Zack Campbell
Zcc254

Lab 10: Testing

Exercise 1:

I added a test that asserts the “isItSunny()” method to be false, which returns fails because it is set to return true. In the failure message it describes the expected and actual values and the line that caused the failure to occur.

I added the following test using the API:

```
@Test
public void testIsSunnyEqualsTrue() {
   assertEquals(Philadelphia.isItSunny(), Philadelphia.isItSunny());
}
```

This test passed because it is comparing if two booleans are equal, in this case, true vs. true which passes the test.

A test that throws and exception:

```
@Test
public void testThrowException() throws Exception {
   throw new Exception();
}
```

This test throws a new exception when the test is run.

Exercise 2:

```
@Test
public void testArrayEmpty() {
   testArray.clear();
   assertTrue(testArray.isEmpty());
}

@Test
public void testArrayContains() {
   assertTrue(testArray.contains(4));
}

@Test
public void testArrayNotContains() {
   assertFalse(testArray.contains(2));
}

@Test
public void testArrayGet() {
   assertEquals(3, (int)testArray.get(0));
}
```

Exercise 3:

```
@Test
public void testFullStatementCoverage() {
   assertEquals(43200, TimeParser.parseTimeToSeconds("0:00:00 pm"));
   assertEquals(0, TimeParser.parseTimeToSeconds("0:00:00 am"));
}

@Test
public void testFullBranchCoverage() {
   assertEquals(82800, TimeParser.parseTimeToSeconds("11:00:00 pm"));
   assertEquals(0, TimeParser.parseTimeToSeconds("0:00:00 cm"));
}

@Test
public void testFirstException() {
   assertThrows(NumberFormatException.class, () -> TimeParser.parseTimeToSeconds("0
pm"));          // Test first exception that there isn't a colon
}

@Test
public void testSecondException() {
   assertThrows(NumberFormatException.class, () -> TimeParser.parseTimeToSeconds("0:00
pm"));          // Test the second exception that there isn't a second colon
}

@Test
public void testThirdException() {
   assertThrows(IllegalArgumentException.class, () ->
TimeParser.parseTimeToSeconds("0:00:60 am"));   // Test the third exception that there
isn't a valid time
}
```

These tests provide full statement coverage, full branch coverage and as much path coverage as possible. Some paths are not possible to achieve, such as failing the first if statement and passing the second one.

Exercise 4:

```
@Test
public void testInvariant() {
   fillWithRandomValues(5);
   assertTrue(invariantHolds());
}
```