Magee - OOP - Java - Sample 1

Mr. D Bylsma
Chris
Adrian

You are given a starter file. It is well documented; we highly recommend you look through it, and try to write a prime-number-checker on your own.

Assignment/Tasks:

1. Change the main method so that in the first step it reads commands (see sample project 1)
2. If the user entered "exit", stop the program
3. If the user entered "test", prompt them for a number and do the test
4. If the user entered "range", prompt them for a lower and upper bound, and print out the "prime test" (n is prime, n is not prime) for all the numbers in the range (hint: for loop! Dont' worry about whether you should include the start/stop number)

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.adrian;

import com.creatifcubed.simpleapi.Console;

/**
 *
 * @author raekye
 */
public class DriverDelta {

    public static void main(String[] args) {
        // possible infinite loop
        while (true) {
            int n = Console.readInt("What number do you want to test? 0 to quit: ");
            // we will "break" out of the loop if the user enters 0.
            // So there won't be an infinite loop
            if (n == 0) {
                break;
            }
            // isPrimeNumber returns a boolean, which we can use directly in the
            // if statement
            if (isPrimeNumber(n)) {
                Console.println(n + " is prime");
            } else {
                Console.println(n + " is not prime");
            }
        }
        Console.println("Done");
    }
```

```
    public static boolean isPrimeNumber(int n) {
        /*
         * A number is prime if it is divisble by 1 or itself
         * We will test all the numbers between 2 and n - 1 (inclusive)
         * If any of them divide evenly into n, n is not a prime
         * IE: n should have no factors other than 1 or itself
         */
        // This loop starts at 2, and finishes at n - 1
        // We don't test 1 or n because n can always be divided by 1 and itself
        for (int i = 2; i < n; i++) {
            // % is the remainder operator. A number divides evenly into another
            // if the remainder is 0
            if (n % i == 0) {
                // divided evenly, n has a factor
                Console.println(n + " can be divided by " + i);
                // at this point, we know n is not prime, so we can stop
                // when Java hits the return keyword, it stops the function
                return false;
            }
        }
        // If we reached here, it means we have tested all the numbers
        // from 2 to (n - 1)
        // And none of them divided evenly (if one did, the function would have
        // returned and exited). So n must be prime
        return true;
    }
}
```