Writing Methods

ASSIGNMENT #2

It is time to get some practice writing methods

For this assignment you will write several methods that determine if a given number falls into a special category or not. The program you will write will then test all the numbers from 0 to 100 to determine if each number fits in each special category.

For example, 0 is an even number, not a prime number, while 1 is not an even number nor a prime number, and 2 is both an even number and a prime number.

The main method

The main method will be based upon the following pseudo-code solution:

```
1. Loop through the numbers from 0 to 100 using the var num, for each value of num:
    a. Output the value of num (with a text label)
    b. Output whether or not num is an even number
    c. Output whether or not num is a prime number
    d. Output whether or not num is a self-divisor
    e. Output whether or not num is a palindromic number
```

Sample Console output:

```
0 is: even, not a prime number, not a self-divisor, palindromic
1 is: odd, not a prime number, a self-divisor, palindromic
…
99 is: odd, not a prime number, a self-divisor, palindromic
100 is: even, not a prime number, not a self-divisor, not palindromic
```

In the `main()` method, you will need to call several methods, each returning `true` or `false` to determine if the current number is or is not in each category.

You will add methods to the `Numbers` class of your `myLib` package:

For example, your first method should be:

```
public static boolean isEven(int number) {
   return true;
}
```

The remaining methods are: `isPrime(int)`, `isSelfDivisor(int)`, and `isPalindromic(int)`

Initially, while you are still working on each of the methods, it will be helpful to have them return true, by default. This will allow you to finish writing your main method.

Your main method should then include a loop with an iterator. Inside the loop you will want output the current value of the iterator followed by " is a:"

Use a `System.out.print()` instead of `System.out.println()` so that you can continue sending more results to the same console line.

Next you will need to write a conditional statement that tests if the current value of your iterator is an even number using the `isEven()` method. Your "`if`" condition should output to the console the text "`even`" and your "`else`" condition "`odd`."

Repeat this process for each method.

You will need to include a `System.out.println();` to end the console output line near the end of your loop.

Don't forget to increment your iterator variable!

Writing the methods

Below you will find pseudo-code algorithms for each method you need to write.

`isEven(int)` Method:

1. If `number` modulus `2` is `0`, the number is even, return `true`.

`isPrime(int)` Method:

1. If `number` is less than `2`, it is not prime, return `false`.
2. Check each number from `2` to `number-1` to see if the modulus is `0`, if it is, return `false`.
3. If all numbers have been checked, then it is a prime number, return `true`.

`isSelfDivisor(int)` Method:

"A positive integer is called a self-divisor if every decimal digit of the number is a divisor of the number, that is, the number is evenly divisible by each and every one of its digits. For example, the number 128 is a self-divisor because it is evenly divisible by 1, 2, and 8. However, the number 26 is not a self-divisor because it is not evenly divisible by the digit 6. Note that 0 is not considered to be a divisor of any number, so any number containing a 0 digit is NOT a self-divisor."

1. If `number` is less than `1`, it is not a self-divisor, return `false`.
2. Store a copy of `number` in the variable `checkNum`.
3. Loop while `checkNum` is greater than `0`.
    a. Store the least significant digit of `checkNum` using modulus `10` in the variable `digit`.
    b. If `digit` is `0` or `number` modulus `digit` is not `0`, then `number` isn't evenly divisible by this digit so the number can't be a self-divisor, return `false`.
    c. Store the value of `checkNum` divided by `10` in the variable `checkNum` so that the next digit can be checked by the loop.
4. If the loop ends, then all the digits have been checked so this number is a self-divisor, return `true`.

`isPalindromic(int)` Method:

"A number is palindromic if it is the same written forward and backward."

The algorithm for determining if a number is Palindromic is similar to that of the Self-Divisor. Use modulus to look at each digit. Take each digit one at a time and construct a new number with the digits in reverse order. If the resulting number is identical to the original number, then it is Palindromic!