# Chapter 7 Arrays
# Programming Project

This assignment will give you practice with arrays. You are going to write a program that adds together large integers. The built-in type int has a maximum value of 2,147,483,647. Anything larger will cause what is known as overflow. Java also has a type called long that has a larger range, but even values of type long can be at most 9,223,372,036,854,775,807.

The approach you are to implement is to store each integer in an array of digits, with one digit per array element. We will be using arrays of length 25, so we will be able to store integers up to 25 digits long. We have to be careful in how we store these digits. Consider, for example, storing the numbers 38423 and 27. If we store these at the "front" of the array with the leading digit of each number in index 0 of the array, then when we go to add these numbers together, we're likely to add them like this:

```
38423
27
```

Thus, we would be adding 3 and 2 in the first column and 8 and 7 in the second column. Obviously this won't give the right answer. We know from elementary school arithmetic that we have to shift the second number to the right to make sure that we have the appropriate columns lined up:

```
38423
   27
```

To simulate this right-shifting of values, we will store each value as a sequence of exactly 25 digits, but we'll allow the number to have leading 0's. For example, the problem above is converted into:

```
0000000000000000000038423
0000000000000000000000027
```

Now the columns line up properly and we have plenty of space at the front in case we have even longer numbers to add to these.

The data for your program will be stored in an array of Strings called nums. Each String will have a different addition problem for you to solve. Each String will have one or more integers to be added together. Take a look at the array at the end of this write-up and the output you are supposed to produce. Notice that you produce a line of output for each array element showing the addition problem you are solving and its answer. Your output should also indicate at the end how many String elements were processed. You must exactly reproduce this output.

Notice that the array is of type String since some of the numbers will be larger than the standard int. Your first task, then, will be to convert a String of digits into an array of 25 digits. As described above, you'll want to shift the number to the right and include leading 0's in front. The String method charAt and the method Character.getNumericValue will be helpful for solving this part of the problem. (e.g. Character.getNumericValue(data.charAt(i)).

You are to add up each line of numbers, which means that you'll have to write some code that allows you to add together two of these numbers or to add one of them to another. This is something you learned in Elementary School to add starting from the right, keeping track of whether there is a digit to carry from one column to the next. Your challenge here is to take a process that you are familiar with and to write code that performs the corresponding task.

Your program also must write out these numbers. In doing so, it should not print any leading 0's. Even though it is convenient to store the number internally with leading 0's, a person reading your output would rather see these numbers without any leading 0's.

You can assume that the array Strings has numbers that have 25 or fewer digits and that the answer is always 25 digits or fewer. Notice, however, that you have to deal with the possibility that an individual number might be 0 or the answer might be 0. There will be no negative integers in the input file.

You should solve this problem using arrays that are exactly 25 digits long. Certain bugs can be solved by stretching the array to something like 26 digits, but it shouldn't be necessary to do that and you would lose style points if your arrays require more than 25 digits.

The choice of 25 for the number of digits is arbitrary (a magic number), so you should introduce a class constant that you use throughout that would make it easy to modify your code to operate with a different number of digits.

Consider the String array as an example of the kind of problems your program must solve. I might use a more complex String array for actual grading.

The Java class libraries include classes called BigInteger and BigDecimal that use a strategy similar to what I am asking you to implement in this program. You are not allowed to solve this problem using BigInteger or BigDecimal. You must solve it using arrays of digits.

You may assume that the String array has no errors. In particular, you may assume that each String begins with at least one number and that each number and each answer will be 25 digits or fewer. You may also assume that there is exactly one space after each number.

You will again be expected to use good style throughout your program and to comment each method and the class itself. A major portion of the style points will be awarded based on how you break this program down into static methods. Think in terms of logical subtasks of the overall task and create different methods for different subtasks. You should have at least four static methods other than main and you are welcome to introduce more than four if you find it helpful.

Your program should be stored in a file called LastnameSum.java.

The main method with the String array nums is shown below.

```
public static void main(String[] args) {
   String nums[] = {
      "82384 ",
      "204 435 ",
      "22 31 12 ",
      "999 483 ",
      "28350 28345 39823 95689 234856 3482 55328 934803 ",
      "7849323789 22398496 8940 32489 859320 ",
      "729348690234239 542890432323 534322343298 ",
      "394869234869234869348623 583493834923485623486342 3 ",
      "999999999999999999999999 432432 58903 34 ",
      "82934 49802390432 8554389 4789432789 0 48372934287 ",
      "0 ",
      "0 0 0 ",
      "7482343 0 4879023 0 8943242 ",
      "3333333333 4723 3333333333 6642 3333333333 ",
```

```
        };
      processStrings(nums);
   }
```

## Output that should be produced

```
82384 = 82384
204 + 435 = 639
22 + 31 + 12 = 65
999 + 483 = 1482
28350 + 28345 + 39823 + 95689 + 234856 + 3482 + 55328 + 934803 = 1420676
7849323789 + 22398496 + 8940 + 32489 + 859320 = 7872623034
729348690234239 + 542890432323 + 534322343298 = 730425903009860
3948692348692348693486235 + 5834938349234856234863423 =
97836306979272049283496 58
99999999999999999999999 + 432432 + 58903 + 34 = 100000000000000000000491368
82934 + 49802390432 + 8554389 + 4789432789 + 0 + 48372934287 = 102973394831
0 = 0
0 + 0 + 0 = 0
7482343 + 0 + 4879023 + 0 + 8943242 = 21304608
3333333333 + 4723 + 3333333333 + 6642 + 3333333333 = 10000011364

Total lines = 14
```