Homework Assignment 05

Purposes:

- accumulator patten.
- selection
- for-loop and while loop

Prime Numbers

We say a number is prime if the only divisor it can have is the number 1 and itself. For example 3 has divisor 1 and 3 only, so it is prime. You can see that prime numbers are not very "divisible". So they are view as the atoms of numbers. Prime number is a great topic that attracts both mathematician and computer scientist.

Testing whether a number is prime is not a hard task. Testing it **fast** is one. One successful story of how undergraduate student can contribute to this very important field and making great breakthrough can be found here.

Today we will just do it in a straight forward way.

In this assignment, the main task we need to is print out a list of prime numbers. You will need to do

1. Write a class named **PrintPrimes** that looks like. You will get the instructions for the functions later.

```
class PrintPrimes{
    public static void main(String[] args)
    {
        //write your main() here.
        //1.read in an integer represent the maximum to check
        //2.call printPrimes() with the maxium number obtain from 1.
    }
    public static void printPrimes(int max)
    {
        //write your printPrimes here.
        //for all number in the range of (1, max). (Yeah, use a for-loop!)
        // if it is prime, print it out. Note that you need to call isPrime() here.
    }
    public static boolean isPrime(int n)
    {
        //write your isPrime() here.
    }
}
```

2. Implement the **boolean** function **isPrime()** to determine whether a number **n** is a prime number.

   The header of the function will look as in the above code.

   - Let the input be **n**, then in order to see if **n** is prime, you just need to check if there is any divisor of **n** in the range of 1 to the square root of **n**. Why? And how to do square root in Java? How to

check whether a number is a divisor of another number?
- Use a for-loop to implement the above check.
- Use a flag (boolean) variable **result** to indicate the result.

Pseudo-code, or the ideas of implementation will look like

```
boolean result= true;
for any number m greater than 2 and less than square root of n:
    if m is a divisor of n,
        then result= false
return your result
```

The above way to implement the algorithm follows a so-called "accumulator patten". To simply put it, there are three steps as you can see in the above.

- Initialize a variable that indicates the result.
- Each in the loop-body, we update the result, if necessary.
- Return the result, or use it for other purpose.

The patten can be seen a lot in accumulating an arithmetic result (we will discuss this in the lab). So it gets the name.

3. Implement the function **printPrime**.

   For any number greater than 2 and less than the maximum number the user input, if it is prime, output it.

Submission

You need to submit your work (the .java file) through Canvas, due Monday, Feb 24.