(int age) {  if(age<18) {  //throw Arithmetic exception if not eligible to vote**  **Int a=10/0;  throw new ArithmeticException("Person is not eligible to vote");  }  else {  System.out.println("Person is eligible to vote!!");  }  }  //main method  public static void main(String args[]){  //calling the function  validate(13);  System.out.println("rest of the code...");  } }**  **Output :**  **Exception in thread "main" java.lang.ArithmeticException: Person is not eligible to vote**  **at Abstraction.TestThrow1.validate(TestThrow1.java:21)**  **at Abstraction.TestThrow1.main(TestThrow1.java:30)** |
| --- |

**Example : throwing checked exception**

| **/\*\* \* @author rohit.tamang \* @Created 10/17/2022 \*/ public class TestThrow2 {   //function to check if person is eligible to vote or not  public static void method() throws FileNotFoundException {   FileReader file = new FileReader("C:\\Users\\Anurati\\Desktop\\abc.txt");  BufferedReader fileInput = new BufferedReader(file);    throw new FileNotFoundException();   }  //main method  public static void main(String args[]){  try  {  method();  }  catch (FileNotFoundException e)  {  e.printStackTrace();  }  System.out.println("rest of the code...");  } }**  **Output :**  **java.io.FileNotFoundException: C:\Users\Anurati\Desktop\abc.txt (The system cannot find the path specified)**  **at java.base/java.io.FileInputStream.open0(Native Method)**  **at java.base/java.io.FileInputStream.open(FileInputStream.java:219)**  **at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)**  **at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)**  **at java.base/java.io.FileReader.<init>(FileReader.java:60)**  **at Abstraction.TestThrow2.method(TestThrow2.java:24)**  **at Abstraction.TestThrow2.main(TestThrow2.java:35)**  **rest of the code...** |
| --- |

**Example : - Custom exception**

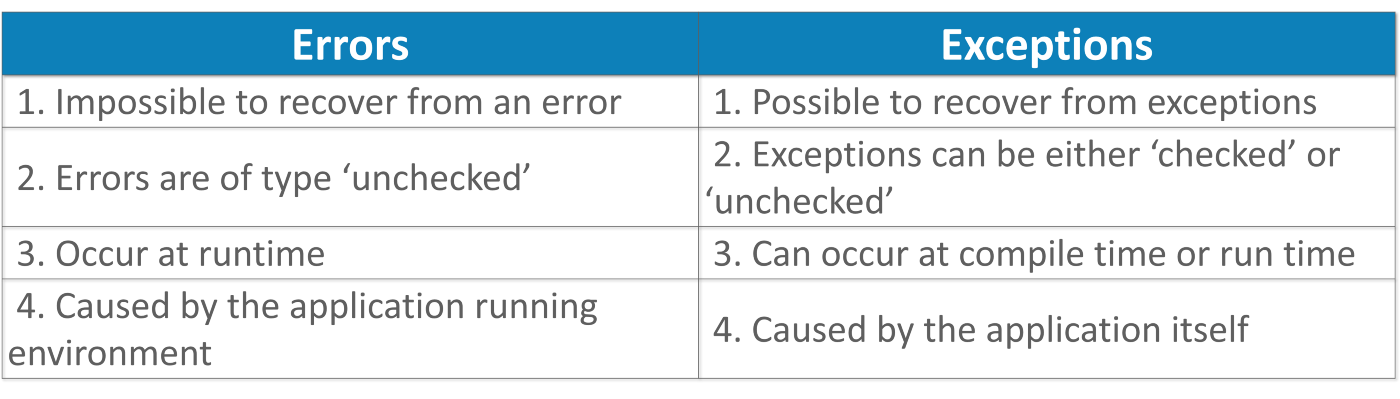
| **/\*\* \* @author rohit.tamang \* @Created 10/17/2022 \*/ class InvalidAgeException extends Exception {  public InvalidAgeException(String str) {  // calling the constructor of parent Exception  super(str);  } }  // class that uses custom exception InvalidAgeException public class TestCustomException1 {   // 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...");  } }** |
| --- |

# 

# 

# 

# **Java error and exception difference**



| class StackExample {  public static void check(int i)  {  if (i == 0)  return;  else {  check(i++);  }  } } public class Main {   public static void main(String[] args)  {  StackExample.check(5);  } }  Output:  Jav Lang stackoverflow error |
| --- |