

Create one directory called `P5_Looping` to store the `.java` files. Please format the source code using appropriate indentation and all common Java conventions.

The purpose of this project is to practice the material from Savitch sections 4.1 - 4.2.

Create a class `public class Utility`, and another class called `public class Program`. Inside the `Program` class, create the `public static void main(String[] args)` method.

You will start by creating various methods, which you should put in the `Utility` class. Then write code in the `Program` class to call and test your methods from the `Utility` class.

**Definition 1.** If  $a$  and  $b$  are integers, we say that  $a$  *divides*  $b$ , and write  $a \mid b$ , if there exists an integer  $k$  such that  $b = ak$ .

If  $a \mid b$ , we say that  $a$  is a *factor* of  $b$ , and that  $b$  is a *multiple* of  $a$ .

**Program 1.** Create a method `public static void mult3or7()` that prints all positive multiples of 3 or 7 which are less than 50, using a while loop.

**Program 2.** Create a method `public static void mult5or9()` that prints all positive multiples of 5 or 9 which are less than 100, using a for loop.

**Program 3.** Create a method `public static void squares(int n)` which prints the first  $n$  perfect squares, starting with 1.

**Program 4.** Create a program `public static void primeFactors(int n)` which produces and prints all prime factors of  $n$ .

Hint: If you factor out primes as you find them, you do not have to remember or even test if a number is prime. Here is some pseudocode for this:

```
initialize k
while n > 0
    increment k
    while k divides n
        print k
        divide n by k
```

The *Fibonacci sequence* a recursively defined sequence  $(F_n)_{n=1}^{\infty}$ , given by setting  $F_1 = 1$ ,  $F_2 = 1$ , and  $F_{k+2} = F_k + F_{k+1}$ . The numbers  $F_n$  are called *Fibonacci numbers*.

The first few Fibonacci numbers are 1, 1, 2, 3, 5, 8, 13, 21,  $\dots$ . To get the next term in the sequence, add the two previous terms.

**Program 5.** Create a method `public static void fibonacci(int n)` which prints the first  $n$  Fibonacci numbers.

A *Pythagorean triple* is a tuple  $(a, b, c)$  of positive integers such that  $a^2 + b^2 = c^2$ . The Babylonians produced tablets containing tables of Pythagorean triples. It is conjectured that they may have known of the formula to generate such triples: let  $u$  and  $v$  be any positive integers with  $u > v$ , and set

- $a = u^2 - v^2$ ;
- $b = 2uv$ ;
- $c = u^2 + v^2$ .

Then

$$\begin{aligned}
 a^2 + b^2 &= (u^2 - v^2)^2 + (2uv)^2 \\
 &= u^4 - 2u^2v^2 + v^4 + 4u^2v^2 \\
 &= u^4 + 2u^2v^2 + v^4 \\
 &= (u^2 + v^2)^2 \\
 &= c^2.
 \end{aligned}$$

This proves the first part of the following proposition.

**Proposition 1.** *Let  $u, v$  be positive integers with  $u > v$  and set  $a = u^2 - v^2$ ,  $b = 2uv$ , and  $c = u^2 + v^2$ . Then  $(a, b, c)$  is a Pythagorean triple. Moreover, all Pythagorean triple may be generated in this way.*

These triples are naturally ordered, in increasing order of there  $u$ 's, then their  $v$ 's.

**Program 6.** Create a method `public static void pythagoreanTriples(int n)` which produces and prints the first  $n$  Pythagorean triples (that is, starting with the smallest  $u$ ).

If  $a$  and  $b$  are positive integers, let  $\text{gcd}(a, b)$  denote the greatest common divisor of  $a$  and  $b$ . A Pythagorean triple  $(a, b, c)$  is *primitive* if  $\text{gcd}(a, b) = 1$ .

**Program 7.** Add the following method to the Utility class.

```

public static int gcd(int n, int m)
{
    int r = 0;
    while (r = n % m)
    {
        n = m;
        m = r;
    }
    return m;
}

```

Write code in the Program class to test this.

Create a program `public static void primitiveTriples(int n)` which produces and prints the first  $n$  primitive Pythagorean triples.