Methods with a variable number of parameters

The last parameter declaration of a method can look as shown (in red) in the following function.

```
(0)     /** return the value of sum plus all the numbers in nums. */
        public static int addTo(int sum, int… nums) {
            for (int k= 0; k < nums.length; k= k+1)  sum= sum + nums[k];
            return sum;
        }
```

The declaration of parameter nums as int… nums means that a corresponding argument list can have 0, 1, 2, or more int expressions, and they will be processed as an int array, as the code in the method body shows. In fact, one can also have a single int array as the argument. Here are some calls on addTo and what they produce:

(1)	addTo(5, 4, 2)	returns the value 11	
(2)	addTo(5, new int[]{4, 2})	returns the value 11	
(3)	addTo(5, 4)	returns the value 9	(array nums contains 1 element)
(4)	addTo(5)	returns the value 5	(array nums contains 0 elements)
(5)	addTo(5, 2, 2, 2, 2, 2)	returns the value 15	

The parameter declaration int… nums just provides more syntactic sugar, allowing a call to have:

a list of arguments, like 4, 2, instead of requiring the creation of a new array, like new int[]{4, 2}.

Here are some points to consider.

1. The use of int… nums in a declaration only provides syntactic sugar and nothing new. When a program is compiled, this will be converted to a declaration int[] nums and the corresponding arguments in a call will be converted into a new-expression —e.g. call (1) will be converted to (2).
2. As a verification of point (1), you can write method main like this!

   ```
   public static void main(String… args) { … }
   ```

3. Instead of type int in int… nums, you can put any type, e.g. String… s.
4. This new notation can be used only in the *last* parameter of a method declaration.
5. In method (0) above, it would have been better to use the following foreach loop instead of the for-loop. We used the for-loop to stress that nums was of type int[].

   ```
   for (int v : nums) sum= sum + v;
   ```

6. Each converted call of method addTo will have a new-expression that creates a new array, even if it has 0, 1, or 2 elements. This can be costly. If calls usually have 1 or 2 elements for nums, you can make it more efficient by overloading addTo with the following methods. Don't waste your time doing this unless it really matters.

   ```
   /** return the value of sum + num1. */
   public static int addTo(int sum, int num1) {
       return sum + num1;
   }

   /** return the value of sum + num1 + num2. */
   public static int addTo(int sum, int num1, int num2) {
       return sum + num1 + num2;
   }
   ```