**EXCEPTION HANDLING**

An Exception can be anything which interrupts the normal flow of the program. When an exception occurs program processing gets terminated and doesn’t continue further. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled. We will cover the handling part later in this same tutorial.

**When an exception can occur?**

Exception can occur at runtime (known as runtime exceptions) as well as at compile-time (known Compile-time exceptions).

**Reasons for Exceptions**

There can be several reasons for an exception. For example, following situations can cause an exception – Opening a non-existing file, Network connection problem, Operands being manipulated are out of prescribed ranges, class file missing which was supposed to be loaded and so on.

**Difference between error and exception**

Errors indicate serious problems and abnormal conditions that most applications should not try to handle. Error defines problems that are not expected to be caught under normal circumstances by our program. For example memory error, hardware error, JVM error etc.

Exceptions are conditions within the code. A developer can handle such conditions and take necessary corrective actions. Few examples –

DivideByZero exception

NullPointerException

ArithmeticException

ArrayIndexOutOfBoundsException

Advantages of Exception Handling

Exception handling allows us to control the normal flow of the program by using exception handling in program.

It throws an exception whenever a calling method encounters an error providing that the calling method takes care of that error.

It also gives us the scope of organizing and differentiating between different error types using a separate block of codes. This is done with the help of try-catch blocks.

Why to handle exception?

If an exception is raised, which has not been handled by programmer then program execution can get terminated and system prints a non user friendly error message.

**Ex:-**Take a look at the below system generated exception

An exception generated by the system is given below

 Exception in thread "main" java.lang.ArithmeticException: / by zero at ExceptionDemo.main(ExceptionDemo.java:5)

 ExceptionDemo : The class name

 main : The method name

 ExceptionDemo.java : The filename

 java:5 : Line number

For a novice user the above message won’t be easy to understand. In order to let them know that what went wrong we use exception handling in java program. We handle such conditions and then prints a user friendly warning message to user, which lets them correct the error as most of the time exception occurs due to bad data provided by user.

**Types of exceptions**

There are two types of exceptions

1)Checked exceptions

2)Unchecked exceptions

Below is a brief about each however if you want a detailed tutorial with examples then you can refer Checked and Unchecked exceptions in Java.

**Checked exceptions**

All exceptions other than Runtime Exceptions are known as Checked exceptions as the compiler checks them during compilation to see whether the programmer has handled them or not. If these exceptions are not handled/declared in the program, it will give compilation error.

**Examples of Checked Exceptions :-**

ClassNotFoundException

IllegalAccessException

NoSuchFieldException

EOFException etc.

**Unchecked Exceptions**

Runtime Exceptions are also known as Unchecked Exceptions as the compiler do not check whether the programmer has handled them or not but it’s the duty of the programmer to handle these exceptions and provide a safe exit.

These exceptions need not be included in any method’s throws list because compiler does not check to see if a method handles or throws these exceptions.

**Examples of Unchecked Exceptions:-**

ArithmeticException

ArrayIndexOutOfBoundsException

NullPointerException

NegativeArraySizeException etc.

Exception hierarchy

Exception classes Hierarchy