Intro. to Exceptions
Ch 8.3
Highlights

- try, throw and catch
There are two major parts to think about when programming:

1. What should the normal execution do?
2. How do I treat unusual events?

We have done mostly part 1, but sometimes we did some of part 2... For example:

```
if(ref == null)
if(array.length == 0)
```
You have no doubt encountered runtime errors during your coding experiences.

Normally when a runtime error happens, the angry red text appears and stops your program.

Your program can actually continue running if you use `try` and `catch` block.
A try block simply has the word try before it

```java
try
{
    array[222222222]=2;
    System.out.println("I have a long array");
}
```

If a runtime error (or a specific error) happens inside this try block, you will be thrown to the catch block
try, throw catch

A `catch` block is similar to a method that takes one argument

```java
catch(Exception e)
{
    System.out.println("My array is not that long");
}
```

If the error from the `try` block matches the argument to `catch`, then that catch block is run

(See: RuntimeErrors and TryThrowCatch.java)
Manual exceptions

You can define your own exceptions if you do not want to use an existing one:

```java
if(i >= names.length)
{
    throw new Exception("Your number is too large!");
}
```

The above is using `throw` with Exception constructor (takes one String argument)

(See: ManualException.java)
Manual exceptions

You can define a class for an exception:

```java
public class TooLarge extends Exception{
    public TooLarge() {
        super("Too large");
    }
}
```

You can then catch this specific error by making the argument to `catch` type TooLarge

(This will no longer catch other errors)

(See: TooLarge and TooLargeTest.java)
Exception class

Just as Object is the parent to all classes, Exception is the parent to all exceptions.

This means if you write: `catch(Exception e)`
then it will catch all runtime exceptions.

If the exception in your `catch` does not match the exception that happened in `try`, then your code will have a normal runtime error.
Multiple catch

You can use multiple catch statements for a single try block

Only the first catch statement which satisfies the exception will run

(If you put catch(Exception e) as first catch, then you will never get to later ones)

(See: MultiCatch.java)