8. Error Handling

Lesson 4:
Throwing exceptions

8.6. Throwing Exceptions

- Before you can catch an exception, some code somewhere must throw one. Any code can throw an exception: your code, code from a package written by someone else such as the packages that come with the Java platform, or the Java runtime environment. Regardless of what throws the exception, it's always thrown with the throw statement.

- You can also create your own exception classes to represent problems that can occur within the classes you write. In fact, if you are a package developer, you might have to create your own set of exception classes to allow users to differentiate an error that can occur in your package from errors that occur in the Java platform or other packages.

8.6. Throwing Exceptions
8.6.1. Throw Statement

- All methods use the throw statement to throw an exception.

- The throw statement requires a single argument: a throwable object. Throwable objects are instances of any subclass of the Throwable class.

- Here's an example of a throw statement.

```
throws someThrowableObject;
```

8.6. Throwing Exceptions

8.6.2. Throwable Class and its Subclasses

- Class Throwable has two direct descendants: Error and Exception.

- Error Class
  - When a dynamic linking failure or other hard failure in the Java virtual machine occurs, the virtual machine throws an Error. Simple programs typically do not catch or throw Errors.

- Exception Class
  - Most programs throw and catch objects that derive from the Exception class. An Exception indicates that a problem occurred, but it is not a serious system problem. Most programs you write will throw and catch Exceptions as opposed to Errors.

8.6. Throwing Exceptions

8.6.3. Chained Exceptions

- An application often responds to an exception by throwing another exception.

- In effect, the first exception causes the second exception. It can be very helpful to know when one exception causes another. Chained Exceptions help the programmer do this.

```
try
{
   …
}
catch (IOException e)
{
   throws new SampleException("Other IOException", e);
}
```

8.7. Built-in Exceptions in Java
8.7.1. Runtime Exceptions

ArithmeticException
Arithmetic error, such as divide by zero.

ArrayIndexOutOfBoundsException
Array index is out of bounds.

ArrayStoreException
Assignment to an array element of an in compatible type

ClassCastException
Invalid cast

IllegalArgumentException
Illegal argument used to invoke a method

IllegalMonitorStateException
Illegal monitor operation, such as waiting on an unlocked thread.

IllegalStateException
Environment or application is in incorrect state

IllegalThreadStateException
Requested operation not compatible with current thread state

IndexOutOfBoundsException
Some type of Index is out of bounds

NegativeArraySizeExeption
Array created with the negative size

NullpointerException
Invalid use of null exception

NumberFormatException
Invalid conversion of a string to a numeric format

SecurityException
Attempt to violate security

StringIndexOutOfBounds
Attempt to index outside the bounds of a string

UnsupportedOperationException
An unsupported operation was encountered

8.7. Built-in Exceptions in Java

8.7.2. Checked Exceptions

ClassnotFoundException	Class not found
CloneNotSupportedException	Attempt to clone an object that does not implement the cloneable interface
IllegalAccessException	Access to a class is denied
InstantiationException	Attempt to create an Object of an abstract class or interface
InterruptedException	One thread has been interrupted by another thread
NoSuchFieldException	A requested field does not exist
NoSuchMethodException	A requested method does not exist

8.8. Creating your own Exception Classes

- You can create your own exception classes by extending any exception class.

```
class MyNewException extends Exception
{
        ...
}
```