Variable-Length Arguments

A veces querrá crear un método que tome un número variable de argumentos, en función de su uso preciso.

Varargs Basics

Un argumento de longitud variable se especifica mediante tres puntos (...).

```
static void vTest1(int... v) {
      System.out.println("Number of args: " + v.length);
      System.out.println("Contents: ");
      for (int i = 0; i < v.length; i++) {
         System.out.println(" arg " + i + ": " + v[i]);
      }
      System.out.println();
   }
```

Ejemplos de llamada:

```
vTest1(1,2,3,4);
vTest1();
```

Un método puede tener parámetros "normales" junto con un parámetro de longitud variable. Sin embargo, el parámetro de longitud variable debe ser el último parámetro declarado por el método. Por ejemplo, esta declaración de método es perfectamente aceptable:

```
static void vTest2(String msg, int... v) {
      System.out.println(msg + v.length);
      System.out.println("Contents: ");
      for (int i = 0; i < v.length; i++) {
         System.out.println(" arg " + i + ": " + v[i]);
      }
      System.out.println();
   }
```

Overloading Varargs Methods

```
static void vaTest(int... v) {
    System.out.println("vaTest(int ...): "
        + "Number of args: " + v.length);
    System.out.println("Contents: ");
    for (int i = 0; i < v.length; i++) {
      System.out.println(" arg " + i + ": " + v[i]);
    }
    System.out.println();
  }

  static void vaTest(boolean... v) {
    System.out.println("vaTest(boolean ...): "
        + "Number of args: " + v.length);
    System.out.println("Contents: ");
    for (int i = 0; i < v.length; i++) {
      System.out.println(" arg " + i + ": " + v[i]);
    }
    System.out.println();
  }

  static void vaTest(String msg, int... v) {
    System.out.println("vaTest(String, int ...): "
        + msg + v.length);
    System.out.println("Contents: ");
    for (int i = 0; i < v.length; i++) {
      System.out.println(" arg " + i + ": " + v[i]);
    }
    System.out.println();    }
```

Posibles errors de compilación que se pueden generar:

```
static void vaTest(int... v) {
    // ...
  }
static void vaTest(boolean... v) {
    // ...
```

```
    }
    public static void main(String args[]) {
        vaTest(1, 2, 3); // OK
        vaTest(true, false, false); // OK
        vaTest(); // Error: Ambiguous!
    }
```

porque como vaTest está sobrecargada, no puede discriminar que cuerpo ejecutar

supongamos ahora

```
public static int sumar(int... nums) {
    int total = 0;

    for (int i=0 ; i < nums.length ; i++) {
        total += nums[i];
    }

    return total;
}
```

la llamada que escribimos como: sumar(1, 2, 3, 4, 5), será convertida automáticamente por el compilador en el equivalente: sumar(new int[] {1, 2, 3, 4, 5}). y la llamada sumar() por sumar(new int[] {}).