4.3 Example: Volume of a Sphere

- Given the radius r, what is the weight of a ball (sphere) of wound twine?
- Object-Centered Design
  - —display prompt for radius
  - —read value for radius
  - —compute weight of sphere
  - —display results on screen
- Note this is generalized for sphere of arbitrary size

Objects

Objects	Type	Kind	Name
Program	??	—	??
Screen	`Screen`	varying	***`theScreen`***
Prompt	`String`	constant	none
Radius	`double`	varying	***`radius`***
Keyboard	`Keyboard`	varying	***`theKeyboard`***
Weight	`double`	varying	***`weight`***
Sphere	??	varying	??

Operations

- Display a **`String`** (prompt) on the screen
- Read a number from keyboard, store it in ***`radius`***
- Compute ***`weight`*** using ***`radius`***
- Display a number (***`weight`***) on screen

New Class Required

- Java has no predefined operation for volume or weight of a sphere
- Also no predefined sphere object
- Solution:
  - — build methods to calculate volume & weight
  - — create a sphere class (*module*)to store the methods

```
class Sphere
{
   // method definitions
}
```

- We will need an additional variable object
  - `density     (weight = density * volume)`

Volume Method – Objects

- Volume = $4\pi r^3 / 3$
- Note
  - — r is the only variable
  - — 4, 3, and π are constants
- These (along with the result, volume) are the objects of this method

Volume Method – Operations and Algorithm

- Receive real value (`radius`) from caller
- Cube the real value (`radius³`)
- Multiply by 4.0 and by π
- Divide by 3.0
- Return result `4.0 * π * radius³/3.0`

Defining the Class and Method

Can start with an empty class

```
class Sphere extends Object
{

}
```

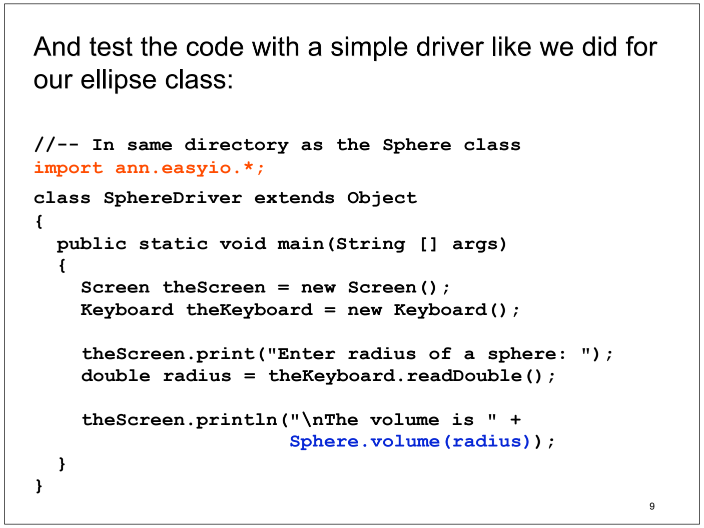
and add a ***method stub***

```
class Sphere extends Object
{
  public static double volume(double radius)
  {
  }
}
```

Compiler error – "no return statement" – will result. If this is the only error, we know the rest is okay. Or we could could add a temporary "return 0" to method stub to avoid this.

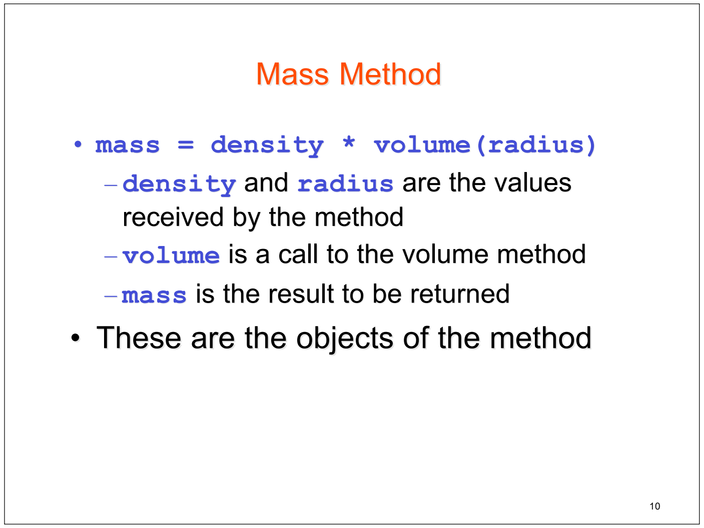
Then code the method's algorithm in the body of the method:

```
class Sphere extends Object
{
  public static double volume(double radius)
  {
    return  4.0 * Math.PI *
                  Math.pow(radius, 3)/3.0;
  }
}
```



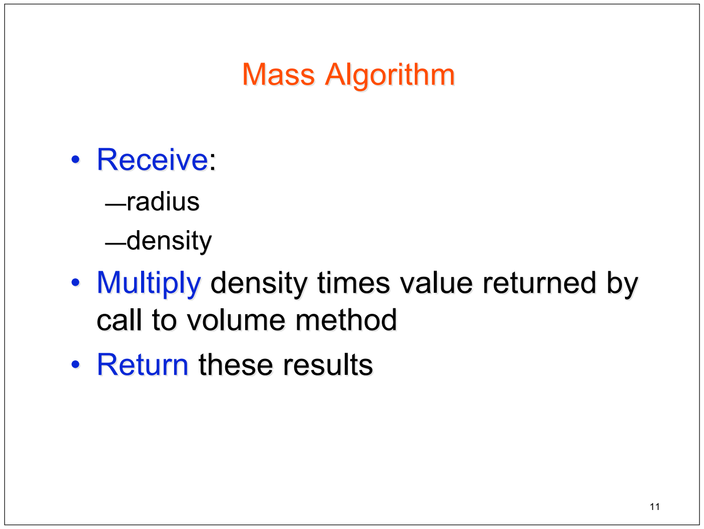
And test the code with a simple driver like we did for our ellipse class:

```
//-- In same directory as the Sphere class
import ann.easyio.*;
class SphereDriver extends Object
{
  public static void main(String [] args)
  {
    Screen theScreen = new Screen();
    Keyboard theKeyboard = new Keyboard();

    theScreen.print("Enter radius of a sphere: ");
    double radius = theKeyboard.readDouble();

    theScreen.println("\nThe volume is " +
                      Sphere.volume(radius));
  }
}
```



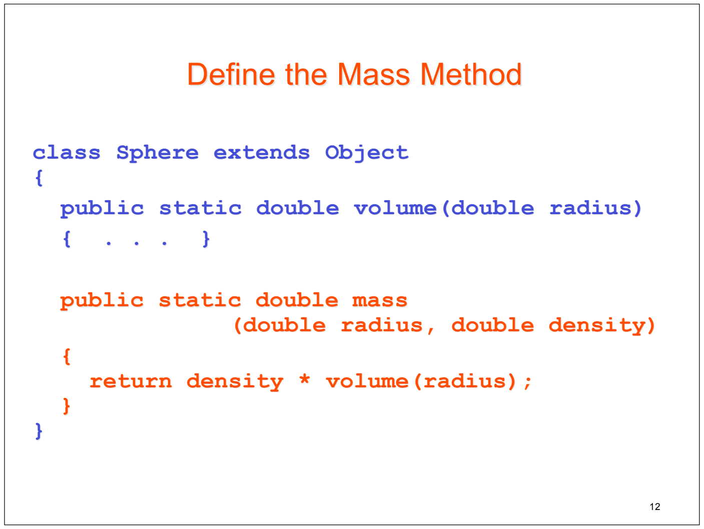
Mass Method

- `mass = density * volume(radius)`
  - `density` and `radius` are the values received by the method
  - `volume` is a call to the volume method
  - `mass` is the result to be returned
- These are the objects of the method



Mass Algorithm

- Receive:
  - —radius
  - —density
- Multiply density times value returned by call to volume method
- Return these results



Define the Mass Method

```
class Sphere extends Object
{
  public static double volume(double radius)
  { . . . }

  public static double mass
               (double radius, double density)
  {
    return density * volume(radius);
  }
}
```

Algorithm for Main Method

- Construct **theKeyboard**, **theScreen**
- **theScreen** displays prompt for **radius**
- **theKeyboard** reads double value into **radius**
- **theScreen** displays prompt for **density**
- **theKeyboard** reads a double into **density**
- Compute **weight**, using **mass()** method from class **Sphere**
- **theScreen** displays **weight** and descriptive text

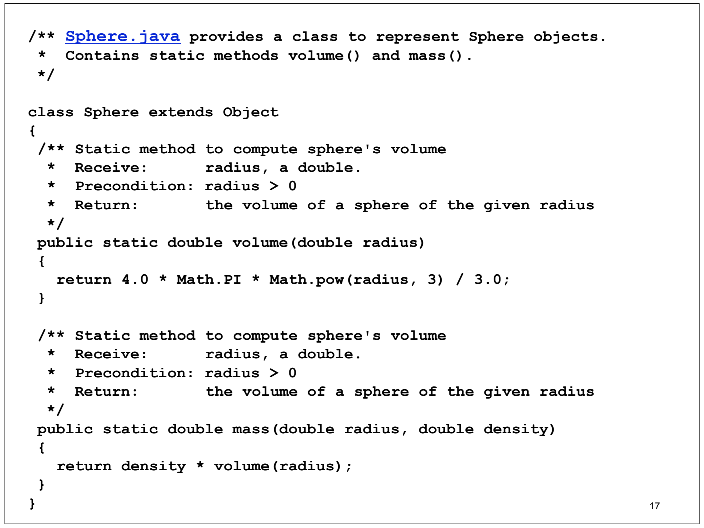
Test the Mass Method

```
//-- In same directory as the Sphere class
import ann.easyio.*;

class SphereDriver extends Object
{
  public static void main(String [] args)
  {
    Screen theScreen = new Screen();
    Keyboard theKeyboard = new Keyboard();

    theScreen.print("Enter radius of a sphere: ");
    double radius = theKeyboard.readDouble();
    theScreen.println("\nThe volume is " +
                          Sphere.volume(radius));
    theScreen.print("Enter density: ");
    double density = theKeyboard.readDouble();
    theScreen.println("\nThe mass is " +
                          Sphere.mass(radius, density));
  }
}
```

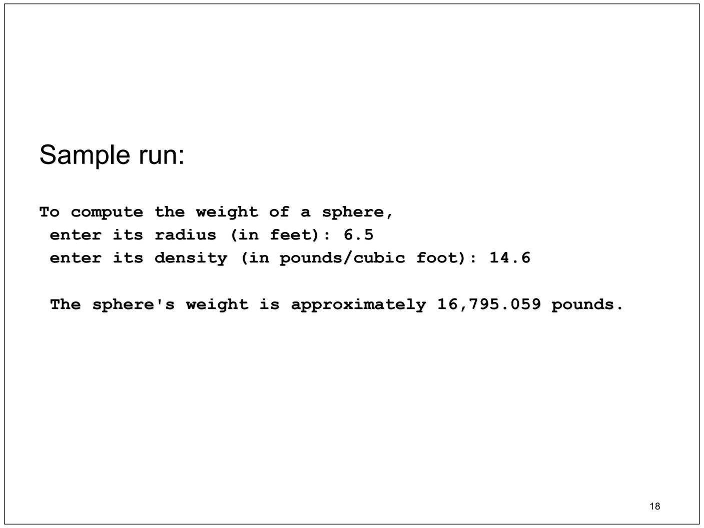
Code and Teste **SphereWeigher** Class for Original Problem

- Note source code in Figure 4.5
  - Delete **import Sphere class**;
    Put **Sphere** class in same directory as the *client program*
  - How it uses methods from **Sphere** class

```
/** SphereWeigher.java computes the weight of an arbitrary sphere.
 *  Input: radius and density, both doubles.
 *  Output: the weight of the sphere.
 */
import ann.easyio.*;                         // Keyboard, Screen, ...
~~import Sphere;~~

class SphereWeigher extends Object
{
 public static void main(String [] args)
 {
   Screen theScreen = new Screen();
   theScreen.print("\nTo compute the weight of a sphere,"
                  + "\n enter its radius (in feet): ");

   Keyboard theKeyboard = new Keyboard();
   double radius = theKeyboard.readDouble();

   theScreen.print(" enter its density (in pounds/cubic foot): ");
   double density = theKeyboard.readDouble();

   double weight = Sphere.mass(radius, density);

   theScreen.print("\nThe sphere's weight is approximately ")
            .printFormatted(weight).println(" pounds.");
 }
}
```

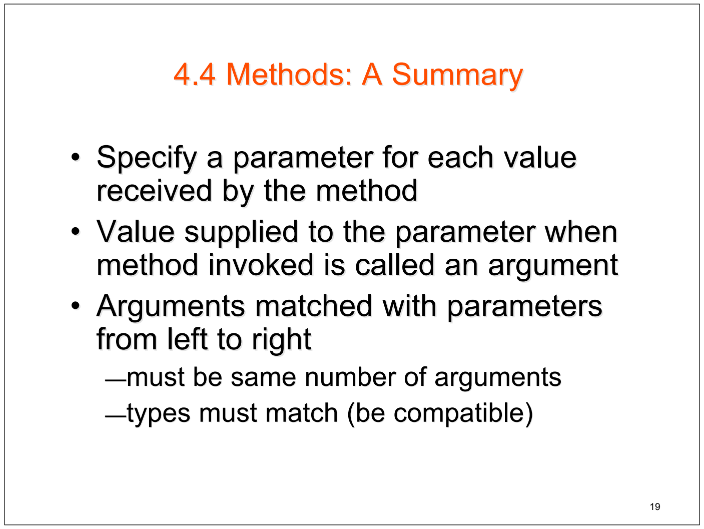


```
/** Sphere.java provides a class to represent Sphere objects.
 *  Contains static methods volume() and mass().
 */

class Sphere extends Object
{
 /** Static method to compute sphere's volume
  *  Receive:      radius, a double.
  *  Precondition: radius > 0
  *  Return:       the volume of a sphere of the given radius
  */
 public static double volume(double radius)
 {
   return 4.0 * Math.PI * Math.pow(radius, 3) / 3.0;
 }

 /** Static method to compute sphere's volume
  *  Receive:      radius, a double.
  *  Precondition: radius > 0
  *  Return:       the volume of a sphere of the given radius
  */
 public static double mass(double radius, double density)
 {
   return density * volume(radius);
 }
}
```



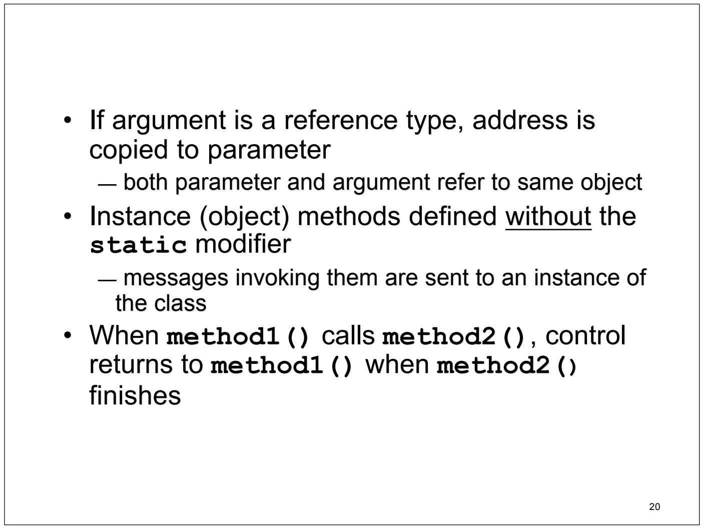
Sample run:

```
To compute the weight of a sphere,
 enter its radius (in feet): 6.5
 enter its density (in pounds/cubic foot): 14.6

 The sphere's weight is approximately 16,795.059 pounds.
```

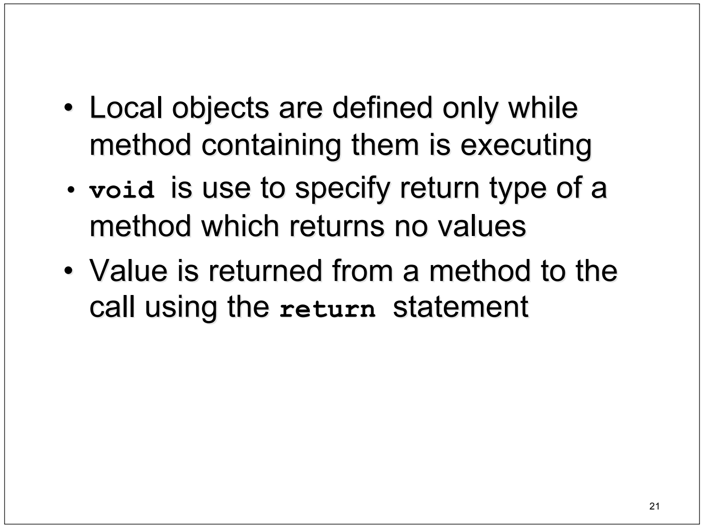


4.4 Methods: A Summary

- Specify a parameter for each value received by the method
- Value supplied to the parameter when method invoked is called an argument
- Arguments matched with parameters from left to right
  - —must be same number of arguments
  - —types must match (be compatible)



- If argument is a reference type, address is copied to parameter
  - — both parameter and argument refer to same object
- Instance (object) methods defined without the **`static`** modifier
  - — messages invoking them are sent to an instance of the class
- When **`method1()`** calls **`method2()`**, control returns to **`method1()`** when **`method2()`** finishes



- Local objects are defined only while method containing them is executing
- **`void`** is use to specify return type of a method which returns no values
- Value is returned from a method to the call using the **`return`** statement