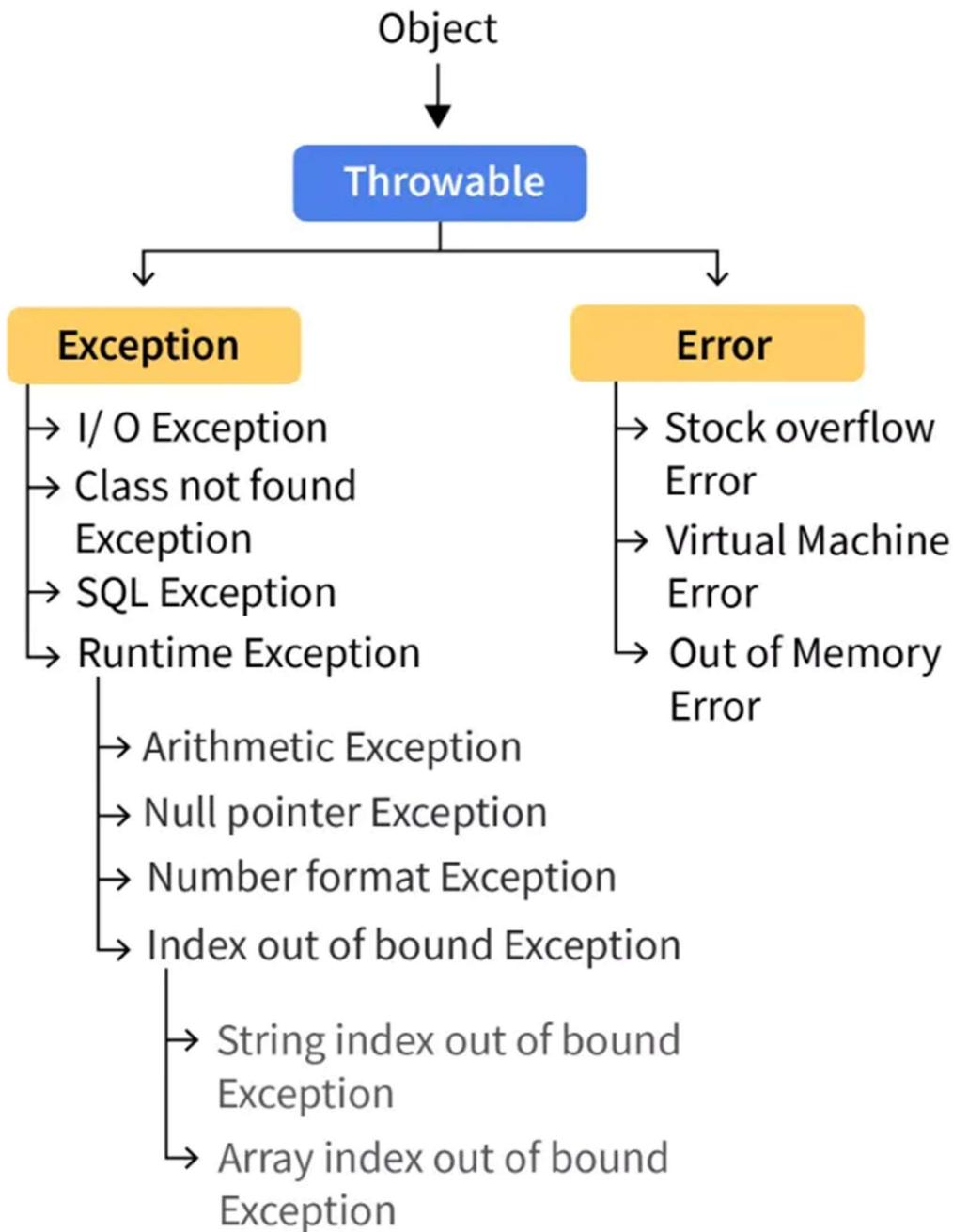


Hierarchy of exception in java



Finally keyword in java :-

What is final keyword :-

- Finally block is used to execute important code such as resource closing, **whether an exception occurs or not.**

Purpose :-

- To ensure cleanup actions (like closing files, releasing resources) always run.

Syntax :-

```
try
{
    // code that may cause exception
}
catch (Exception e)
{
    // handling code
}
finally
{
    // code that always executes
}
```

Implementation of finally keyword

Input :-

```
1 package exception_handling;
2
3 public class finally_demo {
4
5     public static void main(String[] args) {
6
7         try
8         {
9             int a=4/0;
10        }
11        catch(Exception e)
12        {
13            System.out.println(e);
14        }
15        finally
16        {
17            System.out.println("Finally Block Executed....!!!");
18        }
19        System.out.println("Bye....!");
20    }
21 }
```

Output :-

```
java.lang.ArithmetcException: / by zero
Finally Block Executed....!!!
Bye....!
```

Throw keyword in java :-

What is throw keyword :-

- Throw is used to **explicitly throw an exception** in Java.

Purpose :-

- Manually trigger (raise) an exception when a specific condition occurs.

Syntax :-

throw throwableObject;

- **throw:** This is the keyword that initiates the throwing of an exception.
- **throwableObject:** This must be an instance of a class that directly or indirectly inherits from the `java.lang.Throwable` class.
- This includes `Exception` and `Error` classes, and their subclasses

Implementation of throw keyword

Input :-

```
1 package exception_handling;
2
3 import java.util.Scanner;
4
5 public class throw_keyword {
6
7     public static void main(String[] args) throws InvalidAgeException {
8
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the age : ");
11
12        int age=sc.nextInt();
13
14        if(age>=18)
15        {
16            System.out.println("Eligible for voting");
17        }
18        else
19        {
20            throw new InvalidAgeException("Not Eligible for voting");
21        }
22    }
23 }
```

Output :-

```
Enter the age :
16
Exception in thread "main" exception_handling.InvalidAgeException: Not Eligible for voting
at exception_handling.throw_keyword.main(throw_keyword.java:20)
```

Difference between throw and throws

Throw	Throws
Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
Checked exception cannot be propagated using throw only.	Checked exception can be propagated with throws.
Throw is followed by an instance.	Throws is followed by class.
Throw is used within the method.	Throws is used with the method signature.
You cannot throw multiple exceptions.	You can declare multiple exceptions e.g. public void method()throws IOException,SQLEception.

Customized Exception

What is Customized Exception :-

- A customized (user-defined) exception is a type of exception created by the programmer to handle specific application-related errors in Java's built-in exceptions.

Purpose of customized exception :-

- To represent **custom error conditions** in your program.
- Makes the code more **readable and meaningful**.
- Helps in **specific exception handling** according to business logic.

Steps to Create a Custom Exception :-

- **Create a class with name**, for ex- InvalidAgeException.
- **Extends with exception class**.
- **Create a parameterizes constructor**.
- **Pass the message in parent class** as well.

Key Points :-

- **extends Exception** → Handle the custom exception
- **extends RuntimeException** → Optional to handle custom exception
- Custom exceptions help make the program **more descriptive and maintainable**.

Syntax of Checked exceptions :-

```
class MyCustomCheckedException extends Exception
{
    public MyCustomCheckedException(String message)
    {
        super(message);
    }
}
```

Syntax of Unchecked exceptions :-

```
class MyCustomUncheckedException extends
RuntimeException
{
    public MyCustomUncheckedException(String
message)
    {
        super(message);
    }
}
```

// AgeException.java

6

package pack;

1

public class AgeException extends Exception

{

2

3

public AgeException(String s)

4

{

super(s);

5

}

}

Implementation of Customized Exception

Input :-

```
1 package exception_handling;
2
3 public class InvalidAadharNoException extends Exception{
4
5     public InvalidAadharNoException(String msg)
6     {
7         super(msg);
8     }
9 }
1 package exception_handling;
2
3 import java.util.Scanner;
4
5 public class aadhar_number {
6
7     public static void main(String[] args) throws InvalidAadharNoException {
8
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter Your Aadhar Number : ");
11        String num=sc.next();
12
13        if(num.length() !=12)
14        {
15            throw new InvalidAadharNoException("Invalid Aadhar Number");
16        }
17        else
18        {
19            System.out.println("Valid Aadhar Number : " +num);
20        }
21    }
22 }
```

Output :-

```
Enter Your Aadhar Number :
234564568
Exception in thread "main" exception_handling.InvalidAadharNoException: Invalid Aadhar Number
at exception_handling.aadhar_number.main(aadhar_number.java:15)
```