CMT205 Object Oriented Development with Java

Lab Exercises Week 3

Section 1: Methods

1. Create a Java class **Fraction** to represent a fractional number. Two instance variables **nominator** and **denominator** correspond to the nominator and denominator of the fraction, i.e. the represented number is **nominator / denominator**. Methods are included to support common operations between fractional numbers, including **add** (addition), **subtract** (subtraction), **multiple** (multiplication) and **divide**. The method **output()** is used to print out the fraction, in the form '**nominator/denominator**'. A skeleton is given as follows:

   Note: You may prefer that the fraction is most simplified. For example, 1/3 + 1/6 = 1/2, however, it is acceptable if your output is 3/6 or 9/18 etc.

```
public class Fraction
{
    public Fraction( int nominatorValue, int denominatorValue )
    {
      . . .
    }
    public Fraction add (Fraction another)
    {
      . . .
    }
    public Fraction subtract (Fraction another)
    {
      . . .
    }
    public Fraction multiple (Fraction another)
    {
      . . .
    }
    public Fraction divide (Fraction another)
    {
     . . .
    }
    public void output ( )
    {
     . . .
    }
    private int nominator;
    private int denominator;
}
```

FractionTest.java is provided on learning central for your testing. Put it in the same folder as your program (or include it in your Eclipse project).

Section 2: Arrays

1. Write two static methods getMax() and getMin() to find the maximum and minimum numbers given an `int` array as the parameter. Write a `main` method to test these two methods using an arbitrary array of `int`.

2. A prime number is a natural number that is only divisible by two distinct natural numbers, 1 and itself. A traditional method to find all the prime numbers less than or equal to a given integer ***n*** is known as Sieve of Eratosthenes, which works as follows:

    - Create a list of consecutive integers from 2 to ***n***. For example if ***n***=20:
      2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
    - Leave the first number in the list (2 in this case), and cross every 2nd number in the list after the number:-
      2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~, 11, ~~12~~, 13, ~~14~~, 15, ~~16~~, 17, ~~18~~, 19, ~~20~~
    - Leave the next number not yet crossed in the list (3 in this case), and cross every 3rd number in the list (even if they have been crossed out already):-
      2, **3**, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~
    - Leave the next number not yet crossed in the list (5 in this case), and cross every 5th number in the list (even if they have been crossed out already):-
      2, **3**, ~~4~~, **5**, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~
    - Repeat this process by leaving the next number not yet crossed and cross out multiples of this number in the list until all the numbers are processed.
    - The remaining list of numbers contains prime numbers no larger than ***n***: 2, 3, 5, 7, 11, 13, 17, 19.

See more detailed explanation from
http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes.

Given a number ***n***, print all the prime numbers no larger than ***n***. The number ***n*** can be specified at the beginning of the program (hard-coded), or use the command line arguments etc.