CSCI 241 Assignment 6 — Prime Palindromes

Due: Thursday, April 1, 2021 *at 11:59 p.m.*

Note: This project may not seem too difficult at first, but some parts are challenging. Don't leave it until the last minute!

For 3 points, email your instructor the lines of code for the **`getCount()`** method (in the email message) by 11:59 p.m. on Tuesday, March 30.

Objective:

Write a program that uses separate class-level methods to print a table of prime integers that are also palindromes.

Description:

Begin by creating a new BlueJ project named **`Assign6Primes`**. Add a class named **`PrimePalindromes`** (your class and project names MUST match these EXACTLY).

First, a quick reminder:
A *prime number* is a positive number which is evenly divisible by only the number itself and 1.
A *palindrome* is a number, word or phrase that reads the same way in reverse.

By the time your program has been completed and is running correctly, you will see printed output that looks like this (data entered from the keyboard is **`bold and underlined`**):

```
Welcome! This program will find a chosen number of
prime numbers that are also palindromes.
How many numbers would you like to see? 50
      2       3       5       7      11     101     131     151     181     191
    313     353     373     383     727     757     787     797     919     929
 10,301  10,501  10,601  11,311  11,411  12,421  12,721  12,821  13,331  13,831
 13,931  14,341  14,741  15,451  15,551  16,061  16,361  16,561  16,661  17,471
 17,971  18,181  18,481  19,391  19,891  19,991  30,103  30,203  30,403  30,703
```

If you choose to run the program so that it determines 100 integers that are palindromes, you would see:

```
Welcome! This program will find a chosen number of
prime numbers that are also palindromes.
How many numbers would you like to see? 100
      2       3       5       7      11     101     131     151     181     191
    313     353     373     383     727     757     787     797     919     929
 10,301  10,501  10,601  11,311  11,411  12,421  12,721  12,821  13,331  13,831
 13,931  14,341  14,741  15,451  15,551  16,061  16,361  16,561  16,661  17,471
 17,971  18,181  18,481  19,391  19,891  19,991  30,103  30,203  30,403  30,703
 30,803  31,013  31,513  32,323  32,423  33,533  34,543  34,843  35,053  35,153
 35,353  35,753  36,263  36,563  37,273  37,573  38,083  38,183  38,783  39,293
 70,207  70,507  70,607  71,317  71,917  72,227  72,727  73,037  73,237  73,637
 74,047  74,747  75,557  76,367  76,667  77,377  77,477  77,977  78,487  78,787
 78,887  79,397  79,697  79,997  90,709  91,019  93,139  93,239  93,739  94,049
```

The list of 100 numbers took noticeably more time to run on my desktop computer, so don't be surprised if there is a delay in the output when you ask to print a large set of prime palindromes.

A task like this can be quite complex to do it all in a `main()` method. However, we can isolate the subtasks and write a separate method for each, making the overall task much more manageable.

Method Descriptions

You will write 6 static methods for this assignment, including your `main()` method. Any static method can call any other static method. `main()` will start things out, but it will call other methods, which in turn may call others. Remember, a method that calls another will simply be suspended until the method it calls has finished.

Each method should be `public` and `static`. Here are the method details:

- **`void main(String [] args)`**
  1. Prints a welcome message (see sample on previous page *–match it exactly!*)
  2. Calls the `getCount()` method (described next) to retrieve, from the user, how many integers they wish the program to find and print.
  3. Uses a loop (any kind) that will keep running until the program has calculated as many numbers as the user requested. Each time the loop runs:
     a. It uses an `if` statement to check if the integer is a prime and is a palindrome. (See the `isPrime()` and `isPalindrome()` methods described below.)
     b. If both send back a `true` return value, the program should call the `printNum()` method (described below) to print the number (and print a new line character if needed). It should also increment the number of palindromic prime integers found.
     c. If either one of the checks is false, the loop continues to the next value.

As you write your separate methods, place a comment heading describing each method before its code. Check the documentation standards on the grading criteria page in your Course Manual (and posted in Canvas).
Also, remember that you can test each static method separately in the BlueJ window. Do this BEFORE adding method calls from `main()` or other locations.

- **`int getCount()`**
  This method takes no parameters and returns an integer. It sets up a keyboard `Scanner` to read the number of integers requested from the keyboard (this is the ONLY place you need to declare a Scanner). It should use a loop to make sure that the number entered is a positive number and no more than 100 and keep asking if it is not. Once it has number that falls in this range, it returns that number to the calling method.

- **`void printNum(int, int)`**
  This method takes 2 integers as parameters and returns nothing.
  The *first parameter* is the number that you wish to print.
  The *second parameter* is a count of the number of qualifying integers found so far (for example, if this is the first prime palindrome found, this value will be 1, and if it is the 10th prime palindrome found, this value will be 10).

  This method will print the number from the first parameter using formatted output. The second parameter tells you if you need to begin a new line after the number is printed.

  In the sample output shown on the first page all numbers are printed with the same number of positions, and some of them have commas. At most, 10 numbers are printed per line.
  - Use 7 positions to print the number.
    - Hint: you can include a "comma grouping separator" for your numbers by including a comma after the % sign, as in `"%,"`
  - If the second parameter is evenly divisible by 10, make sure that your printing will begin on a new line next time.

- **`boolean isPrime (int)`**
  This method takes one integer as a parameter and returns a `boolean` indicating whether the integer is or is not a prime number.

  Remember that the number 2 is prime and you should return `true`.
  If the number is larger than 2, you will need to check if integers that come before it will divide into it evenly. You'll need a loop (a `for` loop would work well here) to run through all the possibilities that might divide evenly into the number. *Note: this loop can take a long time to run. What is the largest integer (compared to the original) that needs to be checked to see if it divides evenly into the parameter? (Hmmm… wasn't there a lab exercise which did such a thing?) Choosing the right answer here can cut your execution time in half!*

  If any smaller integer divides evenly into the parameter, you should return `false`.

- **`int findReverse(int)`**
  This method takes the original integer as a parameter and returns the integer with its digits reversed. It will be called by the `isPalindrome()` method described next.

  If your variable was a `String` variable, this would actually be quite easy – but it is not for integers. But, using divide and mod on an integer works very well.

  For example, if you have a 2-digit number:
  Number / 10 ← gives you the 10's digit
  Number % 10 ← gives you the 1's digit.

  So … how could you extend this for a 3-digit number, to get all of its individual digits?

- **`boolean isPalindrome(int)`**
  This method takes the original integer as a parameter and returns a `boolean` indicating if the integer matches its version in reverse.

  Since you already wrote the `findReverse()` method described above, you can call it to get the reverse version of your number. If, when you compare them, you find that they have the same value, this method should return `true`, otherwise `false`.

Submission Requirements

1. **Electronic:** Submit the completed implementation in the lab for the **`Assign6Primes`** project through BlueJ's submission process.
2. **Canvas Copy:** Upload/Submit a copy of the `PrimePalindromes.java` file to Canvas.

Grading Criteria

Your program must compile without errors to be accepted. **Note: This requirement will be strictly enforced on this and subsequent assignments!! Contact me if you need help with errors or need help submitting to the CS lab.**

All of the following criteria will be used to evaluate your program:

Email to your instructor by 11:59 p.m. on Tuesday, March 30	*3 pts*
Correct submission (CS lab, Canvas, correct class and project names)	2 pts
Appropriate and complete comments (including before each method)	6 pts
Adherence to style and design conventions (identifier names, indentation, etc.)	4 pts
Correct program behavior	15 pts
Total points:	**30 pts**