# Passing Parameters to and from Methods; Class Math

Mitsu Ogihara

Department of Computer Science
University of Miami

# Table of Contents

# Methods That Receive Parameters

Suppose we are writing a program for producing on the screen a number of right-angle equilateral triangles of various sizes (the height and the width are equal to each other)

Do we need to write the code for all the possible dimensions?

# A Solution That Will Not Work

```
1   ...
2   public static void triangle() {
3     int height = 10;
4     for ( int posV = 1; posV <= height; posV ++ ) {
5       for ( int posH = 1; posH <= posV; posH ++ ) {
6         System.out.print( "#" );
7       }
8       System.out.println();
9     }
10  }
11  public static void main( String[] args ) {
12    int height = 5;
13    triangle();
14    height = 7;
15    triangle();
16  }
17  ...
```

Will this generate a height-5 triangle and then a height-7 triangle?

# The Issue and a Solution

The scope of the `height` in `main` does not intersect with the scope of the `height` in `triangle` and so changing the value of the former has nothing to do with the value of the latter

# The Issue and a Solution

The scope of the `height` in `main` does not intersect with the scope of the `height` in `triangle` and so changing the value of the former has nothing to do with the value of the latter

**Solution:** make `triangle` operate with a value of `height` assigned from outside

```
1   ...
2   public static void triangle( int height ) {
3     for ( int posV = 1; posV <= height; posV ++ ) {
4       for ( int posH = 1; posH <= posV; posH ++ ) {
5         System.out.print( "#" );
6       }
7       System.out.println();
8     }
9   }
10  public static void main( String[] args ) {
11    int height = 5;
12    triangle( height );
13    height = 7;
14    triangle( height );
15  }
16  ...
```

# General Parameter Specification

The parameter specification of a method takes the form of:

```
public static void <name>(<type_1> <name_1>, ...,
                          <type_k> <name_k>)
```

These are called **formal parameters**
The sequence of types appearing in the parameter area of a method is called the **type signature** of the method

# Passing Mechanism

When the method `triangle( height )` is called, the value of the variable `height` in the main method is stored in the variable `height` of `triangle` before the method starts operating

The two methods have disjoint scopes and so any changes made in `triangle` do not affect the value of `height` in the main method

You may give distinct names to the two variables without changing the behavior of the program

# Method Overloading

In a Java class multiple methods having **an identical name and pairwise distinct signatures** can be defined

# Method Overloading

In a Java class multiple methods having **an identical name and pairwise distinct signatures** can be defined

This is called **method overloading**

# Table of Contents

# Methods That Return Values

**A method can be declared to return a value**

# Methods That Return Values

**A method can be declared to return a value**

To return a value the method declaration has to have the type of the value to be returned in place of `void`,

```
public static <type> <name>(<parameters>)
```

# Methods That Return Values

**A method can be declared to return a value**

To return a value the method declaration has to have the type of the value to be returned in place of `void`,

```
public static <type> <name>(<parameters>)
```

This requires the method contain at least one statement of the form

```
return VALUE;
```

for returning a value of the required type, where

# Methods That Return Values

**A method can be declared to return a value**

To return a value the method declaration has to have the type of the value to be returned in place of `void`,

```
public static <type> <name>(<parameters>)
```

This requires the method contain at least one statement of the form

```
return VALUE;
```

for returning a value of the required type, where

- As soon as a `return` statement is encountered the execution of the method is terminated, and so

# Methods That Return Values

## A method can be declared to return a value

To return a value the method declaration has to have the type of the value to be returned in place of `void`,

```
public static <type> <name>(<parameters>)
```

This requires the method contain at least one statement of the form

```
return VALUE;
```

for returning a value of the required type, where

- As soon as a `return` statement is encountered the execution of the method is terminated, and so
- Regardless of the flow of execution a return statement should be encountered
  $\rightarrow$ Enforced by the compiler

# Method for Computing BMI

Design a method `calculate` that receives weight and height, computes BMI, and then returns the BMI value

The method has the header

```
public static double calculate( double weight, double
                        height )
```

The `double` preceding `calculate` implies that the method should return a `double` value

# The Calculation Method

```
2   public static double calculate( double weight, double height ) {
3     double bmi = 703.0 * weight / ( height * height );
4     return bmi;
5   }
```

The value calculation

# The Calculation Method

```
2   public static double calculate( double weight, double height ) {
3     double bmi = 703.0 * weight / ( height * height );
4     return bmi;
5   }
```

Returning the value

Alternatively the body of method can be simplified to a single line

```
return bmi = 703.0 * weight / ( height * height );
```

without having to store the result in a variable

# Class BMIReturnValue

```
6   public static void main( String[] args ) {
7     double weight, height, bmi;
8     System.out.println( "******** This is round 1" );
9     weight = 150.0;
10    height = 67.0;
11    bmi = calculate( weight, height );
12    System.out.println( "weight is " + weight + " lb" );
13    System.out.println( "height is " + height + " in" );
14    System.out.println( "   bmi is " + bmi );
15    System.out.println( "******** This is round 2" );
16    weight = 170.0;
17    height = 72.0;
18    bmi = calculate( weight, height );
19    System.out.println( "weight is " + weight + " lb" );
20    System.out.println( "height is " + height + " in" );
21    System.out.println( "   bmi is " + bmi );
22  }
```

The variable declarations

# Class BMIReturnValue

```
6   public static void main( String[] args ) {
7     double weight, height, bmi;
8     System.out.println( "******** This is round 1" );
9     weight = 150.0;
10    height = 67.0;
11    bmi = calculate( weight, height );
12    System.out.println( "weight is " + weight + " lb" );
13    System.out.println( "height is " + height + " in" );
14    System.out.println( "   bmi is " + bmi );
15    System.out.println( "******** This is round 2" );
16    weight = 170.0;
17    height = 72.0;
18    bmi = calculate( weight, height );
19    System.out.println( "weight is " + weight + " lb" );
20    System.out.println( "height is " + height + " in" );
21    System.out.println( "   bmi is " + bmi );
22  }
```

Assign values to weight and height.

# Class BMIReturnValue

```
6   public static void main( String[] args ) {
7     double weight, height, bmi;
8     System.out.println( "******** This is round 1" );
9     weight = 150.0;
10    height = 67.0;
11    bmi = calculate( weight, height );
12    System.out.println( "weight is " + weight + " lb" );
13    System.out.println( "height is " + height + " in" );
14    System.out.println( "   bmi is " + bmi );
15    System.out.println( "******** This is round 2" );
16    weight = 170.0;
17    height = 72.0;
18    bmi = calculate( weight, height );
19    System.out.println( "weight is " + weight + " lb" );
20    System.out.println( "height is " + height + " in" );
21    System.out.println( "   bmi is " + bmi );
22  }
```

Receive the value from the calculation method

# Class BMIReturnValue

```
6   public static void main( String[] args ) {
7     double weight, height, bmi;
8     System.out.println( "******** This is round 1" );
9     weight = 150.0;
10    height = 67.0;
11    bmi = calculate( weight, height );
12    System.out.println( "weight is " + weight + " lb" );
13    System.out.println( "height is " + height + " in" );
14    System.out.println( "   bmi is " + bmi );
15    System.out.println( "******** This is round 2" );
16    weight = 170.0;
17    height = 72.0;
18    bmi = calculate( weight, height );
19    System.out.println( "weight is " + weight + " lb" );
20    System.out.println( "height is " + height + " in" );
21    System.out.println( "   bmi is " + bmi );
22  }
```

Output the result

# Class BMIReturnValue

```
6   public static void main( String[] args ) {
7     double weight, height, bmi;
8     System.out.println( "******** This is round 1" );
9     weight = 150.0;
10    height = 67.0;
11    bmi = calculate( weight, height );
12    System.out.println( "weight is " + weight + " lb" );
13    System.out.println( "height is " + height + " in" );
14    System.out.println( "   bmi is " + bmi );
15    System.out.println( "******** This is round 2" );
16    weight = 170.0;
17    height = 72.0;
18    bmi = calculate( weight, height );
19    System.out.println( "weight is " + weight + " lb" );
20    System.out.println( "height is " + height + " in" );
21    System.out.println( "   bmi is " + bmi );
22  }
```

Second round

# The Metric BMI

Design a special method for computing the BMI **when the height and the weight are given in metric**

# The Metric BMI

Design a special method for computing the BMI **when the height and the weight are given in metric**

IDEA: Use two methods:

- A method that receives a length in meters and returns the value in inches
- A method that receives a weight in kilograms and returns the value in pounds

# Methods Can Be Used to Calculate Parameter Values

```
1  public class BMIMetric {
2    //-- convert kilograms to pounds
3    public static double kgToLb( double kgWeight ) {
4      return 2.20462 * kgWeight;
5    }
6    //-- convert meters to inches
7    public static double mToIn( double meter ) {
8      return 39.3701 * meter;
9    }
10   //-- calculation for pounds and inches
11   public static double calculate(
12       double weight, double height ) {
13     return 703.0 * weight / ( height * height );
14   }
15   //-- calculation for kilograms and meters
16   public static double metricCalculate(
17       double weight, double height ) {
18     return calculate( kgToLb( weight ), mToIn( height ) );
19   }
```

Kilogram-to-pound Conversion

# Methods Can Be Used to Calculate Parameter Values

```
1  public class BMIMetric {
2    //-- convert kilograms to pounds
3    public static double kgToLb( double kgWeight ) {
4      return 2.20462 * kgWeight;
5    }
6    //-- convert meters to inches
7    public static double mToIn( double meter ) {
8      return 39.3701 * meter;
9    }
10   //-- calculation for pounds and inches
11   public static double calculate(
12       double weight, double height ) {
13     return 703.0 * weight / ( height * height );
14   }
15   //-- calculation for kilograms and meters
16   public static double metricCalculate(
17       double weight, double height ) {
18     return calculate( kgToLb( weight ), mToIn( height ) );
19   }
```

Meter-to-inch Conversion

# Methods Can Be Used to Calculate Parameter Values

```
1  public class BMIMetric {
2    //-- convert kilograms to pounds
3    public static double kgToLb( double kgWeight ) {
4      return 2.20462 * kgWeight;
5    }
6    //-- convert meters to inches
7    public static double mToIn( double meter ) {
8      return 39.3701 * meter;
9    }
10   //-- calculation for pounds and inches
11   public static double calculate(
12       double weight, double height ) {
13     return 703.0 * weight / ( height * height );
14   }
15   //-- calculation for kilograms and meters
16   public static double metricCalculate(
17       double weight, double height ) {
18     return calculate( kgToLb( weight ), mToIn( height ) );
19   }
```

BMI with pounds and inches

# Methods Can Be Used to Calculate Parameter Values

```
1  public class BMIMetric {
2    //-- convert kilograms to pounds
3    public static double kgToLb( double kgWeight ) {
4      return 2.20462 * kgWeight;
5    }
6    //-- convert meters to inches
7    public static double mToIn( double meter ) {
8      return 39.3701 * meter;
9    }
10   //-- calculation for pounds and inches
11   public static double calculate(
12       double weight, double height ) {
13     return 703.0 * weight / ( height * height );
14   }
15   //-- calculation for kilograms and meters
16   public static double metricCalculate(
17       double weight, double height ) {
18     return calculate( kgToLb( weight ), mToIn( height ) );
19   }
```

BMI with kilograms and meters; the returned values from the conversion methods are **passed directly to the American standard version**

# The Rest of the Code

```
20    //-- main method
21    public static void main( String[] args ) {
22      double weight, height, bmi;
23      weight = 65.5;
24      height = 1.75;
25      bmi = metricCalculate( weight, height );
26      System.out.println( "weight is " + weight + " Kg" );
27      System.out.println( "height is " + height + " m" );
28      System.out.println( "   bmi is " + bmi );
29    }
30  }
```

Declaration of the variables and assignment to them

# The Rest of the Code

```
20    //-- main method
21    public static void main( String[] args ) {
22      double weight, height, bmi;
23      weight = 65.5;
24      height = 1.75;
25      bmi = metricCalculate( weight, height );
26      System.out.println( "weight is " + weight + " Kg" );
27      System.out.println( "height is " + height + " m" );
28      System.out.println( "   bmi is " + bmi );
29    }
30  }
```

Calculation and output generation

# Table of Contents

# Math

Math is a class provides a number of useful mathematical constantts and functions

Math provides support for computing the standard mathematical functions
The use of Math functions takes the form of:

```
Math.FUNCTION-NAME(PARAMETERS)
```

The use of Math constants takes the form of:

```
Math.CONSTANT-NAME
```

# Constants and Methods Requiring No Parameter

- `Math.PI`: returns the value of $\pi$
- `Math.E`: returns the value of the base of the natural logarithm
- `Math.random()`: returns a random double value between 0 and 1 (1 is never generated) under a uniform distribution

# Methods with One Parameter

Both the parameter type and the return type are double

| Name | What it computes |
|---|---|
| `sin` | The sine of the parameter value (radian) |
| `cos` | The cosine of the parameter value (radian) |
| `tan` | The tangent of the parameter value (radian) |
| `asin` | The inverse of sine, return value in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ |
| `acos` | The inverse of sine, return value in $[0, \pi]$ |
| `atan` | The inverse of sine, return value in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ |
| `sqrt` | The square root |
| `cbrt` | The cubic root |
| `log` | The natural logarithm |
| `log10` | The logarithm base 10 |
| `signum` | The sign of the number, -1.0, 0.0, or +1.0 |
| `exp` | The exponential function base the natural log. |

# Methods with One Parameter (cont'd)

Both the parameter type and the return type are double

| | |
|---|---|
| `ceil` | The smallest whole number that is >= parameter |
| `floor` | The largest whole number <= parameter |
| `round` | The rounded whole number, as an int |
| `abs` | The absolute value |

# Math Methods with Two Parameters

| | |
|---|---|
| `max` | The maximum of the two numbers given as parameters |
| `min` | The minimum of the two numbers given as parameters |

These methods are defined for each number type by way of method overloading

| | |
|---|---|
| `pow` | The first parameter raised to the power of the second |

The value returned by `pow` is double regardless of the number types provided as parameters

# No Parameters

```
1  // examples of math functions required no parameters
2  public class MathNoParameters {
3    public static void main( String[] args ) {
4      System.out.println( "Math.PI = " + Math.PI );
5      System.out.println( "Math.E= " + Math.E );
6      for ( int count = 1; count <= 5; count ++ ) {
7        System.out.print( "Round " + count );
8        System.out.println( ": Math.random() = " + Math.random() );
9      }
10   }
11 }
```

Math.PI

# No Parameters

```
1  // examples of math functions required no parameters
2  public class MathNoParameters {
3    public static void main( String[] args ) {
4      System.out.println( "Math.PI = " + Math.PI );
5      System.out.println( "Math.E= " + Math.E );
6      for ( int count = 1; count <= 5; count ++ ) {
7        System.out.print( "Round " + count );
8        System.out.println( ": Math.random() = " + Math.random() );
9      }
10   }
11 }
```

Math.E

# No Parameters

```
1  // examples of math functions required no parameters
2  public class MathNoParameters {
3    public static void main( String[] args ) {
4      System.out.println( "Math.PI = " + Math.PI );
5      System.out.println( "Math.E= " + Math.E );
6      for ( int count = 1; count <= 5; count ++ ) {
7        System.out.print( "Round " + count );
8        System.out.println( ": Math.random() = " + Math.random() );
9      }
10   }
11 }
```

Math.random

# Result of No Parameter Methods

```
% java MathNoParameters
Math.PI = 3.141592653589793
Math.E= 2.718281828459045
Round 1: Math.random() = 0.056618315818746656
Round 2: Math.random() = 0.30658632116385387
Round 3: Math.random() = 0.07808433189065977
Round 4: Math.random() = 0.27893273824439646
Round 5: Math.random() = 0.752651071169672
```

# One Parameter Methods

```
1  // Math functions requiring 1 parameter, excluding trigonometry
2  public class MathOneParameter{
3    public static void main( String[] args ) {
4      double real;
5      real = 10.0;
6      System.out.println( "real is " + real );
7      System.out.println( "log(real) is " + Math.log( real ) );
8      System.out.println( "log10(real) is " + Math.log10( real ) );
9      real = 64.0;
10     System.out.println( "real is " + real );
11     System.out.println( "sqrt(real) is " + Math.sqrt( real ) );
12     System.out.println( "cbrt(real) is " + Math.cbrt( real ) );
13     System.out.println( "exp(real) is " + Math.exp( real ) );
14     System.out.println( "abs(real) is " + Math.abs( real ) );
15     System.out.println( "signum(real) is " + Math.signum( real ) );
16     real = 95.789;
17     System.out.println( "real is " + real );
18     System.out.println( "ceil(real) is " + Math.ceil( real ) );
19     System.out.println( "round(real) is " + Math.round( real ) );
20     System.out.println( "floor(real) is " + Math.floor( real ) );
21   }
22 }
```

Analytic functions requiring positive parameter

# One Parameter Methods

```
1  // Math functions requiring 1 parameter, excluding trigonometry
2  public class MathOneParameter{
3    public static void main( String[] args ) {
4      double real;
5      real = 10.0;
6      System.out.println( "real is " + real );
7      System.out.println( "log(real) is " + Math.log( real ) );
8      System.out.println( "log10(real) is " + Math.log10( real ) );
9      real = 64.0;
10     System.out.println( "real is " + real );
11     System.out.println( "sqrt(real) is " + Math.sqrt( real ) );
12     System.out.println( "cbrt(real) is " + Math.cbrt( real ) );
13     System.out.println( "exp(real) is " + Math.exp( real ) );
14     System.out.println( "abs(real) is " + Math.abs( real ) );
15     System.out.println( "signum(real) is " + Math.signum( real ) );
16     real = 95.789;
17     System.out.println( "real is " + real );
18     System.out.println( "ceil(real) is " + Math.ceil( real ) );
19     System.out.println( "round(real) is " + Math.round( real ) );
20     System.out.println( "floor(real) is " + Math.floor( real ) );
21   }
22 }
```

Other analytic functions

# One Parameter Methods

```
1  // Math functions requiring 1 parameter, excluding trigonometry
2  public class MathOneParameter{
3    public static void main( String[] args ) {
4      double real;
5      real = 10.0;
6      System.out.println( "real is " + real );
7      System.out.println( "log(real) is " + Math.log( real ) );
8      System.out.println( "log10(real) is " + Math.log10( real ) );
9      real = 64.0;
10     System.out.println( "real is " + real );
11     System.out.println( "sqrt(real) is " + Math.sqrt( real ) );
12     System.out.println( "cbrt(real) is " + Math.cbrt( real ) );
13     System.out.println( "exp(real) is " + Math.exp( real ) );
14     System.out.println( "abs(real) is " + Math.abs( real ) );
15     System.out.println( "signum(real) is " + Math.signum( real ) );
16     real = 95.789;
17     System.out.println( "real is " + real );
18     System.out.println( "ceil(real) is " + Math.ceil( real ) );
19     System.out.println( "round(real) is " + Math.round( real ) );
20     System.out.println( "floor(real) is " + Math.floor( real ) );
21   }
22 }
```

Rounding functions

# Result of One Parameter Methods

```
% java MathOneParameter
real is 10.0
log(real) is 2.302585092994046
log10(real) is 1.0
real is 64.0
sqrt(real) is 8.0
cbrt(real) is 4.0
exp(real) is 6.235149080811617E27
abs(real) is 64.0
signum(real) is 1.0
real is 95.789
ceil(real) is 96.0
round(real) is 96
floor(real) is 95.0
```

# Trigonometry

```
1  // examples of trigonometric functions
2  public class MathTrigonometry {
3    public static void main( String[] args ) {
4      double angle = - Math.PI / 6;
5      System.out.println( "angle is " + angle + " (in radian)" );
6      System.out.println( "sin is " + Math.sin( angle ) );
7      System.out.println( "cos is " + Math.cos( angle ) );
8      System.out.println( "tan is " + Math.tan( angle ) );
9      double value = - 0.5;
10     System.out.println( "value = " + value );
11     System.out.println( "asin/PI is "
12             + Math.asin( value ) / Math.PI );
13     System.out.println( "acos/PI is "
14             + Math.acos( value ) / Math.PI );
15     System.out.println( "atan/PI is "
16             + Math.atan( value ) / Math.PI );
17   }
18 }
```

sin, cos, tan

# Trigonometry

```
1  // examples of trigonometric functions
2  public class MathTrigonometry {
3    public static void main( String[] args ) {
4      double angle = - Math.PI / 6;
5      System.out.println( "angle is " + angle + " (in radian)" );
6      System.out.println( "sin is " + Math.sin( angle ) );
7      System.out.println( "cos is " + Math.cos( angle ) );
8      System.out.println( "tan is " + Math.tan( angle ) );
9      double value = - 0.5;
10     System.out.println( "value = " + value );
11     System.out.println( "asin/PI is "
12             + Math.asin( value ) / Math.PI );
13     System.out.println( "acos/PI is "
14             + Math.acos( value ) / Math.PI );
15     System.out.println( "atan/PI is "
16             + Math.atan( value ) / Math.PI );
17   }
18 }
```

asin, acos, atan

# Result of MathTrigonometry

```
% java MathTrigonometry
c = -0.5235987755982988 (in radian)
Math.sin(c) = -0.49999999999999994
Math.cos(c) = 0.8660254037844387
Math.tan(c) = -0.5773502691896257
d = -0.5
Math.asin(d)/Math.PI = -0.16666666666666669
Math.acos(d)/Math.PI = 0.6666666666666667
Math.atan(d)/Math.PI = -0.14758361765043326
```

# Two Parameter Methods

```
1  // examples of math functions with two parameters
2  public class MathTwoParameters {
3    public static void main( String[] args ) {
4      double real1 = 5.5, real2 = 12.0;
5      System.out.print( "real1 is " + real1 );
6      System.out.println( ", real2 is " + real2 );
7      System.out.println( "pow( real1, real2 ) is "
8              + Math.pow( real1, real2 ) );
9      System.out.println( "max( real1, real2 ) is "
10             + Math.max( real1, real2 ) );
11     System.out.println( "min( real1, real2 ) is "
12             + Math.min( real1, real2 ) );
13     int int1 = -3, int2 = -4;
14     System.out.print( "int1 is " + int1 );
15     System.out.println( ", int2 is " + int2 );
16     System.out.println( "pow( int1, int2 ) is "
17             + Math.pow( int1, int2 ) );
18     System.out.println( "max( int1, int2 ) is "
19             + Math.max( int1, int2 ) );
20     System.out.println( "min( int1, int2 ) is "
21             + Math.min( int1, int2 ) );
22   }
23 }
```

Double

# Two Parameter Methods

```
1  // examples of math functions with two parameters
2  public class MathTwoParameters {
3    public static void main( String[] args ) {
4      double real1 = 5.5, real2 = 12.0;
5      System.out.print( "real1 is " + real1 );
6      System.out.println( ", real2 is " + real2 );
7      System.out.println( "pow( real1, real2 ) is "
8              + Math.pow( real1, real2 ) );
9      System.out.println( "max( real1, real2 ) is "
10             + Math.max( real1, real2 ) );
11     System.out.println( "min( real1, real2 ) is "
12             + Math.min( real1, real2 ) );
13     int int1 = -3, int2 = -4;
14     System.out.print( "int1 is " + int1 );
15     System.out.println( ", int2 is " + int2 );
16     System.out.println( "pow( int1, int2 ) is "
17             + Math.pow( int1, int2 ) );
18     System.out.println( "max( int1, int2 ) is "
19             + Math.max( int1, int2 ) );
20     System.out.println( "min( int1, int2 ) is "
21             + Math.min( int1, int2 ) );
22   }
23 }
```

int

# Result of Two Parameter Methods

```
% java MathTwoParameters
real1 is 5.5, real2 is 12.0
pow( real1, real2 ) is 7.662178654104004E8
max( real1, real2 ) is 12.0
min( real1, real2 ) is 5.5
int1 is -3, int2 is -4
pow( int1, int2 ) is 0.012345679012345678
max( int1, int2 ) is -3
min( int1, int2 ) is -4
```

# The End