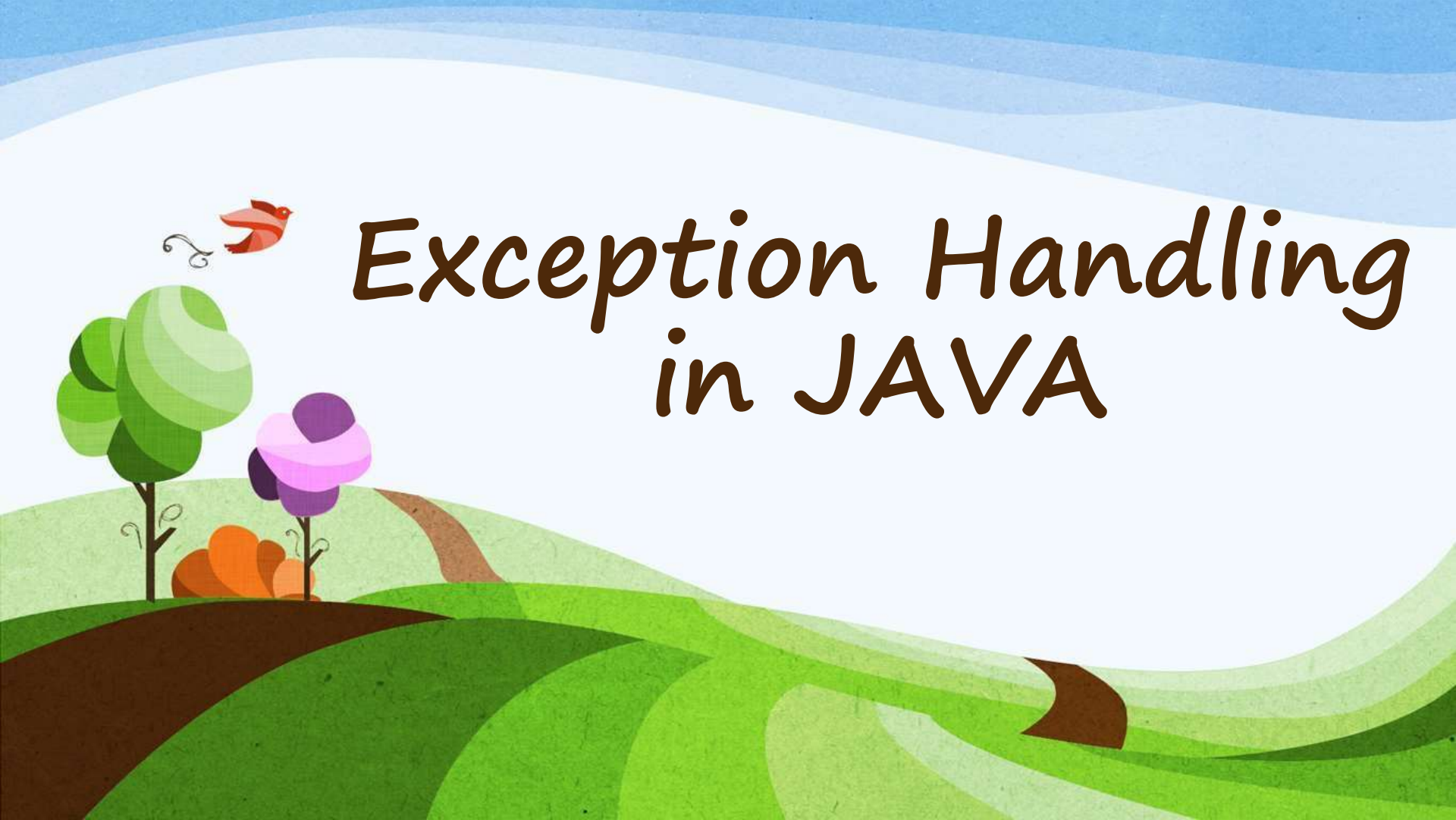

Exception Handling
in JAVA

What is Exception?

- **An exception is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program terminates abnormally**

- **Exception Handling is a mechanism to handle runtime errors.**

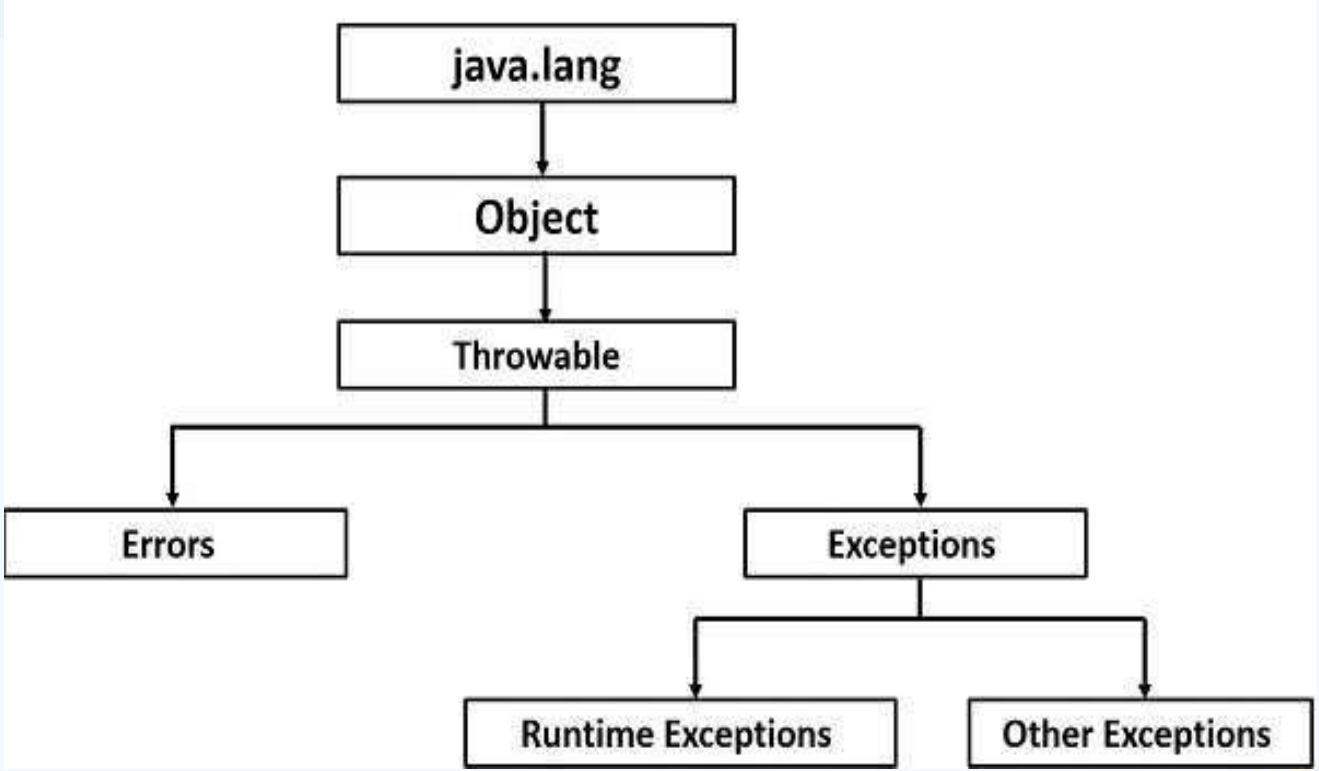
Situations in which exception can occur :-

- User entering invalid data.
- Opening a non-existing file.
- Network connections problem.
- Number format exception.

Types of Exception

- **Checked exceptions** – A checked exception is an exception that occurs at the compile time, these are also called as Compile time exceptions.
- **Unchecked exceptions** – An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions.
- **Errors** – These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. e.g. OutOfMemoryError, VirtualMachineError.

Exception Hierarchy



Try-Catch Block

- **Try block-** It is used to enclose the code that might throw an exception. It must be used within the method.
- **Catch block-** It is used to handle the Exception. It must be used after the try block only. It involves declaring the type of exception you are trying to catch.

Syntax:-

```
try
{
 // Protected code
}
catch(ExceptionName e)
{
// Catch block
}
```

Example (without exception handling)

```
public class Testtrycatch1{
    public static void main(String args[]){
        int data=50/0;
        System.out.print("rest ");
        System.out.print(“of");
        System.out.print(“the");
        System.out.print(“code");
    }
}
```

Output- Exception in thread main java.lang.ArithmeticException:/ by zero

Example (with exception handling)

```
public class Testtrycatch2{
  public static void main(String args[]){
   try{
 int data=50/0;
   }
catch(ArithmeticException e)
{   System.out.println(e);
}
   System.out.println("rest of the code...");
} }
```

Output- Exception in thread main java.lang.ArithmeticException:/ by zero
rest of the code...

Multiple Catch Block

```
public class TestMultipleCatchBlock{
  public static void main(String args[]){
  try{
  int a[]=new int[5];
 a[5]=30/0;
  }
 catch(ArithmeticException e){System.out.println(“e");}
catch(ArrayIndexOutOfBoundsException e){System.out.println(“e");}
System.out.println("rest of the code...");   }
}
```

Output- Exception in thread main java.lang.ArithmeticException:/ by zero
rest of the code...

Some common Sub-Classes of exception are:-

- **ArithmeticException** – If we divide any number by zero, there occurs an ArithmeticException

```
int a=50/0;
```

- **NullPointerException** - If we have null value in any variable, performing any operation by the variable occurs an NullPointerException.

```
String s=null;

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

- **ArrayIndexOutOfBoundsException** - If we are inserting any value in the wrong index, it would result ArrayIndexOutOfBoundsException.

```
int a[]=new int[5];

a[10]=50;
```

- **NumberFormatException** – The wrong formatting of any value, may occur NumberFormatException.

```
String s="abc";

int i=Integer.parseInt(s);
```

Finally block

- **Finally block is a block that is used *to execute important code.***
- **It is always executed whether exception is handled or not.**
- **The finally block follows a try block or a catch block.**

```
try {
// Protected code
 }
catch (ExceptionType1 e1)
{ // Catch block
}
catch (ExceptionType2 e2)
{ // Catch block
 }
catch (ExceptionType3 e3)
{ // Catch block
}
finally
{
// The finally block always executes.
}
```

Example

```
public class TestFinallyBlock2{
 public static void main(String args[]){
 try{
  int data=25/0;
      System.out.println(data); }
 catch(ArithmeticException e){
      System.out.println(e);}
 finally{
      System.out.println("finally  block will execute");}
 } }
```

Output:Exception in thread main java.lang.ArithmeticException:/ by zero
finally block will execute

User defined exception

- If you are creating your own Exception that is known as custom exception or user-defined exception.

- All exceptions must be a child of Throwable.

Throw keyword

- **Java throw keyword is used to explicitly throw an exception.**
- **We can throw either checked or uncheked exception in java by throw keyword.**
- **The throw keyword is mainly used to throw user defined exception.**

Syntax:- *throw exception;*

Throws Keyword

- The throws keyword is used to declare an exception.
- It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.
- Syntax:-

```
return_type method_name() throws exception_class_name{
//method code }
```

Difference between throw and throws in Java

throw	throws
Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
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,SQLException.