Counting and Summing Digits

The problem of counting the digits in a positive integer or summing those digits can be solved recursively. For example, to count the number of digits think as follows:

- If the integer is less than 10 there is only one digit (the base case).
- Otherwise, the number of digits is 1 (for the units digit) plus the number of digits in the rest of the integer (what's left after the units digit is taken off). For example, the number of digits in 3278 is 1 + the number of digits in 327.

The following is the recursive algorithm implemented in Java.

```
public int numDigits (int num)
{
    if (num < 10)
      return (1);   // a number < 10  has only one digit
    else
      return (1 + numDigits (num / 10));
}
```

Note that in the recursive step, the value returned is 1 (counts the units digit) + the result of the call to determine the number of digits in *num / 10*. Recall that *num/10* is the quotient when *num* is divided by 10 so it would be all the digits except the units digit.

The file *DigitPlay.java* contains the recursive method *numDigits* (note that the method is static—it must be since it is called by the static method main). Copy this file to your directory, compile it, and run it several times to see how it works. Modify the program as follows:

1. Add a static method named *sumDigits* that finds the *sum* of the digits in a positive integer. Also add code to main to test your method. The algorithm for *sumDigits* is very similar to *numDigits*; you only have to change two lines!

2. Most identification numbers, such as the ISBN number on books or the Universal Product Code (UPC) on grocery products or the identification number on a traveller's check, have at least one digit in the number that is a *check digit*. The check digit is used to detect errors in the number. The simplest check digit scheme is to add one digit to the identification number so that the sum of all the digits, including the check digit, is evenly divisible by some particular integer. For example, American Express Traveller's checks add a check digit so that the sum of the digits in the id number is evenly divisible by 9. United Parcel Service adds a check digit to its pick up numbers so that a weighted sum of the digits (some of the digits in the number are multiplied by numbers other than 1) is divisible by 7. Modify the main method that tests your sumDigits method to do the following: input an identification number (a positive integer), then determine if the sum of the digits in the identification number is divisible by 7 (use your sumDigits method but don't change it—the only changes should be in main). If the sum is not divisible by 7 print a message

indicating the id number is in error; otherwise print an ok message. (FYI: If the sum is divisible by 7, the identification number could still be incorrect. For example, two digits could be transposed.) Test your program on the following input:

- 3429072 --- error
- 1800237 --- ok
- 88231256 --- ok
- 3180012 --- error

```
// ********************************************************************
//   DigitPlay.java
//
//   Finds the number of digits in a positive integer.
// ********************************************************************

import java.util.Scanner;

public class DigitPlay
{

    public static void main (String[] args)
    {
      int num;    //a number

      Scanner scan = new Scanner(System.in);

      System.out.println ();
      System.out.print ("Please enter a positive integer: ");
      num = scan.nextInt ();

      if (num <= 0)
          System.out.println ( num + " isn't positive -- start over!!");
      else
          {
            // Call numDigits to find the number of digits in the number
            // Print the number returned from numDigits
            System.out.println ("\nThe number " + num + " contains " +
                                + numDigits(num) + " digits.");
            System.out.println ();
          }
```

```
    }


    // ------------------------------------------------------------
    //  Recursively counts the digits in a positive integer
    // ------------------------------------------------------------
    public static int numDigits(int num)
    {
      if (num < 10)
          return (1);
      else
          return (1 + numDigits(num/10));
    }
}
```