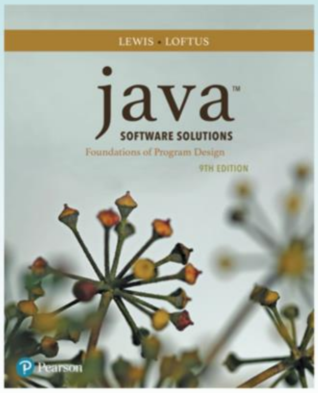
Chapter 8
Arrays



Java Software Solutions

Foundations of Program Design

9th Edition

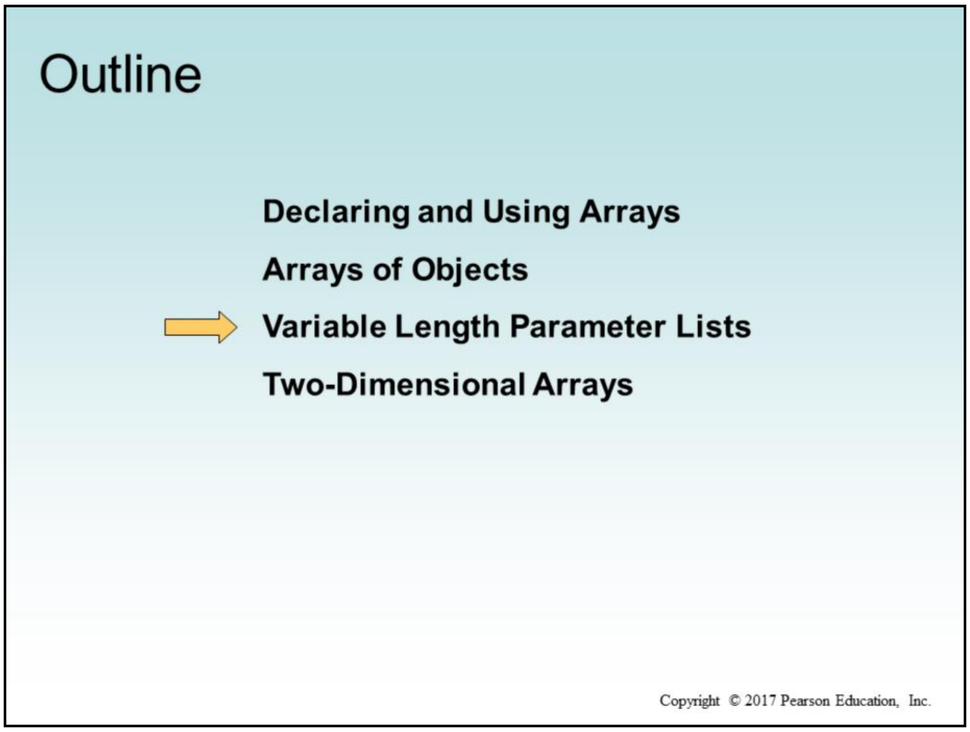
John Lewis
William Loftus

Copyright © 2017 Pearson Education, Inc.

Arrays

- Arrays are objects that help us organize large amounts of information
- Chapter 8 focuses on:
  - array declaration and use
  - bounds checking and capacity
  - arrays that store object references
  - variable length parameter lists
  - multidimensional arrays

Copyright © 2017 Pearson Education, Inc.



Outline

Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

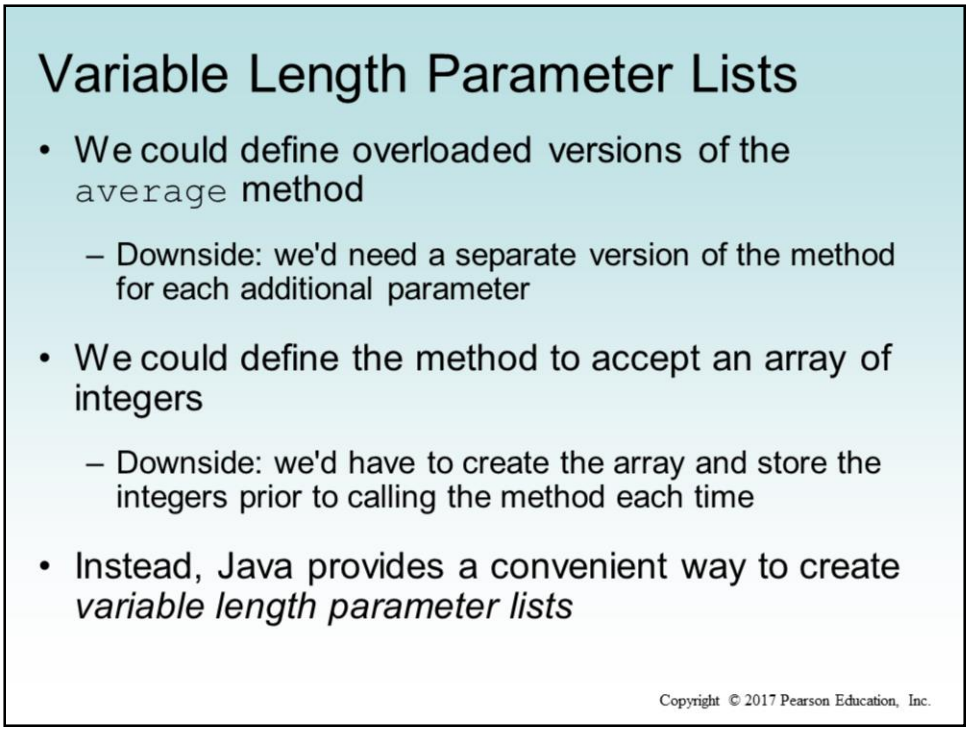
Copyright © 2017 Pearson Education, Inc.

Variable Length Parameter Lists

- Suppose we wanted to create a method that processed a different amount of data from one invocation to the next
- For example, let's define a method called `average` that returns the average of a set of integer parameters

```
// one call to average three values
mean1 = average (42, 69, 37);

// another call to average seven values
mean2 = average (35, 43, 93, 23, 40, 21, 75);
```

Copyright © 2017 Pearson Education, Inc.


Variable Length Parameter Lists

- We could define overloaded versions of the `average` method
  - Downside: we'd need a separate version of the method for each additional parameter
- We could define the method to accept an array of integers
  - Downside: we'd have to create the array and store the integers prior to calling the method each time
- Instead, Java provides a convenient way to create *variable length parameter lists*

Copyright © 2017 Pearson Education, Inc.

-Recall overloaded methods are methods with the same name but different signatures

-In the example from the previous slide, the two overloaded methods would look like:

```
public double average( int a, int b, int c );
public double average( int a, int b, int c, int d, int e, int f, int g );
```

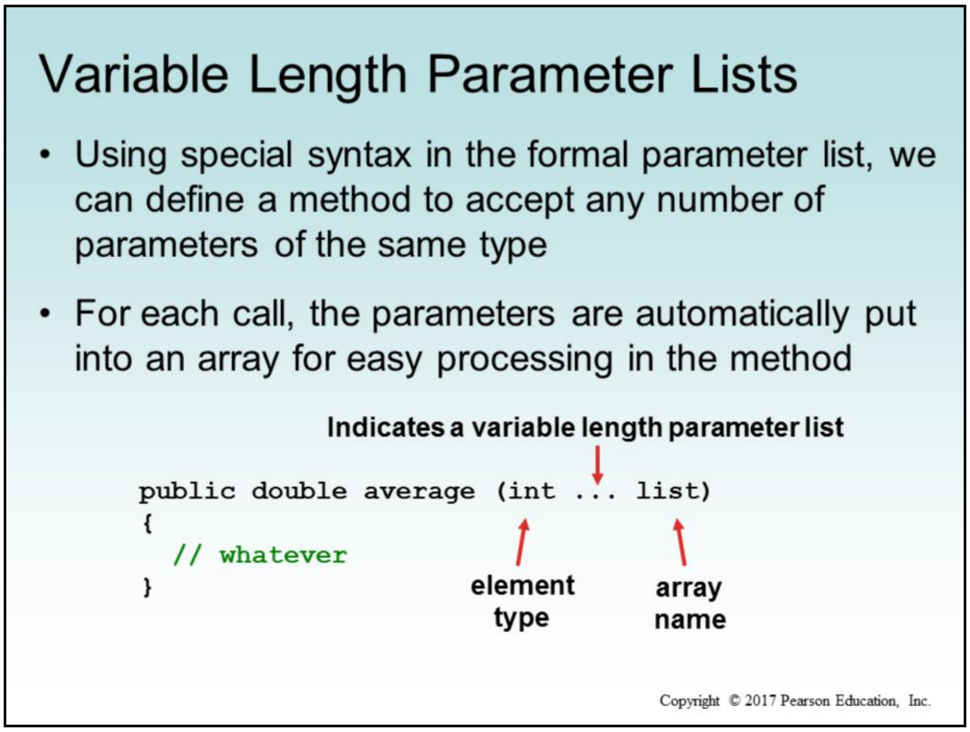
-Instead of using overloaded methods, we can use variable length parameter lists

-Variable length parameter lists allow a method to be called with an arbitrary number of variables (of same type)

-Without this feature, we would have to use overloaded methods, or use an array as the parameter

-These allow a method to be called without requiring data to be packed into an array

-Makes it more flexible when calling the method



Variable Length Parameter Lists

- Using special syntax in the formal parameter list, we can define a method to accept any number of parameters of the same type
- For each call, the parameters are automatically put into an array for easy processing in the method

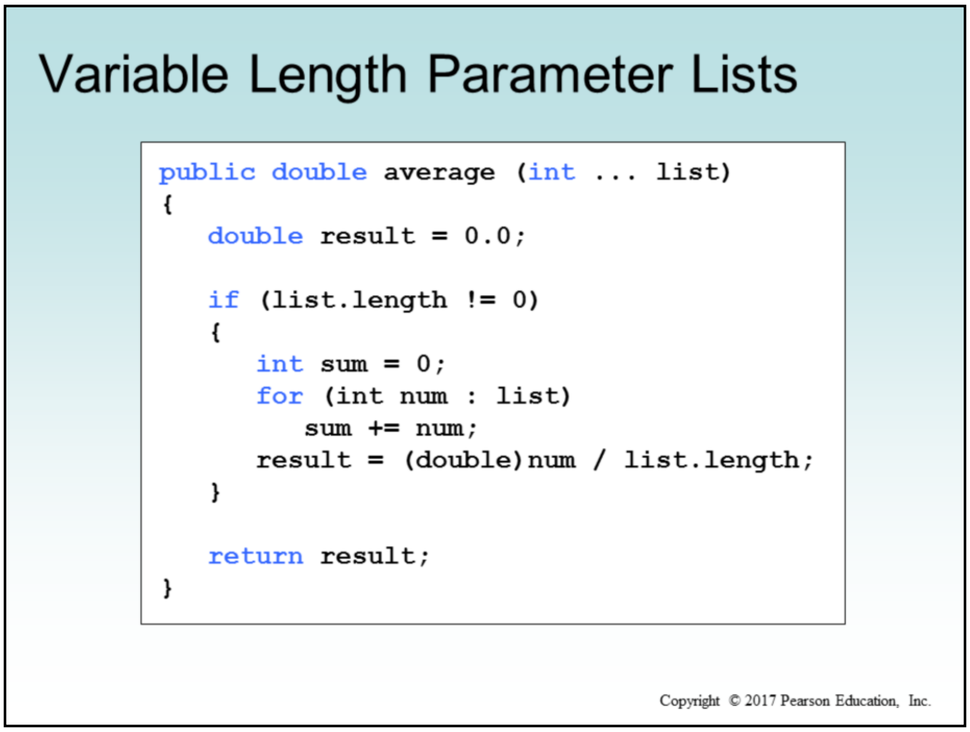
Indicates a variable length parameter list
element type
array name

```
public double average (int ... list)
{
  // whatever
}
```

Copyright © 2017 Pearson Education, Inc.

-Note that Java automatically creates an array from the variable length parameter list!

-This makes it easy to work with the variables that have been passed



Variable Length Parameter Lists

```
public double average (int ... list)
{
   double result = 0.0;

   if (list.length != 0)
   {
      int sum = 0;
      for (int num : list)
         sum += num;
      result = (double)num / list.length;
   }

   return result;
}
```

Copyright © 2017 Pearson Education, Inc.

-Note that the ellipsis (…) is the **actual** syntax for the variable length parameter list

-Note how we treat the variable parameter list the same as we would an array in the body of the method

-Java automatically converts the variable parameter list into an array

-Note the statement that first checks to see that at least one parameter or more has been entered:

if(list.length != 0)

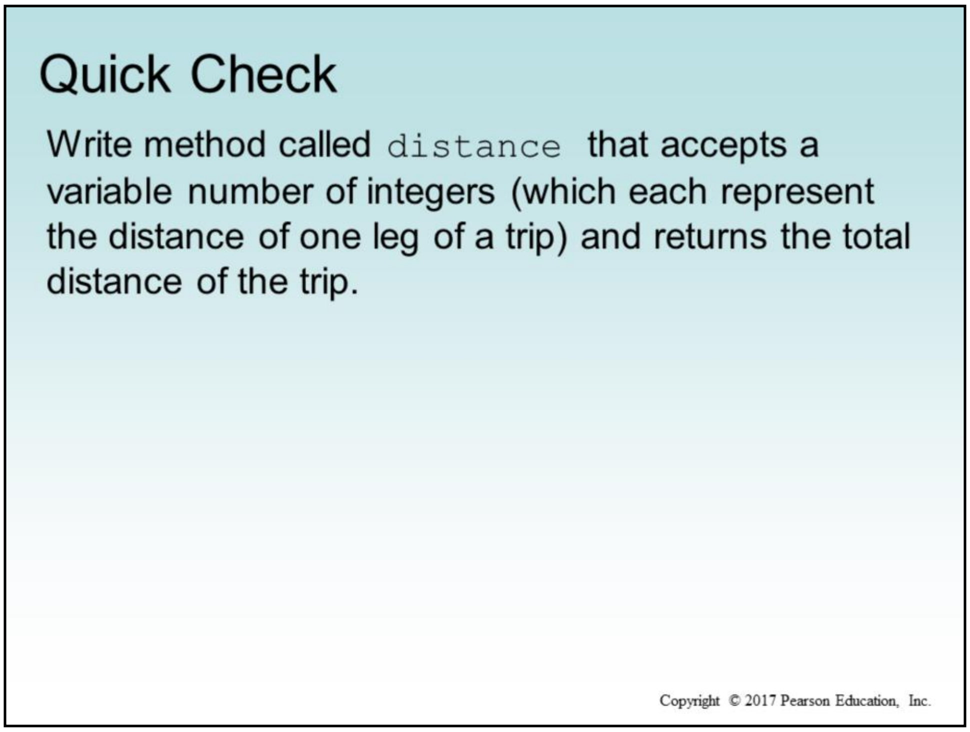
-This checks for the scenario that the method may have been called without **any** parameters

Variable Length Parameter Lists

- The type of the parameter can be any primitive or object type:

```
public void printGrades (Grade ... grades)
{
   for (Grade letterGrade : grades)
      System.out.println (letterGrade);
}
```

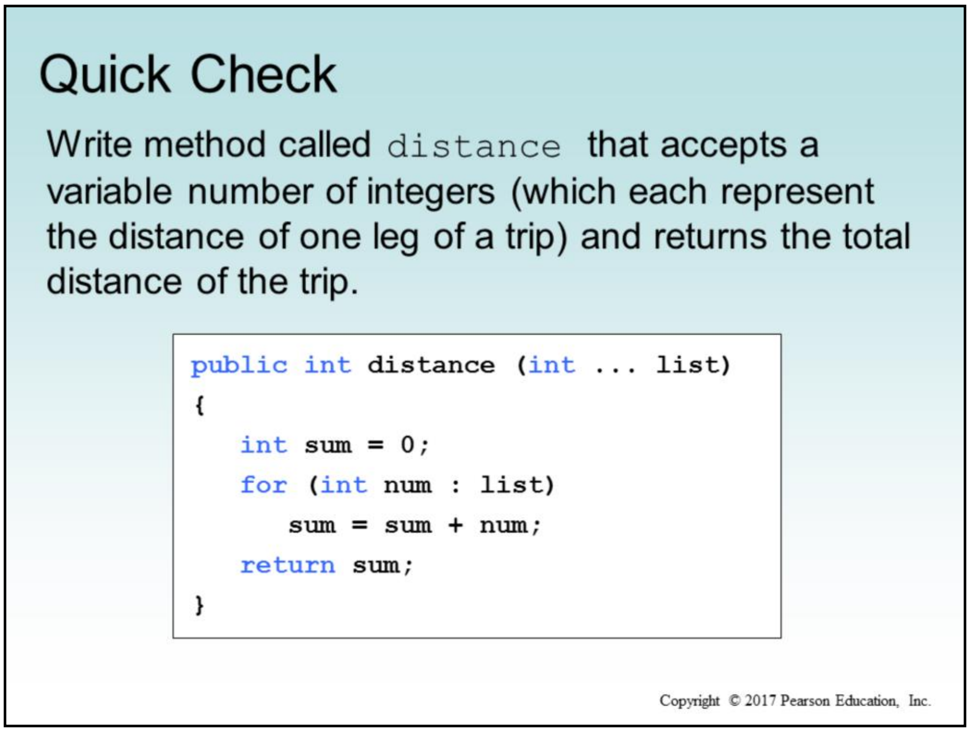
Copyright © 2017 Pearson Education, Inc.



Quick Check

Write method called `distance` that accepts a variable number of integers (which each represent the distance of one leg of a trip) and returns the total distance of the trip.

Copyright © 2017 Pearson Education, Inc.

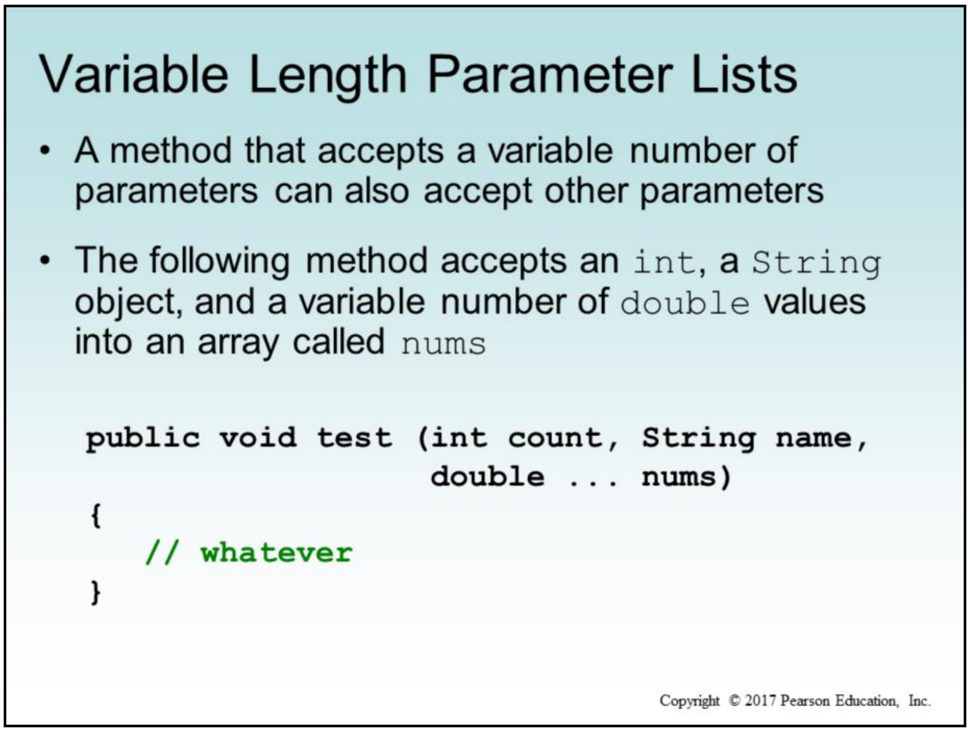


Quick Check

Write method called `distance` that accepts a variable number of integers (which each represent the distance of one leg of a trip) and returns the total distance of the trip.

```
public int distance (int ... list)
{
   int sum = 0;
   for (int num : list)
      sum = sum + num;
   return sum;
}
```

Copyright © 2017 Pearson Education, Inc.



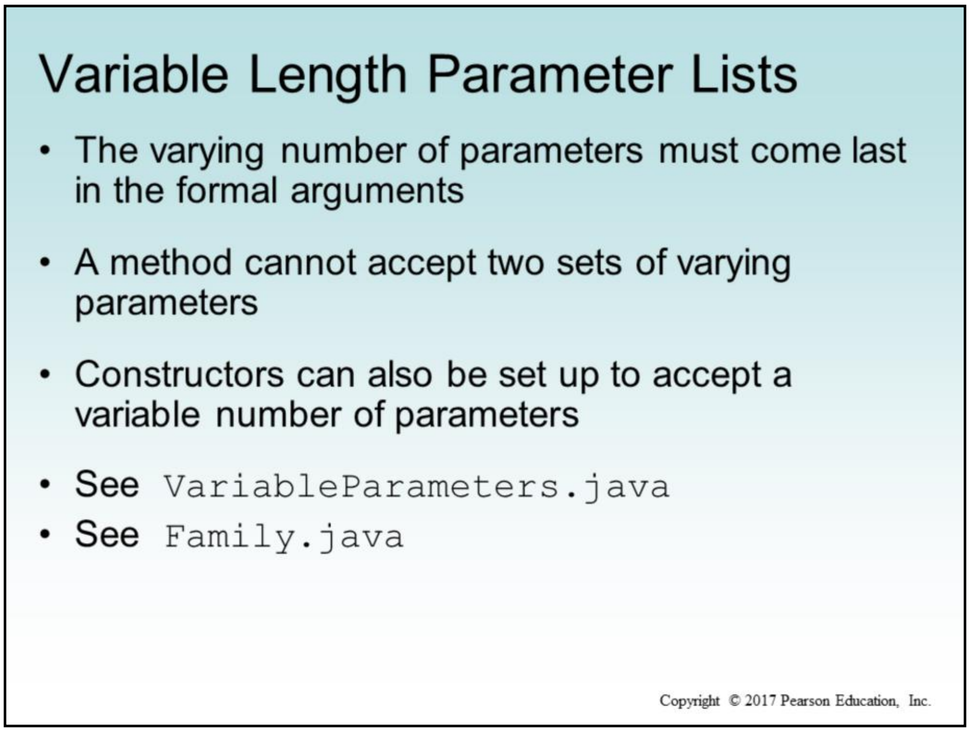
Variable Length Parameter Lists

- A method that accepts a variable number of parameters can also accept other parameters
- The following method accepts an `int`, a `String` object, and a variable number of `double` values into an array called `nums`

```
public void test (int count, String name,
                  double ... nums)
{
   // whatever
}
```

Copyright © 2017 Pearson Education, Inc.

-When other parameters are used as above, the variable length list **must** be specified **last**

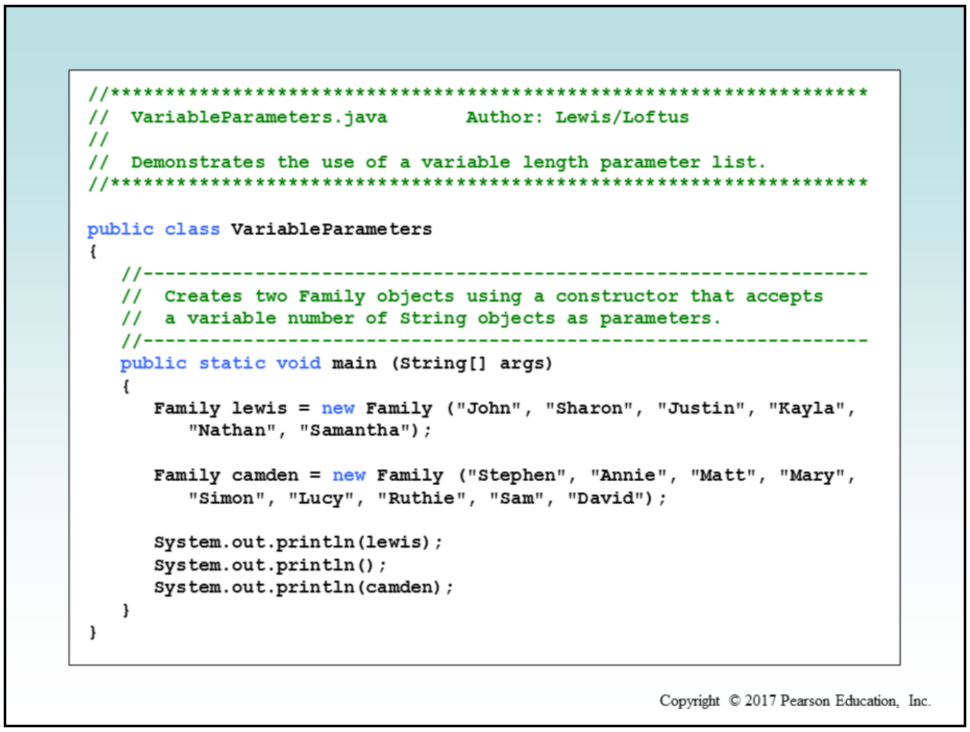

Variable Length Parameter Lists

- The varying number of parameters must come last in the formal arguments
- A method cannot accept two sets of varying parameters
- Constructors can also be set up to accept a variable number of parameters
- See `VariableParameters.java`
- See `Family.java`

Copyright © 2017 Pearson Education, Inc.

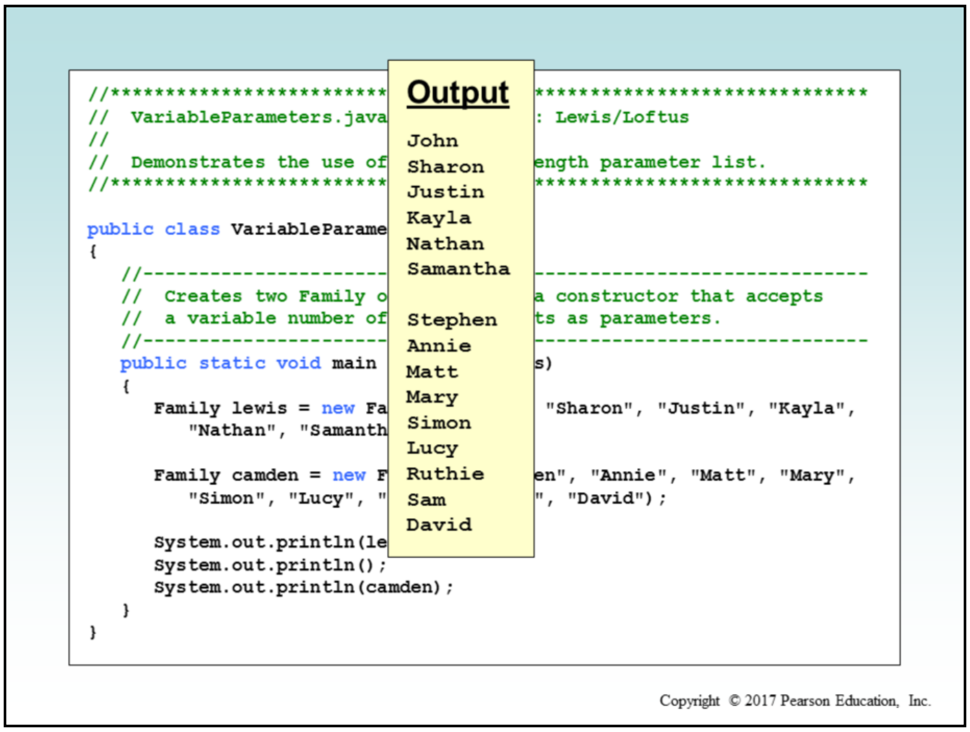
-Note we cannot have two variable parameter lists; the following, for example, is **illegal**

public void function1(String … s, int … i)



```java
//********************************************************************
//  VariableParameters.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a variable length parameter list.
//********************************************************************

public class VariableParameters
{
   //-----------------------------------------------------------------
   //  Creates two Family objects using a constructor that accepts
   //  a variable number of String objects as parameters.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Family lewis = new Family ("John", "Sharon", "Justin", "Kayla",
         "Nathan", "Samantha");

      Family camden = new Family ("Stephen", "Annie", "Matt", "Mary",
         "Simon", "Lucy", "Ruthie", "Sam", "David");

      System.out.println(lewis);
      System.out.println();
      System.out.println(camden);
   }
}
```

Copyright © 2017 Pearson Education, Inc.

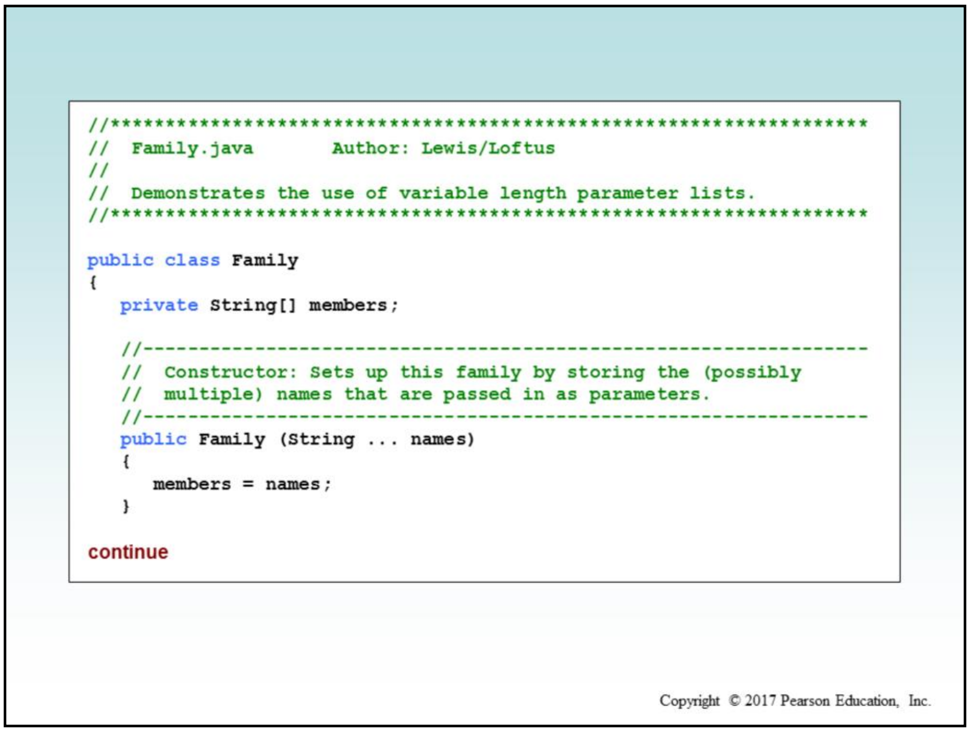


```
//**************************                    ********************************
//  VariableParameters.java                    : Lewis/Loftus
//
//  Demonstrates the use of                    ength parameter list.
//**************************                    ********************************

public class VariableParame
{
   //----------------------                    -------------------------------
   //  Creates two Family o                    a constructor that accepts
   //  a variable number of                    ts as parameters.
   //----------------------                    -------------------------------
   public static void main                    s)
   {
      Family lewis = new Fa                    "Sharon", "Justin", "Kayla",
         "Nathan", "Samanth

      Family camden = new F                    en", "Annie", "Matt", "Mary",
         "Simon", "Lucy", "                    ", "David");

      System.out.println(le
      System.out.println();
      System.out.println(camden);
   }
}
```

Output

```
John
Sharon
Justin
Kayla
Nathan
Samantha

Stephen
Annie
Matt
Mary
Simon
Lucy
Ruthie
Sam
David
```

Copyright © 2017 Pearson Education, Inc.



```
//********************************************************************
//  Family.java       Author: Lewis/Loftus
//
//  Demonstrates the use of variable length parameter lists.
//********************************************************************

public class Family
{
   private String[] members;

   //-----------------------------------------------------------------
   //  Constructor: Sets up this family by storing the (possibly
   //  multiple) names that are passed in as parameters.
   //-----------------------------------------------------------------
   public Family (String ... names)
   {
      members = names;
   }
```

continue

Copyright © 2017 Pearson Education, Inc.

-Note the constructor of this class is using a variable-length parameter list to load an array of Strings

-Since the variable list is converted to an array, we can assign it to the instance array in the class continue

```
   //-----------------------------------------------------------------
   //  Returns a string representation of this family.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result = "";

      for (String name : members)
         result += name + "\n";

      return result;
   }
}
```

Copyright © 2017 Pearson Education, Inc.