Exception Interview Questions

1) What is an Exception?

An *exception* is an abnormal condition that arises in a code sequence at run time. In other words, an exception is a run-time error.

2) What is a Java Exception?

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

3) What are the different ways to generate and Exception?

There are two different ways to generate an Exception.

1. Exceptions can be generated by the Java run-time system.
   Exceptions thrown by Java relate to fundamental errors that violate the rules of the Java language or the constraints of the Java execution environment.
2. Exceptions can be manually generated by your code.
   Manually generated exceptions are typically used to report some error condition to the caller of a method.

4) Where does Exception stand in the Java tree hierarchy?

- java.lang.**Object**
- java.lang.**Throwable**
- java.lang.**Exception**
- java.lang.**Error**

5) Is it compulsory to use the finally block?

It is always a good practice to use the finally block. The reason for using the finally block is, any unreleased resources can be released and the memory can be freed. For example while closing a connection object an exception has occurred. In finally block we can close that object. Coming to the question, you can omit the finally block when there is a catch block associated with that try block. A try block should have at least a catch or a finally block.

6) How are try, catch and finally block organized?

A try block should associate with at least a catch or a finally block. The sequence of try, catch and finally matters a lot. If you modify the order of these then the code won't compile. Adding to this there can be multiple catch blocks associated with a try block. The

final concept is there should be a single try, multiple catch blocks and a single finally block in a try-catch-finally block.

7) What is a throw in an Exception block?

"throw" is used to manually throw an exception (object) of type **Throwable** class or a subclass of **Throwable**. Simple types, such as **int** or **char**, as well as non-**Throwable** classes, such as **String** and **Object**, cannot be used as exceptions. The flow of execution stops immediately after the **throw** statement; any subsequent statements are not executed.

```
throw ThrowableInstance;

ThrowableInstance must be an object of type Throwable or a subclass of Throwable.

throw new NullPointerException("thrownException");
```

8) What is the use of throws keyword?

If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception. You do this by including a **throws** clause in the method's declaration. A **throws** clause lists the types of exceptions that a method might throw.

```
type method-name(parameter-list) throws exception-list
{
   // body of method
}
Warning: main(http://www.javabeat.net/javabeat/templates/faqs/faqs_middle.html):
failed to open stream: HTTP request failed! HTTP/1.1 404 Not Found in
/home/content/k/k/s/kkskrishna/html/faqs/exception/exception-faqs-1.html on line 1

Warning: main(http://www.javabeat.net/javabeat/templates/faqs/faqs_middle.html):
failed to open stream: HTTP request failed! HTTP/1.1 404 Not Found in
/home/content/k/k/s/kkskrishna/html/faqs/exception/exception-faqs-1.html on line 1

Warning: main(): Failed opening 'http://www.javabeat.net/javabeat/templates/faqs/f
(include_path='.:/usr/local/lib/php') in /home/content/k/k/s/kkskrishna/html/faqs/
 on line 195


Here, exception-list is a comma-separated list of the exceptions that a method can


static void throwOne() throws IllegalAccessException {

System.out.println("Inside throwOne.");
```

9) What are Checked Exceptions and Unchecked Exceptions?

The types of exceptions that need not be included in a methods **throws** list are called **Unchecked Exceptions**.

- ArithmeticException
- ArrayIndexOutOfBoundsException
- ClassCastException
- IndexOutOfBoundsException
- IllegalStateException
- NullPointerException
- SecurityException

The types of exceptions that must be included in a methods **throws** list if that method can generate one of these exceptions and does not handle it itself are called **Checked Exceptions**.

- ClassNotFoundException
- CloneNotSupportedException
- IllegalAccessException
- InstantiationException
- InterruptedException
- NoSuchFieldException
- NoSuchMethodException

10) What are Chained Exceptions?

The chained exception feature allows you to associate another exception with an exception. This second exception describes the cause of the first exception. Lets take a simple example. You are trying to read a number from the disk and using it to divide a number. Think the method throws an **ArithmeticException** because of an attempt to divide by zero (number we got). However, the problem was that an I/O error occurred, which caused the divisor to be set improperly (set to zero). Although the method must certainly throw an **ArithmeticException**, since that is the error that occurred, you might also want to let the calling code know that the underlying cause was an I/O error. This is the place where chained exceptions come in to picture.

```
Throwable getCause( )

Throwable initCause(Throwable causeExc)
```