

Exception Handling in Java:

Exception:

- An 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.

Exception Handling:

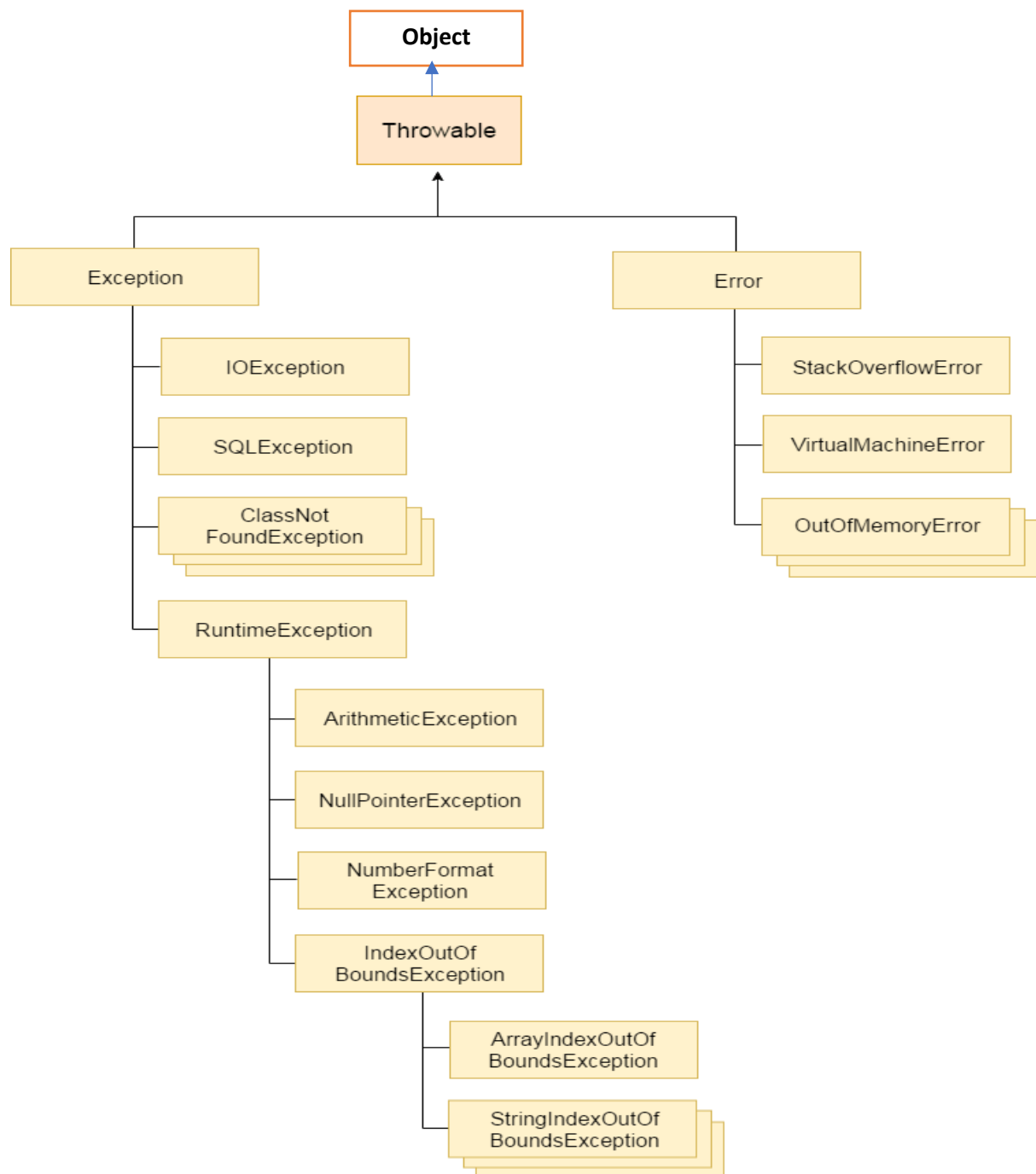
- The **Exception Handling in Java** is one of the powerful *mechanisms to handle the runtime errors* so that normal flow of the application can be maintained.
- Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.
- 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 use exception handling.

Error vs Exception

Error: An Error indicates serious problem that a reasonable application should not try to catch.

Exception: Exception indicates conditions that a reasonable application might try to catch.

Hierarchy of Java Exception classes:



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

Types of Java Exceptions:

1. Checked Exception
2. Unchecked Exception

1. Checked Exception: The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

2. Unchecked Exception: The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

Note:

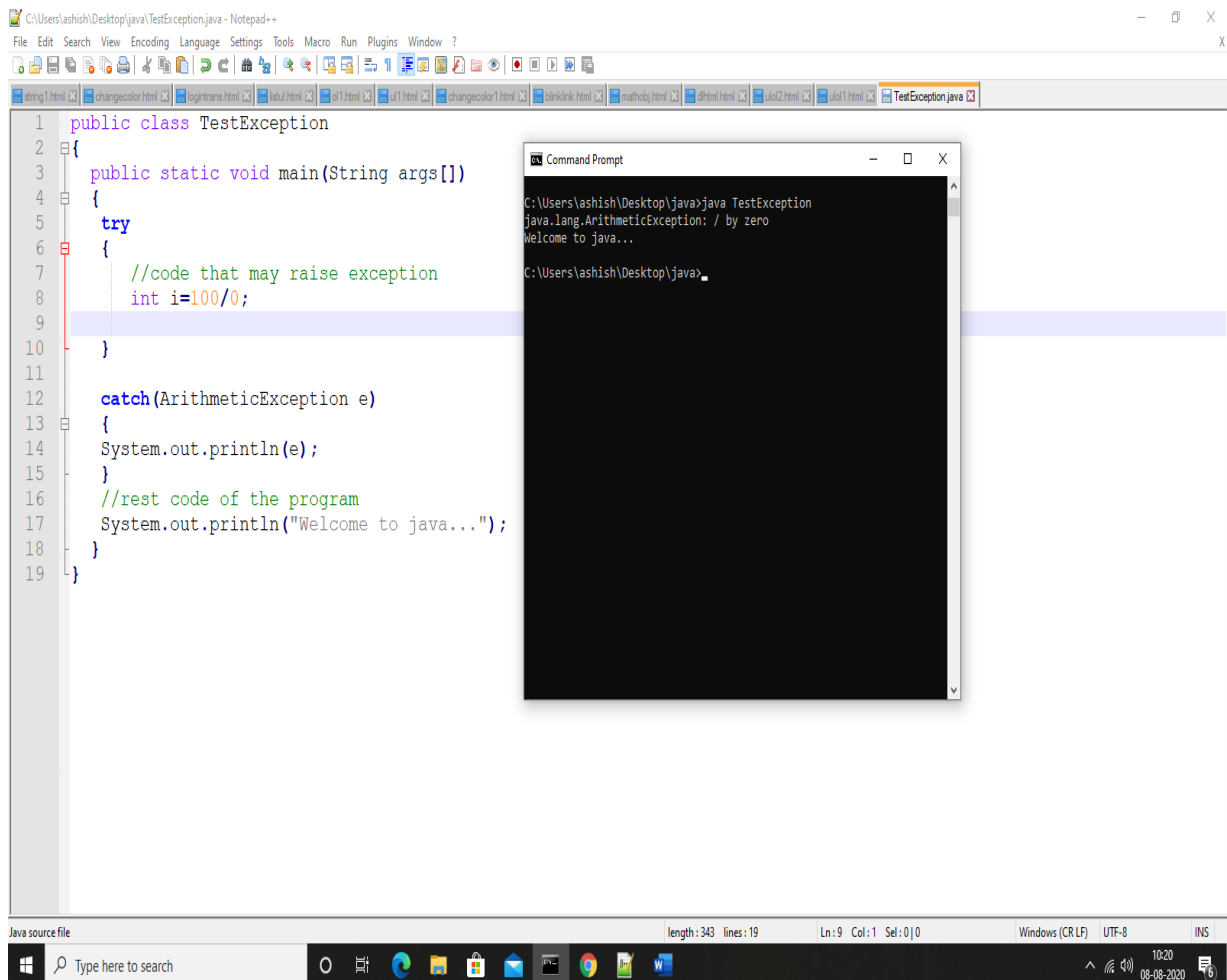
Error: Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Java Exception Keywords:

There are 5 keywords which are used in handling exceptions in Java:

- a. try:** The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
- b. 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.
- c. finally:** The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
- d. throw:** The "throw" keyword is used to throw an exception.
- e. throws:** The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Example:



The screenshot displays a Notepad++ window with a Java source file named `TestException.java`. The code is as follows:

```
1 public class TestException
2 {
3     public static void main(String args[])
4     {
5         try
6         {
7             //code that may raise exception
8             int i=100/0;
9         }
10    }
11
12    catch(ArithmeticException e)
13    {
14        System.out.println(e);
15    }
16    //rest code of the program
17    System.out.println("Welcome to java...");
18 }
19 }
```

Overlaid on the Notepad++ window is a Command Prompt window. It shows the command `C:\Users\ashish\Desktop\java>java TestException` being executed. The output of the program is displayed in the Command Prompt:

```
C:\Users\ashish\Desktop\java>java TestException
java.lang.ArithmeticException: / by zero
Welcome to java...
C:\Users\ashish\Desktop\java>
```

The status bar at the bottom of the Notepad++ window indicates the file is a Java source file, has a length of 343, and contains 19 lines. The cursor is at line 9, column 1. The Windows taskbar at the bottom shows the date and time as 10:20 on 08-08-2020.