University of Colorado at Colorado Springs

Home Work Assignment 5
Out 11/4/2019, Due 11/18/2019

1 Polygons (35 pts code + 5 pts pseudocode = 40 pts)

Write a program `Polygons.java` that computes the area of different types of shapes. The program can calculate the area for a pentagon, hexagon or regular polygon using these formulas:

$$Pentagon\ Area = \frac{5s^2}{4\tan(\pi/5)}, \quad \text{and} \quad Hexagon\ Area = \frac{6s^2}{4\tan(\pi/6)}, \tag{1}$$

$$Regular\ Polygon\ Area = \frac{ns^2}{4\tan(\pi/n)}, \tag{2}$$

where s is the length of a side (all sides the same), and n is the number of sides. A pentagon or hexagon's area can be calcualted as a special case when a regular polygon has 5 or 6 sides. In this problem, s and n are user input.

- The program can assume input n is always an integer, but must verify that $n \geq 5$. s can be either an integer or a double, but the program must verify that $s > 0$, i.e., 0 not included. If the input is invalid, the program will exit using `System.exit(1)`.
- When n is 5, the program must print "The **pentagon's** area is ..." When n is 6, the program must print "The **hexagon's** area is ..." Otherwise, it must print "The regular n-gon's area is ..." See examples on the next page.
- Finally, display the result with **two decimal points**.

Design your program using the following methods:

```
/** Return true if user input is valid */
public static boolean isValid(double s, int n)

/** Calculate the area */
public static double computeArea(double s, int n)
```

Here are some examples of how the program should work.

Case #1: Invalid user input.

```
Please enter s (the length of a side) and n (the number of sides): -5 6
Sorry, your input is not valid. Bye.
```

Case #2: Valid user input (note the program must print 8-gon).

```
Please enter s (the length of a side) and n (the number of sides): 5 8
The regular 8-gon's area is 120.71.
```

Case #3: Valid user input.

```
Please enter s (the length of a side) and n (the number of sides): 3.5 6
The hexagon's area is 31.83.
```

2 Prime Numbers (45 pts code + 5 pts pseudocode = 50 pts)

The program you are going to write in this problem, called `Prime.java`, deals with prime numbers. There are two parts to the program.

- The program will prompt the user to input a positive, non-zero integer. The user input can be assumed to be integers. But if the value is invalid, the program will exit using `System.exit(1)` from the `main` method. If the value is valid, the program will then determine if this integer is prime and report accordingly.
- If this integer is not prime, the program will determine the nearest integer that is smaller than the user input, and is prime. It will report what this nearest prime is. Note that if the user input is smaller than 2, we cannot find a nearest prime.

Please do not use arrays to implement this. Design your program using the following methods:

```
/** Return true if user input is valid */
public static boolean isValid (int number)

/** Return true if number is prime */
public static boolean isPrime (int number)

/** Get the nearest prime that is smaller than number */
public static int getNearestPrime(int number)
```

Please refer to Midterm Question 7 regarding how to determine if an integer is prime. Here are some examples of how the program should work.

Case #1: Invalid user input.

```
Please enter a positive, non-zero integer: -5
Sorry, your input is not valid. Bye.
```

Case #2: Valid user input, and is prime.

```
Please enter a positive, non-zero integer: 13
13 is prime.
```

Case #3: Valid user input, not prime, and can find the nearest prime.

```
Please enter a positive, non-zero integer: 25
25 is not prime.
The nearest prime is 23.
```

Case #4: Valid user input, not prime, but cannot find a nearest prime.

```
Please enter a positive, non-zero integer: 1
1 is not prime.
We cannot find the nearest prime.
```

Submission

Please save your programs in two Java files, each containing **pseudocode**. You may include your pseudocode in a block comment using `/* ... */`. **10 pts are given to your coding style** (comments – header and in-code comments: up to 4 pts, naming conventions: up to 3 pts, proper indentation/spacing: up to 3 pts). We will run each program several times with our input and verify that the results are correct.

Please place your files in a folder called **hw5-firstname-lastname** and zip it. The zipped file should be named **hw5-firstname-lastname.zip**. Please submit the zipped file to Canvas by the due date.