Testing for Exceptions

This lesson explains how to test some code for exception types.

WE'LL COVER THE FOLLOWING

`std.exception` module

It is common to test some code for exception types that it should or should not throw under certain conditions. The `std.exception` module contains two functions that help with testing for exceptions:

- `assertThrown`: ensures that a specific exception type is thrown from an expression
- `assertNotThrown`: ensures that a specific exception type is not thrown from an expression

For example, a function that requires that both of its slice parameters have equal lengths and works with empty slices can be tested as in the following tests:

```
import std.exception;
int[] average(int[] a, int[] b) {
    // ...
}
unittest {
    /* Must throw for uneven slices */
    assertThrown(average([1], [1, 2]));
    /* Must not throw for empty slices */
    assertNotThrown(average([], []));
}
```

Normally, `assertThrown` ensures that some type of exception is thrown

without regard to the actual type of that exception. When needed, it can test against a specific exception type as well. Likewise, `assertNotThrown` ensures that no exception is thrown whatsoever, but it can be instructed to test that a specific exception type is not thrown. The specific exception types are specified as template parameters to these functions:

```
/* Must throw UnequalLengths for uneven slices */
assertThrown!UnequalLengths(average([1], [1, 2]));
/* Must not throw RangeError for empty slices (it may
 * throw other types of exceptions) */
assertNotThrown!RangeError(average([], []));
```

The main purpose of these functions is to make code more succinct and more readable. For example, the following `assertThrown` line is the equivalent of the lengthy code below it:

```
assertThrown(average([1], [1, 2]));
// ...
    /* The equivalent of the line above */
    {
        auto isThrown = false;

        try {
            average([1], [1, 2]);
        } catch (Exception exc) {
            isThrown = true;
        }

        assert(isThrown);
    }
```

In the next lesson, we will explore test-driven development for unit testing.