EXCEPTIONS IN JAVA PROGRAMMING LANGUAGE

Introduction

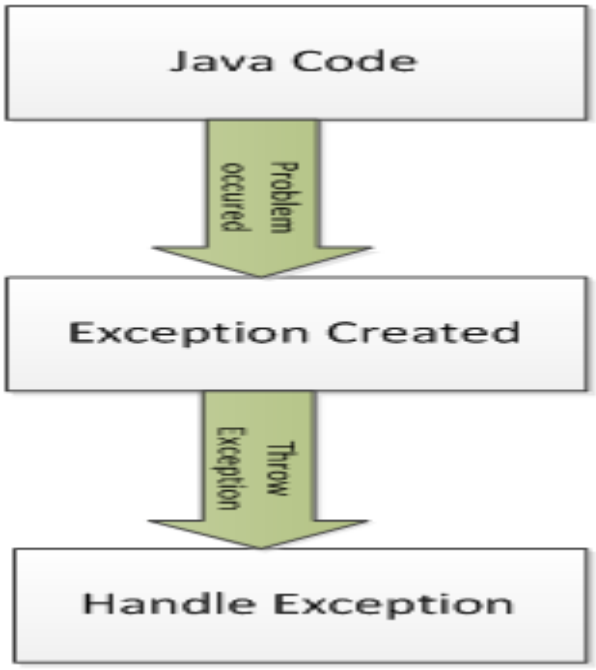
A Java exception is an object that describes an exceptional (that is, error) condition that has occurred in a piece of code. When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error. That method may choose to handle the exception itself, or pass it on. Either way, at some point, the exception is caught and processed.

The programs you write can generate many types of potential exceptions, such as when you do the following:

In Java there are three types of loops:

- You issue a command to read a file from a disk, but the file does not exist there.
- You attempt to write data to a disk, but the disk is full or unformatted.
- Your program asks for user input, but the user enters invalid data.
- The program attempts to divide a value by 0, access an array with a subscript that is too large, or calculate a value that is too large for the answer's variable type.

These errors are called exceptions because, presumably, they are not usual occurrences; they are “exceptional.” The object-oriented techniques to manage such errors comprise the group of methods known as exception handling.



Exception handling works by transferring the execution of a program to an appropriate exception handler when an exception occurs.

Handling Exceptions:

There are two ways for handling exception, first catch the exception and take corrective action or throws exception to the calling method which will forces the calling method to handle it.

When a Java method is going to throw an exception, to indicate that as part of the method signature ‘throws‘ keyword should be used followed by the exception.

It means that the caller of this method should handle the exception given in the throws clause. There can be multiple exceptions declared to be thrown by a method. If the caller of that method does not handles the exception, then it propagates to one level higher in the method call stack to the previous caller and similarly till it reaches base of the method call stack which will be the java’s runtime system. For this approach we use throws keyword in method declaration which will instructs compiler to handle exception using try-catch block. When we add throws keyword in divide method declaration compile time error.

Nested Try-catch block:

The try statement can be nested. That is, a try statement can be inside the block of another try.Each time a try statement is entered, the context of that exception is pushed on the stack. If an inner try statement does not have a catch handler for a particular exception, the stack is unwound and the next try statement’s catch handlers are inspected for a match. This continues until one of the catch statements succeeds, or until all of the nested try statements are exhausted.If no catch statement matches, then the Java runtime system will handle the exception.

Use of finally block:

When you have actions you must perform at the end of a try...catch sequence, you can use a finally block. The code within a finally block executes regardless of whether the preceding try block identifies an Exception. Usually, you use a finally block to perform cleanup tasks that must happen whether or not any Exceptions occurred, and whether or not any Exceptions that occurred were caught. In application where database connection, files are being operated, we need to take of closing those resources in exceptional condition as well as normal condition.

Summary:

We have learned about how exceptions are generated and various ways of handling exceptions. Catching exception or propagating exceptions. We have learned keywords like try, catch, finally, throws and programmatic use of these keywords.

Source: http://www.w3resource.com/java-tutorial/exception-in-java.php