CS 4501/6501: Quiz 3

12-Sep-2017

Names:

Instruction: Answer the questions as concisely as you can. Please write neatly; if I can't read it I have to mark it wrong.
Consider the following JUnit Test Class, which (unusually) includes the method under test:

```
public class MiniMinTest {

   int[] testValues;

   //  @return min element in values
   //  @throws IllegalArgumentException if values is empty
   //  @throws NullPointerException if values is null

   public static int min( int[] values ) {
      if (values.length == 0) throw new IllegalArgumentException();  // Question 1
      int result = values[0];
      for (int i = 1; i < values.length; i++) {
         if (values[i] < result) {                // Question 2
            result = values[i];
         }
      }
      return result;
   }

   @Test public void test1() {
      try {
         int result = min(new int[0]);
      } catch (Exception e) {
          return;
      }
      fail("Testing empty array");
   }

   @Test public void test2() {
      testValues = new int[2];
      testValues[0] = 2;
      testValues[1] = 1;
      int result = min(testValues);
      assertTrue("Double element array test", result == 1);
   }
}
```

1. If the line of code labeled "Question 1" is deleted, the behavior of `min()` changes.
   - Explain how
     Answer:
     If the line of code is deleted, `min()` throws a different exception, namely `IndexOutOfBoundsException` instead of `IllegalArgumentException`.
   - test1 does not detect this change. Fix test1.
     Answer:
     As written, test1 passes if *any* exception is thrown. test1 should be:

     ```
     ...
           } catch (IllegalArgumentException e) {
     ...
     ```

     Grading note: Rewriting test1 to use the `@Test(expected==IllegalArgumentException.class)` format is also a fine answer.

2. Consider the line of code labeled "Question 2". Suppose it were changed to be:

```
if (values[i] != result) {                // != instead of <
```

   - test2 still passes. Why?
     Answer: Because != and < return the same truth value on the comparison between the two values.

   - Write a (simple) variant of test2 that detects this fault. You may write a new test case or edit test2.
     Answer:

     ```
     @Test public void test2() {
        testValues = new int[2];
        testValues[0] = 1;        // simply switch the values.
        testValues[1] = 2;
        int result = min(testValues);
        assertTrue("Single element array test", result == 1);
     }
     ```

     Grading note: Adding a test case (instead of editing test2) is also fine.

3. Write a test that checks for `NullPointerException` being thrown under the correct conditions. Don't worry too much about JUnit/Java syntax.
   Answer:

   ```
   @Test(expected = NullPointerException.class)
   public void test3() {
      int result = min(null);
   }
   ```