

1. Utility classes

Find all of the errors in the following utility class; note that the class does compile:

```
import java.util.Date;

public class DateUtil {

    /**
     * Today's date.
     */
    public static final Date TODAY = new Date();

    /**
     * Returns the year of today's date.
     *
     * @return the year of today's date
     */
    public int getYear() {
        // Date is weird; it represents the year as the number of
        // years after 1900
        return DateUtil.TODAY.getYear() + 1900;
    }
}
```

2. JUnit 1

Consider the following method from `java.lang.Math`:

```
public class Math {

    /**
     * Returns the smaller of two int values.
     *
     * @param a
     *         an argument
     * @param b
     *         another argument
     * @return
     *         the smaller of a and b.
     */
    public static int min(int a, int b) { // implementation not shown }
}
```

Complete the following JUnit test by choosing an appropriate set of test values and the expected return value for the `min` method.

```
public class MathTest {  
  
    @Test  
    public void test_min() {  
  
        assertEquals(exp, actual);  
    }  
  
}
```

3. Test cases

Recall that a test case is a specific set of arguments to pass to a method, and the expected return value (if any) and the expected results when the method is called with the specified arguments.

Provide two test cases for the following method:

```
/**  
 * Given a list containing exactly 2 integers, negates the values of  
 * the integers in the list.  
 *  
 * @param t  
 *         a list containing exactly 2 integers  
 * @throws IllegalArgumentException  
 *         if the list does not contain exactly 2 integers  
 */  
public static void negate2(List<Integer> t) { // implementation not shown }
```

One test case should test that an exception is thrown, and the other test case should test that the values in the list are negated.

4. Test cases 2

Consider the method `isInside(double x, double y)` from Lab 1:

```
/**
 * Determine if the point (x, y) is strictly inside the circle with center
 * (0, 0) and having radius equal to 1. A point on the perimeter of
 * the circle is considered outside of the circle.
 *
 * @param x the x-coordinate of the point
 * @param y the y-coordinate of the point
 * @return true if (x, y) is inside the unit circle, and false otherwise
 */
public static boolean isInside(double x, double y)
```

Provide 5 test cases for the method `isInside(double x, double y)`. Make sure to provide test cases for typical argument values that produce return values of true and false, and test cases that test boundary cases.