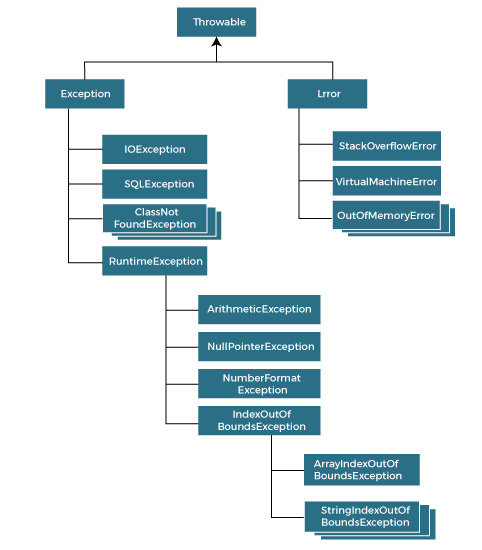
**Day 21**

# **Exception Handling in Java** The **Exception Handling in Java** is one of the powerful mechanisms to handle the runtime errors so that the normal flow of the application can be maintained.

* An exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.
* Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



### **Types of Java Exceptions**

There are mainly three types of exceptions: namely:

1. Checked Exception
2. Unchecked Exception
3. Error

**1) Checked Exception:** Checked exceptions are checked at compile-time. The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc.

**2)** **Unchecked Exception:** Unchecked exceptions are not checked at compile-time, but they are checked at runtime. The classes that inherit the RuntimeException are known as unchecked exceptions.

ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc.

### **3) Error:** Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

## **Java Exception Keywords**

1. **try:** The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
2. **catch:** The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
3. **finally:** The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
4. **throw:** The "throw" keyword is used to throw an exception.
5. **throws:** The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

Syntax

try-catch

try

{

//code

}catch(Exception class reference)

try-finally

try

{

//code

}

Finally

{

}

Example

class Test

{

public static void main(String args[])

{

int d = 50/0; // arithmetic exception

System.out.println(“Rest of the code”);

}

}

class Test

{

public static void main(String args[])

{

try

{

int d = 50/0; // arithmetic exception

} catch(arithmeticException e)

{

System.out.println(e);

}

System.out.println(“Rest of the code”);

}

}

class Test

{

public static void main(String args[])

{

try

{

int d = 50/0; // arithmetic exception

} catch(arithmeticException e)

{

System.out.println(e);

}

finally

{

System.out.println(“Finally block executed”);

}

System.out.println(“Rest of the code”);

}

}

Throw keyword

Example

class Test

{

public static void validate(int age)

{

if(age<18)

{

throw new ArithmeticException(“Person not elegible”);

}

else

{

System.out.println(“Elegible”);

}

}

public static void main(String args[])

{

validate(10);

System.out.println(“Rest of the code”);

}

}

**Common Sinarios of Exception**

* Arithmetic Exception:

int a = 10/0;

* NullPointer Exception:

String s = null;

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

* NumberFormat Exception:

String s = “abc”;

int a = Integer.parseInt(s);

* ArrayIndexOutOfBound Exception:

int a[] = new int[5];

a[10] = 30;

**Throws Keyword**

Syntax

method name() throws Exception\_class\_name

{

//method body

}

Example:

class test

{

void m() throws IOException

{

throw new IOException(“device error”);

}

}

class Test

{

public static void main(String args[])

{

try

{

int d =25/5;

System.out.println(d);

}

catch(NullPointerException e)

{

System.out.println(e);

}

finally

{

System.out.println("Finally block");

}

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

}

}

OUTPUT

5

Finally block

Rest of the code

// catching exception in above program

class Test

{

public static void main(String args[])

{

try

{

System.out.println("Inside block");

int d =25/0;

System.out.println(d);

}

catch(NullPointerException e)

{

System.out.println(e);

}

finally

{

System.out.println("Finally block");

}

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

}

}

Output:

Inside block

Finally block

// since in catch block NullPointerException is catched, but, here in code it is arithmetic exception is occurred. So, catch block is not executed, jumped to finally block.

// catching exception in above program

class Test

{

public static void main(String args[])

{

try

{

System.out.println("Inside block");

int d =25/0;

System.out.println(d);

}

catch(ArithmeticException e)

{

System.out.println(Arithmetic Exception Handled);

System.out.println(e);

}

finally

{

System.out.println("Finally block");

}

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

}

}

Output:

Inside block

Arithmetic Exception handled

Arithmetic exeception

Finally block

Rest of the code