Jack Diao
qd572

Exercise 1

1. Same output verified
2. The failed test method shows an X mark instead of a check mark, and an AssertionFailed Error is shown in the Failure Trace.
3.
```
@Test
public void testSevenEqualsEight() {
        Integer expectedNum1 = 7;
        Integer actualNum1 = 8;
        assertEquals(expectedNum1, actualNum1);
}
```
an AssertionFailed Error is shown in the Failure Trace
4.
```
@Test
public void testDivideByZero() {
          Philadelphia philly = new Philadelphia();
          assertThrows(ArithmeticException.class, () -> philly.divide(1, 0), "throws DivideByZero exception");
}
```
The test is passed. However, if the some other exception, such as NullPointerException is used, an AssertionFailed Error would be shown in the Failure Trace.

Exercise 2

1. Same output verified
2.
```
@Test
public void testClear() {
        testArray.clear();
        assertTrue(testArray.isEmpty());
}
```
3.
```
@Test
public void testContainTrue() {
        assertTrue(testArray.contains(3));
}
```
4.
```
@Test
public void testContainFalse() {
        assertFalse(testArray.contains(7));
}
```
5.
```
@Test
public void testGet() {
        assertEquals(5, testArray.get(4));
}
```

Exercise 3

1. Full statement coverage

```
@Test
public void testFirstColonMissing() { // true
        assertThrows(NumberFormatException.class,
                        () -> TimeParser.parseTimeToSeconds("12345 pm"),
                        "Not detecting unrecognized time format");
}

@Test
public void testSecondColonMissing() { // false, true
        assertThrows(NumberFormatException.class,
                        () -> TimeParser.parseTimeToSeconds("12:345 pm"),
                        "Not detecting unrecognized time format");
}

@Test
public void testIllegalArgumentPm() { // false, false, if, true
        assertThrows(IllegalArgumentException.class,
                        () -> TimeParser.parseTimeToSeconds("15:34:05 pm"),
                        "Not detecting unacceptable time specified");
}

@Test
public void test12Am() { // false, false, else if, false
        assertEquals(320, TimeParser.parseTimeToSeconds("12:05:20 am"));
}
```

2. Full branch coverage

   All of the full statement coverage test methods and the following:

```
@Test
public void test12Pm() { // false, false, else, false
        assertEquals(43520, TimeParser.parseTimeToSeconds("12:05:20 pm"));
}
```

3. Full path coverage

   All of the full statement coverage and full branch coverage test methods and the following:

```
@Test
public void test6Pm() { // false, false, if, false
        assertEquals(65120, TimeParser.parseTimeToSeconds("6:05:20 pm"));
}
```

```
@Test
public void testIllegalArgument12Am() { // false, false, else if, true
        assertThrows(IllegalArgumentException.class,
                        () -> TimeParser.parseTimeToSeconds("12:34:60 am"),
                        "Not detecting unacceptable time specified");
}

@Test
public void testIllegalArgument12Pm() { // false, false, else, true
        assertThrows(IllegalArgumentException.class,
                        () -> TimeParser.parseTimeToSeconds("12:64:30 pm"),
                        "Not detecting unacceptable time specified");
}
```

Exercise 4

1. Changed the invariantHolds() to the following, and everything else is the same as in the `MinHeapArrayInvariant1Test` class. Running the tests didn't find any errors in the `HeapArray` class.

```
  private boolean invariantHolds() {
     Integer top = heap.peek();
     if (top == null) {
        return true;
     }

     Integer[] contents = new Integer[heap.size()];
     Arrays.asList(heap.toArray(contents));
     for (int i = 0; i < heap.size()/2; i++) {
          if (contents[i] > contents[2*i+1]) {
                System.out.println("Whoops!");
                return false;
          }
          else if (2*i+2 < heap.size()) {
                if (contents[i] > contents[2*i+2]) {
                     System.out.println("Whoops!");
                     return false;
                }
          }
     }

     return true;
  }
```