How To Comment

Did I dock off points on your commenting? Womp womp.

I'm messing around, but I do want to address this, because I think it's really important for you guys to understand what I'm looking for as far as comments.

Why Comment?

Imagine working on a group project, and you push your code into the main branch on Github, or show your group-mates your code. Your code makes sense to you, but what about to other people, especially if your approach is something non-trivial?

This is why we comment. Unfortunately, writing a complicated series of for loops won't click in everyones mind as far as what you are doing. But thankfully, the English dictionary exists, and we can use that to help us explain our code.

Okay jokes aside, comments just make everyones life easier, even yourself. If there's a bug somewhere in your implemented sort, you can go back to that sorting algorithm, and re-read your comments to see if your high level approach was wrong.

The Basics

Here's some general guidelines that I use when commenting.

1. Single Line vs Multi-line

Usually, 90% of the time you want to use single line comments. Multi-line comments honestly just look janky, 9 times out of 10 you don't need that extra consecutive line to describe what you are doing. Only time you should be using Multi-line comments is when you are describing what a particular method does above the method signature, *or* when you just typed a crazy complex algorithm, and you need a couple of lines to explain the crazy optimization you made.

```
/*
MULTILINE COMMENT HERE
*/
public void wabledeedabledee() {
        // Single Lines in here
        System.out.println("foo");
        /*
        If you trained a whole neural network in this method
        then you have permission to use multiline comments.
        */
}
```

Simple Algorithm -> Single Line Comment
Complex Algorithm -> Single Line/Multiline comment
Voodoo Bananas Algorithm -> Multiline comment

2. Too Much vs Too Little Commenting

Okay this ones a tough one, since it's so broad. But let me break it down for you.

Here's a rule of thumb: if the algorithm is trivial for a given problem, comment briefly. If the algorithm is some witchcraft, comment a little more.

Here's a great example:

```

/*
Method prints all prime numbers from low to high, inclusive.
*/
public static void findAllPrimes(int low, int high) {
        // iterate through the range of numbers.
        for(int i = low; i <= high; i++) {
                boolean isPrime = true;
                // check if each number from 2 to i is divisible.
                for(int n = 2; n < i && isPrime; n++) {
                        if(i % n == 0) {
                                isPrime = false;
                        }
                }
                // Only print when isPrime stays true.
                if(isPrime) {
                        System.out.print(i + " ");
                }
        }

}
```

As you can see, this is a pretty naive approach to the problem, therefore I shouldn't comment as much; everything looks trivial to the eye of another programmer. But what about this:

```
public static void findAllPrimes(int low, int high) {
        boolean[] isPrime = new boolean[high + 1];
        Arrays.fill(isPrime, true);
        isPrime[0] = false; // base case #1, 0 is not prime.
        isPrime[1] = false; // base case #2, 1 is also not prime.

        // Iterate from 2 up to high, when i*i > high i*i would be non-prime.
        for (int i = 2; i * i <= high; i++) {
                // For each prime number i, all multiples of i are non-prime.
                if (isPrime[i]) {
                        // // Start marking from i * i (lower multiples will
already be marked)
                        for (int j = i * i; j <= high; j += i) {
                                // mark every multiple of i as non-prime.
                                isPrime[j] = false;
                        }
                }
        }

        // Adjust the range if low is less than 2
        if (low < 2) {
                low = 2;
        }

        // Print prime numbers in the range [low, high]
        for (int i = low; i <= high; i++) {
                if (isPrime[i]) {
                        System.out.println(i);
                }
        }
}
```

By the way, this method runs in $\mathbb{O}(\log n \cdot \log n)$ time. My codespace isn't FILLED with comments, but my comments are more elaborate than the first example, since I'm doing something that you wouldn't normally think of naively. This kind of commenting is plenty (if you want you can use a multiline comment as to why we check for i * i < high).

3. Method Explanations

Speaking of methods, we should probably touch on how to comment method descriptions. Honestly, here's what I'm looking for:

1. **What does the method do?**
2. **What are the parameters?**
3. **What are the preconditions?**

Satisfy these three requirements, you're set.
