461L: Tutorial 10: Testing
Nicholas White: NWW295

Exercise 1:

Original Test Results: 1 test passed
---
Added test case results: 1 test passed, 1 test failed
---
Static assertion of equals method w/ boolean variable set to true
Results: 2 tests passed, 1 test failed
---
New test throws exception
Results: 3 tests passed, 1 test failed
---

Exercise 2:

Original test fixture: verified
---
```
//Tests the clear method and verify the array is empty
   @Test
   public void testClear() {
      testArray.clear();
      assertNull(testArray);
   }
```
---
```
//Tests the contains method and verifies an element in the array
   @Test
   public void testContains() {
      assertTrue(testArray.contains(1));
   }
```
---
```
//Tests the contains method by verifying that returns false for an element that does not exist in the array
   @Test
   public void testContainsFalse() {
      assertFalse(testArray.contains(54));
   }
```
---
```
//Tests the get method and verifies true
   @Test
   public void testGet() {
      int currentNum = 1;
      int correctIndex = 1;
```

```
        assertEquals((int)testArray.get(correctIndex), currentNum);
    }
```

Exercise 3:

Write a minimal test suite that provides full statement coverage:

```
//Minimal full statement coverage
    @Test
    public void parseTimeToSeconds() throws Exception {


        assertEquals(TimeParser.parseTimeToSeconds("12:00:00 am"), 0);

    }
```

---

Write a minimal test suite for branch coverage:

```
//Full branch coverage testing
    @Test(expected = NumberFormatException.class)
    public void parseNegativeCoverage() {

        int newTime = TimeParser.parseTimeToSeconds("21");
    }

    @Test(expected = IllegalArgumentException.class)
    public void parseMinutesFormat() {
        TimeParser.parseTimeToSeconds("12:2112");
    }

    @Test(expected = NumberFormatException.class)
    public void parseMissingColon() {
        TimeParser.parseTimeToSeconds("12:21");
    }
```

---

Write a minimal test suite for path coverage

```
//Full path coverage
    //Illegal arguments and branches cover previously -- focus on paths for legal args
    @Test
    public void testPM() {
        assertEquals(TimeParser.parseTimeToSeconds("1:00:00 pm"), 46800);
    }
```

---

Exercise 4:

Write a test suite that tests the following addition class invariant:

"For all elements in the array, array[n] <= array[2*n] and array[n] <= array[2*n+1]"

```
//Test for property array[n]<=array[2*n]
   @Test
   public void test2N() {
      boolean checkState;
      fillWithRandomValues(VALUES_TO_TEST);
      for(int i = 0; i < VALUES_TO_TEST/2; i++) {
         int initial = heap.get(i);
         int j = 2*i;
         int val2N = heap.get(j);

         if(initial > val2N) {

            checkState = false;
            assertFalse(checkState);
         }

      }
      checkState = true;
      assertTrue(checkState);

   }

   //Test for property array[n] = array[2*n +1]
   @Test
   public void testCase2() {
      boolean checkState;
      fillWithRandomValues(VALUES_TO_TEST);
      for(int i = 0; i < VALUES_TO_TEST/2; i++) {
         int initial = heap.get(i);
         int j = (2*i) + 1;
         int val = heap.get(j);

         if(initial > val) {
            checkState = false;
            assertFalse(checkState);
         }
      }

      checkState = true;
      assertTrue(checkState);
   }
```