Sukkur Institute of Business Administration University

Department of Computer Science

Object Oriented Programming using Java

BS – II (CS/AI/SE)

Spring-2024

Lab # 12: Let's learn about Exception handling

Instructor: Nimra Mughal

Objectives

After performing this lab, students will be able to understand:

- Interfaces
- Abstract classes
- Exception Handling

What is an exception in Java?

https://www.geeksforgeeks.org/exceptions-in-java/

https://www.javatpoint.com/exception-handling-in-java

https://www.javatpoint.com/finally-block-in-exception-handling

https://www.w3schools.com/java/java_try_catch.asp

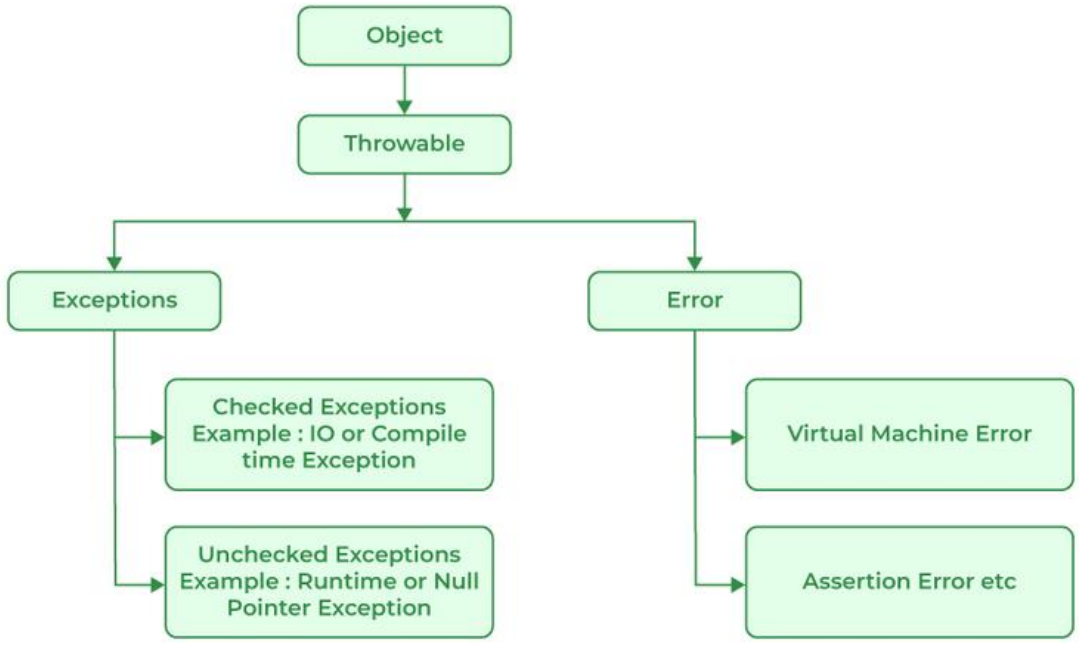
https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

Exception is an abnormal condition. In Java, 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, ArithematicException, ArrayIndexOutofBoundException, IOException, etc.

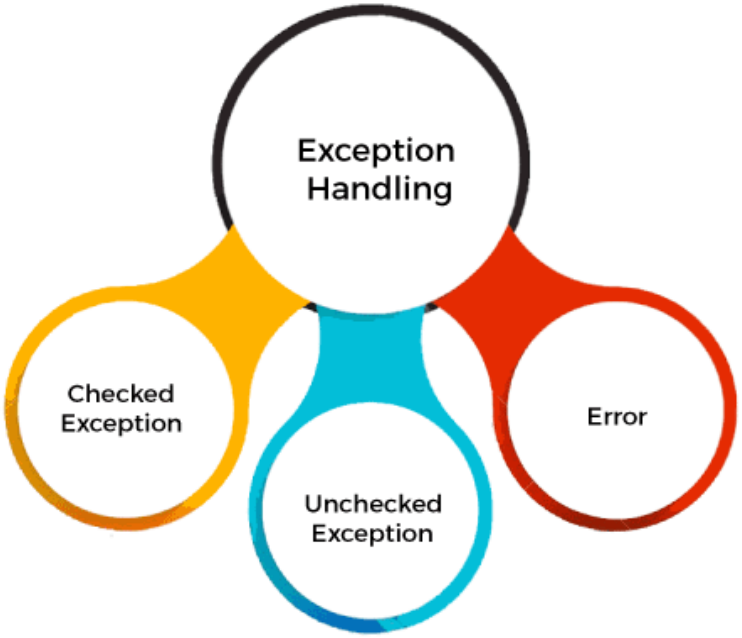
Hierarchy of Java Exception classes

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 **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.



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.

Java Exception Keywords

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

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

Java Exception Handling Examples

Please visit eLearning and execute the programs provided to you.

```
1. public class JavaExceptionExample{
2.   public static void main(String args[]){
3.    try{
4.       //code that may raise exception
5.       int data=100/0;
6.    }
7. catch(ArithmeticException e){
8.       System.out.println(e);
9. }
```

```
10.   //rest code of the program
11.   System.out.println("rest of the code...");
12.  }
13. }
```

Exercises

Exercise 1

Write a program that calculates the average of N integers. The program should prompt the user to enter the value for N and then afterward must enter all N numbers. If the user enters a no positive value for N, then an exception should be thrown (and caught) with the message " N must be positive." If there is any exception as the user is entering the N numbers, an error message should be displayed, and the user prompted to enter the number again.

Exercise 2

Here is a snippet of code that inputs two integers and divides them:

```
Scanner scan = new Scanner(System.in);

int n1, n2;

double r;

n1 = scan.nextInt();

n2 = scan.nextInt();

r = ( double) n1 / n2;
```

Place this code into a try-catch block with multiple catches so that different error messages are printed if we attempt to divide by zero or if the user enters textual data instead of integers (`java.util.InputMismatchException`). If either of these conditions occurs, then the program should loop back and let the user enter new data.

Exercise 3

Modify the previous exercise so that the snippet of code is placed inside a method. The method should be named ReturnRatio, read the input from the keyboard, and throw different exceptions if there is a division by zero or an input mismatch between text and an integer. Create your own exception class for the case of division by zero. Invoke ReturnRatio from your main method and catch the exceptions in main. The main method should invoke the ReturnRatio method again if any exception occurs.

Exercise 4

Write a program that can serve as a simple calculator. This calculator keeps track of a single number (of type double) that is called result and that starts out as 0.0 . Each cycle allows the user to repeatedly add, subtract, multiply, or divide by a second number. The result of one of these operations becomes the new value of result . The calculation ends when the user enters the letter R for "result" (either in upper- or lowercase). The user is allowed to do another calculation from the beginning as often as desired. The input format is shown in the following sample dialogue. If the user enters any operator symbol other than + , −, * , or / , then an UnknownOperatorException is thrown and the user is asked to reenter that line of input. Defining the class UnknownOperatorException is part of this project.

Sample output:

Calculator is on.

```
result = 0.0

+5

result + 5.0 = 5.0

new result = 5.0

* 2.2

result * 2.2 = 11.0

updated result = 11.0

% 10

% is an unknown operation.

Reenter, your last line:

* 0.1

result * 0.1 = 1.1

updated result = 1.1

r

Final result = 1.1

Again? (y/n)

yes

result = 0.0
```

```
+10

result + 10.0 = 10.0

new result = 10.0

/2

result / 2.0 = 5.0

r

Final result = 5.0

Again? (y/n)

N
```

End of Program