Variable Number of Arguments

- Methods can have a variable number of arguments

- Methods with variable # of arguments
  can be overloaded

```java
package varargs1;
public class VarArgs1 {
    // Demonstrate variable-length arguments.
      // vaTest() uses a vararg.
      static void vaTest(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();
      }
    public static void main(String[] args) {
        // Notice how vaTest() can be called with a
        // variable number of arguments.
        vaTest(10);       // 1 arg
        vaTest(1, 2, 3); // 3 args
        vaTest();         // no args
    }
}
```

```java
package varargs1;
public class VarArgs1 {
    // Demonstrate variable-length arguments.
      // vaTest() uses a vararg.
      static void vaTest(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();
      }
    public static void main(String[] args) {
        // Notice how vaTest() can be called with a
        // variable number of arguments.
        vaTest(10);      // 1 arg
        vaTest(1, 2, 3); // 3 args
        vaTest();        // no args
    }
}
```

```
Number of args: 1
Contents:
  arg 0: 10

Number of args: 3
Contents:
  arg 0: 1
  arg 1: 2
  arg 2: 3
Number of args: 0
Contents:
```

```java
package varargs2;
public class VarArgs2 {
      // Here, msg is a normal parameter and v is a
      // varargs parameter.
      static void vaTest(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();
      }

    public static void main(String[] args) {
        vaTest("One vararg: ", 10);
        vaTest("Three varargs: ", 1, 2, 3);
        vaTest("No varargs: ");
    }
}
```

```java
package varargs2;
public class VarArgs2 {
      // Here, msg is a normal parameter and v is a
      // varargs parameter.
      static void vaTest(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();
      }

    public static void main(String[] args) {
        vaTest("One vararg: ", 10);
        vaTest("Three varargs: ", 1, 2, 3);
        vaTest("No varargs: ");
    }
}
```

```
One vararg: 1
Contents:
  arg 0: 10

Three varargs: 3
Contents:
  arg 0: 1
  arg 1: 2
  arg 2: 3

No varargs: 0
Contents:
```

- Ambiguity can occur with variable arguments

- With overloaded methods and a call such as
  vaTest(); // no args

Given these two method definitions:
- static void vaTest (int … v) {
- static void vaTest (int n, int ...v) {

If a call is made in the main such as:
vaTest(1);
Which method will be used?