Khiem Lu
3/12/18

Homework Assignment #3

7.1
Most of these comments are obvious and unnecessary, so we can take them out. For this code, we should be commenting why we do what we do rather than just explaining what each line does. This can be found in the following link provided on Euclid's algorithm: en.wikipedia.org/wiki/Euclidean_algorithm

```
// Use Euclid's algorithm to calculate the GCD.
        private long GCD( long a, long b )
        {
            a = Math.abs( a );
            b = Math.abs( b );
            for( ; ; )
            {
                // Set remainder to the remainder of a / b
                long remainder = a % b;
                If( remainder == 0 ) return b;
                a = b;
                b = remainder;
            };
        }
```

7.2

1) Comments can be repetitive and over explained when trying to describe every detail. This can usually happen during the top-down design.
2) Comments can also be bad when the programmer writes comments after each line of code. This prevents the comments from being insightful and explaining why and how it fits into the rest of the code. It usually leads to comments simply explaining what each code does, which can be obvious.

7.4
Exercise 3 validation code already seems offensive because it validates inputs and results.

7.5

Error handling is not necessary because the code throws exceptions that are passed to the calling code already.

7.7

1) Go to car
2) Start car
3) Exit parking structure
4) Turn right out of driveway
5) Follow the street to stop sign
6) Turn right and follow the road down 3 stop lights
7) Turn right into supermarket plaza
8) Park in empty spot
9) Exit car
10) Enter supermarket

Assumptions:

- Car works
- You drive safely
- Street signs are functional
- There are parking spots
- Supermarket is open

8.1

Pseudocode

```
boolean relativelyPrime (int x, int y) {
        x = absolute value (x)
        y = absolute value (y)

        if ( x == 1) or (y == 1) return true
        if(x == 0) or ( y == 0) return false

        int min = find smaller value between x and y
        for (int i = 2; i <= min; i ++){
                if (a % i == 0) and (b % i == 0) return false
        }
        else return true;
}
```

8.3
Testing techniques for relativelyPrime method would be a black-box test because it doesn't show how it works. An exhaustive text wouldn't work well because there would be way too many values to test out.

8.5
One problem I noticed was that some integer values would not work, specifically the min and max values. Writing tests helped me notice this issue and also raised awareness to other possible problems

8.9
Exhaustive tests can be black-box tests because they don't need to know the details of the method being tested.

8.11
Using Lincoln indexes technique to calculate for each pair of testers

- Alice & Bob = 10
- Bob & Carmen = 20
- Alice & Carmen = 12.5
- Total estimate = 14

8.12
If the two testers don't find any bugs in common then the Lincoln estimate doesn't show how many bugs are left. Instead, you can get a lower bound by assuming they had 1 bug in common.