Exception Handling in Java

BY
M. BABY ANUSHA,
ASST.PROF IN CSE DEPT.,
RGUKT,NUZVID

Introduction :

- The **Exception Handling in Java** is one of the powerful *mechanism to handle the runtime errors*.
- With Exception Handling mechanism the normal flow of the application can be maintained.

What is Exception in Java?

- **Dictionary Meaning:** Exception is an abnormal condition.
- **Definition:** An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.
- An exception (or exceptional event) 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/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

What is Exception in Java?

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the JVM has run out of memory.

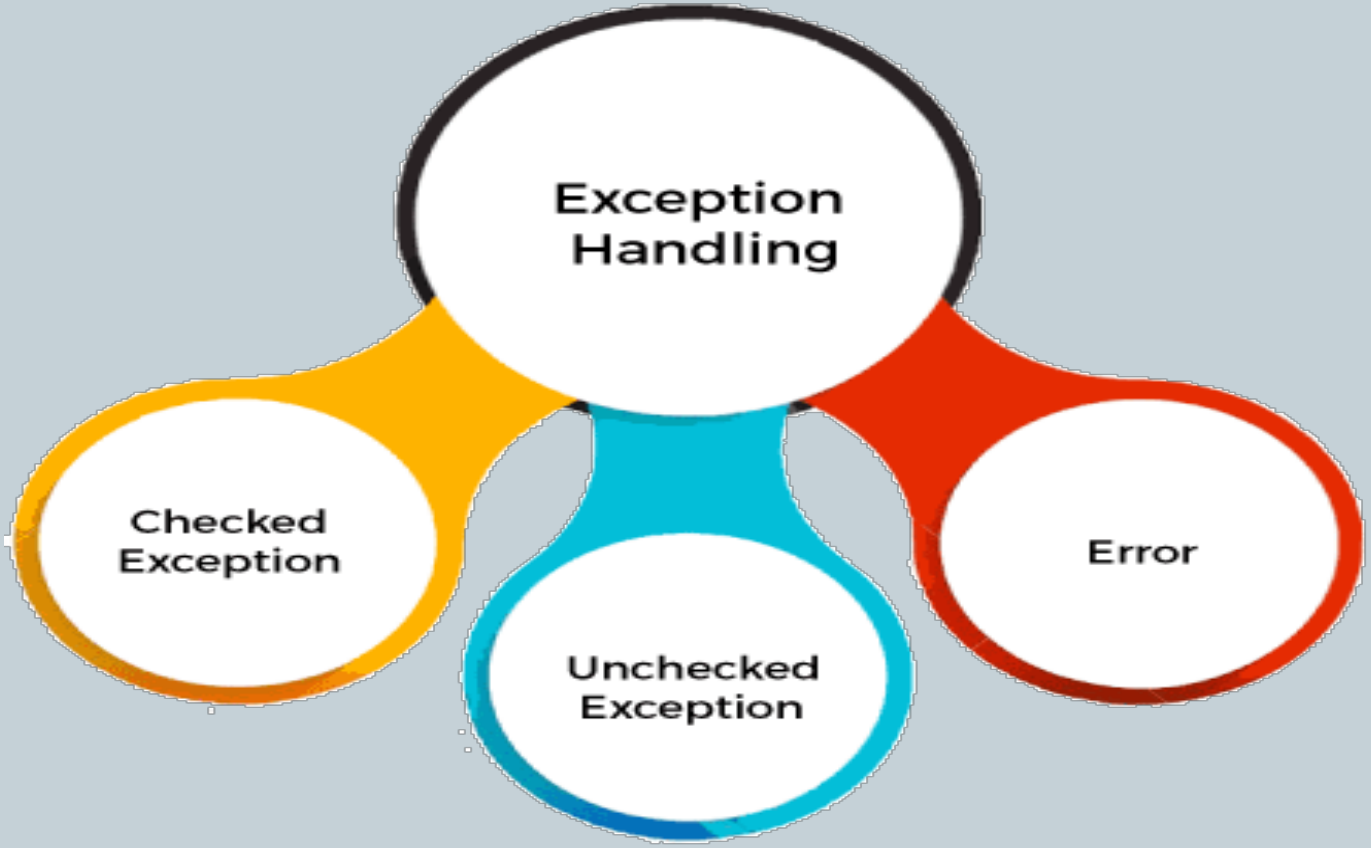
What is Exception in Java

- Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.
- Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.

Types of Java Exceptions :

- There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:
- Checked Exception
- Unchecked Exception
- Error

Types of Java Exceptions :



Difference between Checked and Unchecked Exceptions :

1) Checked Exception

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

2) Unchecked Exception

- The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3) Error

- Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

Checked exceptions :

- **Checked exceptions** – A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions.
- These exceptions cannot simply be ignored, the programmer should take care of (handle) these exceptions.
- For example, if you use **FileReader** class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a *FileNotFoundException* occurs, and the compiler prompts the programmer to handle the exception.

Checked exceptions : Example

```
import java.io.File;
 import java.io.FileReader;
public class FilenotFound_Demo
 {
   public static void main(String args[])
   {
   File file = new File("E://file.txt");
   FileReader fr = new FileReader(file);
   }
}
```

Checked exceptions : Example

- If you try to compile the above program, you will get the following exceptions.

Output :

- C:\>javac FilenotFound_Demo.java FilenotFound_Demo.java:8: error: unreported exception FileNotFoundException; must be caught or declared to be thrown FileReader fr = new FileReader(file); ^ 1 error

Checked exceptions : Example

- **Note** – Since the methods **read()** and **close()** of FileReader class throws IOException, you can observe that the compiler notifies to handle IOException, along with FileNotFoundException.

Unchecked exceptions :

- **Unchecked exceptions** – An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions**. These include programming bugs, such as logic errors or improper use of an API.
- Runtime exceptions are ignored at the time of compilation.
- For example, if you have declared an array of size 5 in your program, and trying to call the 6^{th} element of the array then an *ArrayIndexOutOfBoundsExceptionexception* occurs.

Unchecked exceptions : Example

```
public class UncheckedDemo
{
   public static void main(String args[])
   {
        int num[] = {1, 2, 3, 4};
        System.out.println(num[5]);
   }
 }
```

Unchecked exceptions : Example

If you compile and execute the above program, you will get the following exception.

Output :

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 5 at
Exceptions.Unchecked_Demo.main(Unchecked_De
mo.java:8)
```

Errors :

- **Errors** – These are not exceptions at all, but problems that arise beyond the control of the user or the programmer.
- Errors are typically ignored in your code because you can rarely do anything about an error.
- For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

What is Exception Handling?

- Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.
- The core advantage of exception handling is **to maintain the normal flow of the application**.
- An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

Advantage of Exception Handling :

- Let's consider a scenario:

statement 1;

statement 2;

statement 3;

statement 4;

statement 5;//exception occurs

statement 6;

statement 7;

statement 8;

statement 9;

statement 10;

Advantage of Exception Handling :

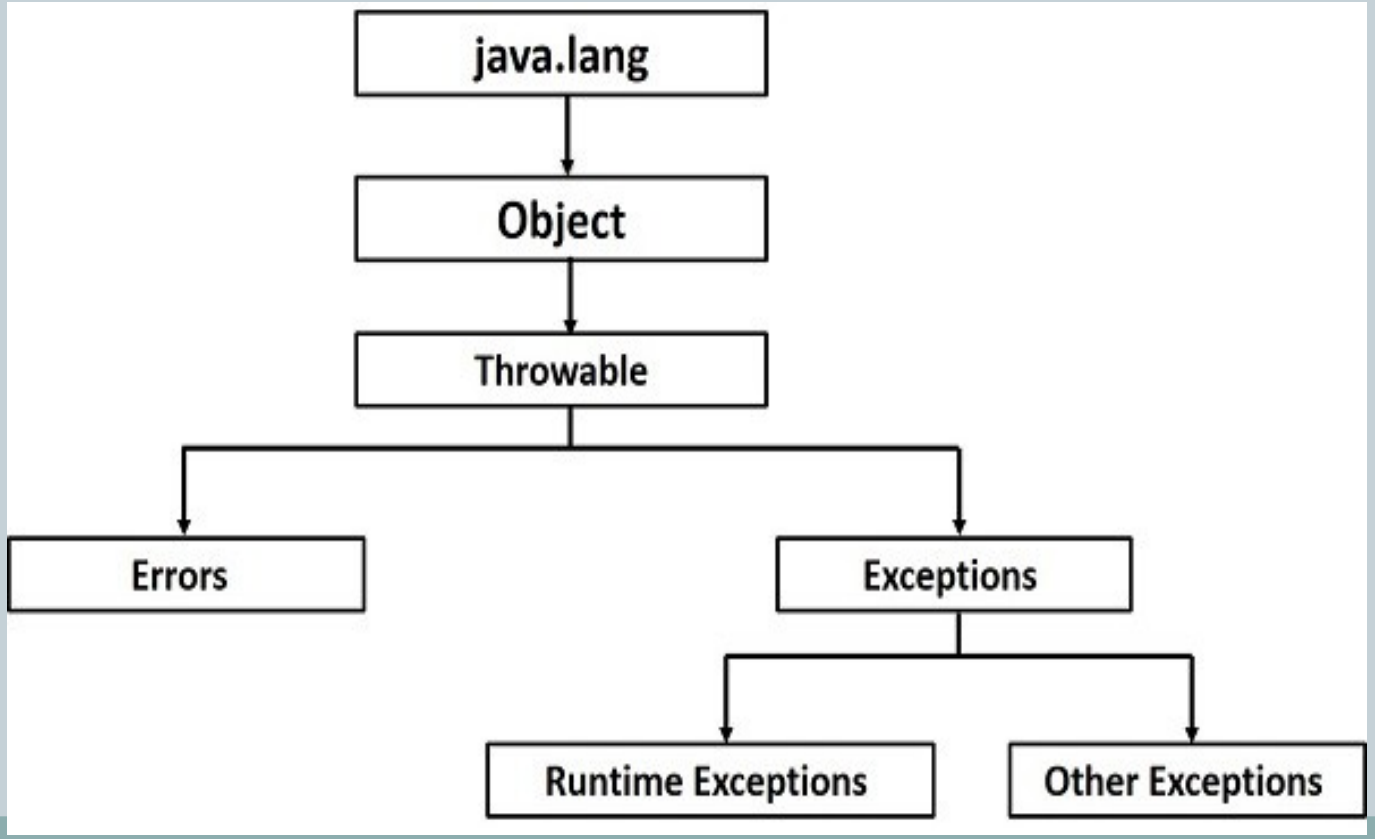
- Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed.
-]However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in Java.

Exception Hierarchy :

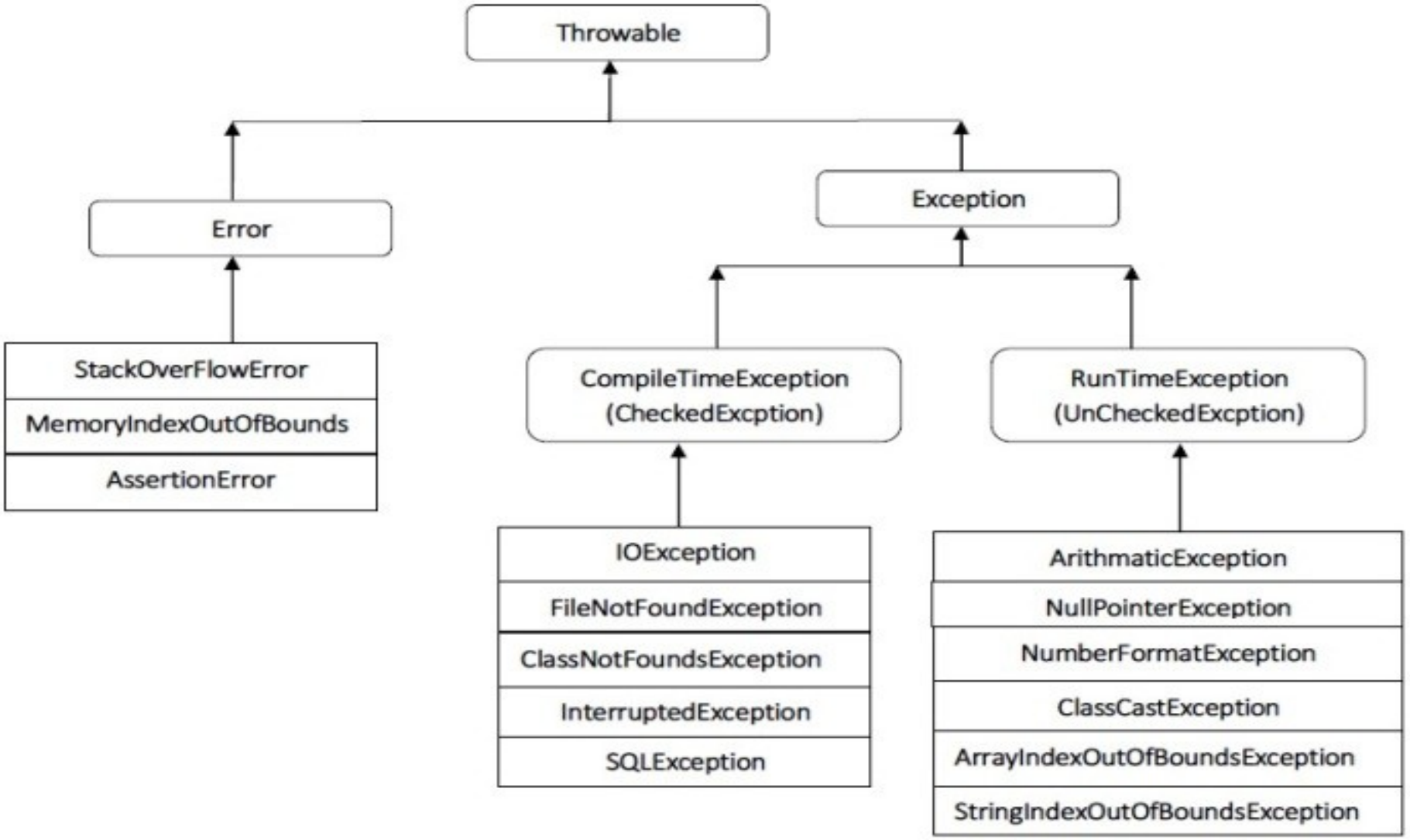
- All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class.
- Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment.
- Example: JVM is out of memory. Normally, programs cannot recover from errors.

Exception Hierarchy :

- The Exception class has two main subclasses: IOException class and RuntimeException Class.



Exception Hierarchy :



Java Exception Keywords :

- Java provides five keywords that are used to handle the exception. The following table describes each.

try keyword :

- 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.

catch keyword:

- 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.

finally keyword:

- The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.

throw keyword :

- The "throw" keyword is used to throw an exception.

throws keyword :

- 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.

Java Exception Handling : Example

- Let's see an example of Java Exception Handling in which we are using a try-catch statement to handle the exception.

```
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...");
 }
}
```

Java Exception Handling : Example

Output:

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


Thank you!
shutterstock.com · 567687052