Passing Parameters to and from Methods; Class Math

Mitsu Ogihara

Department of Computer Science
University of Miami
Table of Contents

1. Passing Values to Methods
2. Receiving Value from a Method
3. What is Class Math?
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

```java
... public static void triangle() {
    int height = 10;
    for ( int posV = 1; posV <= height; posV ++ ) {
        for ( int posH = 1; posH <= posV; posH ++ ) {
            System.out.print( "#" );
        }
        System.out.println();
    }
}
public static void main( String[] args ) {
    int height = 5;
    triangle();
    height = 7;
    triangle();
}
...```

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

```java
public static void triangle( int height ) {
    for ( int posV = 1; posV <= height; posV ++ ) {
        for ( int posH = 1; posH <= posV; posH ++ ) {
            System.out.print( "#" );
        }
        System.out.println();
    }
}

public static void main( String[] args ) {
    int height = 5;
    triangle( height );
    height = 7;
    triangle( height );
}
```

...
General Parameter Specification

The parameter specification of a method takes the form of:

```java
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
When the method \texttt{triangle(height)} is called, the value of the variable \texttt{height} in the main method is stored in the variable \texttt{height} of \texttt{triangle} before the method starts operating.

The two methods have disjoint scopes and so any changes made in \texttt{triangle} do not affect the value of \texttt{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.
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

1. Passing Values to Methods
2. Receiving Value from a Method
3. What is Class Math?
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
  → 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

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

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

```java
public static double calculate( double weight, double height ) {
    double bmi = 703.0 * weight / ( height * height );
    return bmi;
}
```

The value calculation
The Calculation Method

```java
public static double calculate( double weight, double height ) {
    double bmi = 703.0 * weight / ( height * height );
    return bmi;
}
```

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

```java
public static void main( String[] args ) {
    double weight, height, bmi;
    System.out.println( "******* This is round 1" );
    weight = 150.0;
    height = 67.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
    System.out.println( "******* This is round 2" );
    weight = 170.0;
    height = 72.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
}
```

The variable declarations
Class BMIReturnValue

```java
public static void main( String[] args ) {
    double weight, height, bmi;
    System.out.println( "******** This is round 1" );
    weight = 150.0;
    height = 67.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
    System.out.println( "******** This is round 2" );
    weight = 170.0;
    height = 72.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
}
```

Assign values to weight and height.
public static void main( String[] args ) {
    double weight, height, bmi;
    System.out.println( "******** This is round 1" );
    weight = 150.0;
    height = 67.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
    System.out.println( "******** This is round 2" );
    weight = 170.0;
    height = 72.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
}

Receive the value from the calculation method
public static void main( String[] args ) {
    double weight, height, bmi;
    System.out.println( "******** This is round 1" );
    weight = 150.0;
    height = 67.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
    System.out.println( "******** This is round 2" );
    weight = 170.0;
    height = 72.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
}

Output the result
public static void main( String[] args ) {
    double weight, height, bmi;
    System.out.println( "******** This is round 1" );
    weight = 150.0;
    height = 67.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
    System.out.println( "******** This is round 2" );
    weight = 170.0;
    height = 72.0;
    bmi = calculate( weight, height );
    System.out.println( "weight is " + weight + " lb" );
    System.out.println( "height is " + height + " in" );
    System.out.println( " bmi is " + bmi );
}
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

```java
public class BMIMetric {
    //-- convert kilograms to pounds
    public static double kgToLb( double kgWeight ) {
        return 2.20462 * kgWeight;
    }

    //-- convert meters to inches
    public static double mToIn( double meter ) {
        return 39.3701 * meter;
    }

    //-- calculation for pounds and inches
    public static double calculate( double weight, double height ) {
        return 703.0 * weight / ( height * height );
    }

    //-- calculation for kilograms and meters
    public static double metricCalculate( double weight, double height ) {
        return calculate( kgToLb( weight ), mToIn( height ) );
    }
}
```

Kilogram-to-pound Conversion
Methods Can Be Used to Calculate Parameter Values

```java
public class BMIMetric {
    //-- convert kilograms to pounds
    public static double kgToLb( double kgWeight ) {
        return 2.20462 * kgWeight;
    }
    //-- convert meters to inches
    public static double mToIn( double meter ) {
        return 39.3701 * meter;
    }
    //-- calculation for pounds and inches
    public static double calculate( double weight, double height ) {
        return 703.0 * weight / ( height * height );
    }
    //-- calculation for kilograms and meters
    public static double metricCalculate( double weight, double height ) {
        return calculate( kgToLb( weight ), mToIn( height ) );
    }
}
```

Meter-to-inch Conversion
Methods Can Be Used to Calculate Parameter Values

```java
class BMIMetric {
    // convert kilograms to pounds
    public static double kgToLb( double kgWeight ) {
        return 2.20462 * kgWeight;
    }
    // convert meters to inches
    public static double mToIn( double meter ) {
        return 39.3701 * meter;
    }
    // calculation for pounds and inches
    public static double calculate( double weight, double height ) {
        return 703.0 * weight / ( height * height );
    }
    // calculation for kilograms and meters
    public static double metricCalculate( double weight, double height ) {
        return calculate( kgToLb( weight ), mToIn( height ) );
    }
}
```

BMI with pounds and inches
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

```java
//-- main method
public static void main( String[] args ) {
    double weight, height, bmi;
    weight = 65.5;
    height = 1.75;
    bmi = metricCalculate( weight, height );
    System.out.println( "weight is " + weight + " Kg" );
    System.out.println( "height is " + height + " m" );
    System.out.println( " bmi is " + bmi );
}
```

Declaration of the variables and assignment to them
The Rest of the Code

```java
//-- main method
public static void main( String[] args ) {
    double weight, height, bmi;
    weight = 65.5;
    height = 1.75;
    bmi = metricCalculate( weight, height );
    System.out.println( "weight is " + weight + " Kg" );
    System.out.println( "height is " + height + " m" );
    System.out.println( " bmi is " + bmi );
}
```

Calculation and output generation
# Table of Contents

1. Passing Values to Methods
2. Receiving Value from a Method
3. What is Class Math?
Math is a class that provides a number of useful mathematical constants 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

<table>
<thead>
<tr>
<th>Name</th>
<th>What it computes</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin</td>
<td>The sine of the parameter value (radian)</td>
</tr>
<tr>
<td>cos</td>
<td>The cosine of the parameter value (radian)</td>
</tr>
<tr>
<td>tan</td>
<td>The tangent of the parameter value (radian)</td>
</tr>
<tr>
<td>asin</td>
<td>The inverse of sine, return value in $[-\frac{\pi}{2}, \frac{\pi}{2}]$</td>
</tr>
<tr>
<td>acos</td>
<td>The inverse of sine, return value in $[0, \pi]$</td>
</tr>
<tr>
<td>atan</td>
<td>The inverse of sine, return value in $[-\frac{\pi}{2}, \frac{\pi}{2}]$</td>
</tr>
<tr>
<td>sqrt</td>
<td>The square root</td>
</tr>
<tr>
<td>cbrt</td>
<td>The cubic root</td>
</tr>
<tr>
<td>log</td>
<td>The natural logarithm</td>
</tr>
<tr>
<td>log10</td>
<td>The logarithm base 10</td>
</tr>
<tr>
<td>signum</td>
<td>The sign of the number, -1.0, 0.0, or +1.0</td>
</tr>
<tr>
<td>exp</td>
<td>The exponential function base the natural log.</td>
</tr>
</tbody>
</table>
Methods with One Parameter (cont’d)

Both the parameter type and the return type are double

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceil</td>
<td>The smallest whole number that is $\geq$ parameter</td>
</tr>
<tr>
<td>floor</td>
<td>The largest whole number $\leq$ parameter</td>
</tr>
<tr>
<td>round</td>
<td>The rounded whole number, as an int</td>
</tr>
<tr>
<td>abs</td>
<td>The absolute value</td>
</tr>
</tbody>
</table>
Math Methods with Two Parameters

<table>
<thead>
<tr>
<th>max</th>
<th>The maximum of the two numbers given as parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>The minimum of the two numbers given as parameters</td>
</tr>
</tbody>
</table>

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

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

Math.PI
No Parameters

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

Math.E
No Parameters

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

Math.random
Result of No Parameter Methods

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

```java
// Math functions requiring 1 parameter, excluding trigonometry
public class MathOneParameter{
    public static void main( String[] args ) {
        double real;
        real = 10.0;
        System.out.println( "real is " + real );
        System.out.println( "log(real) is " + Math.log( real ) );
        System.out.println( "log10(real) is " + Math.log10( real ) );
        real = 64.0;
        System.out.println( "real is " + real );
        System.out.println( "sqrt(real) is " + Math.sqrt( real ) );
        System.out.println( "cbrt(real) is " + Math.cbrt( real ) );
        System.out.println( "exp(real) is " + Math.exp( real ) );
        System.out.println( "abs(real) is " + Math.abs( real ) );
        System.out.println( "signum(real) is " + Math.signum( real ) );
        real = 95.789;
        System.out.println( "real is " + real );
        System.out.println( "ceil(real) is " + Math.ceil( real ) );
        System.out.println( "round(real) is " + Math.round( real ) );
        System.out.println( "floor(real) is " + Math.floor( real ) );
    }
}
```

Analytic functions requiring positive parameter
One Parameter Methods

```java
// Math functions requiring 1 parameter, excluding trigonometry
public class MathOneParameter{
    public static void main( String[] args ) {
        double real;
        real = 10.0;
        System.out.println( "real is " + real );
        System.out.println( "log(real) is " + Math.log( real ) );
        System.out.println( "log10(real) is " + Math.log10( real ) );
        real = 64.0;
        System.out.println( "real is " + real );
        System.out.println( "sqrt(real) is " + Math.sqrt( real ) );
        System.out.println( "cbrt(real) is " + Math.cbrt( real ) );
        System.out.println( "exp(real) is " + Math.exp( real ) );
        System.out.println( "abs(real) is " + Math.abs( real ) );
        System.out.println( "signum(real) is " + Math.signum( real ) );
        real = 95.789;
        System.out.println( "real is " + real );
        System.out.println( "ceil(real) is " + Math.ceil( real ) );
        System.out.println( "round(real) is " + Math.round( real ) );
        System.out.println( "floor(real) is " + Math.floor( real ) );
    }
}
```

Other analytic functions
One Parameter Methods

```java
// Math functions requiring 1 parameter, excluding trigonometry
public class MathOneParameter{
    public static void main( String[] args ) {
        double real;
        real = 10.0;
        System.out.println( "real is " + real );
        System.out.println( "log(real) is " + Math.log( real ) );
        System.out.println( "log10(real) is " + Math.log10( real ) );
        real = 64.0;
        System.out.println( "real is " + real );
        System.out.println( "sqrt(real) is " + Math.sqrt( real ) );
        System.out.println( "cbrt(real) is " + Math.cbrt( real ) );
        System.out.println( "exp(real) is " + Math.exp( real ) );
        System.out.println( "abs(real) is " + Math.abs( real ) );
        System.out.println( "signum(real) is " + Math.signum( real ) );
        real = 95.789;
        System.out.println( "real is " + real );
        System.out.println( "ceil(real) is " + Math.ceil( real ) );
        System.out.println( "round(real) is " + Math.round( real ) );
        System.out.println( "floor(real) is " + Math.floor( real ) );
    }
}
```

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

```java
// examples of trigonometric functions
public class MathTrigonometry {
    public static void main( String[] args ) {
        double angle = - Math.PI / 6;
        System.out.println( "angle is " + angle + " (in radian)" );
        System.out.println( "sin is " + Math.sin( angle ) );
        System.out.println( "cos is " + Math.cos( angle ) );
        System.out.println( "tan is " + Math.tan( angle ) );
        double value = - 0.5;
        System.out.println( "value = " + value );
        System.out.println( "asin/PI is "
            + Math.asin( value ) / Math.PI );
        System.out.println( "acos/PI is "
            + Math.acos( value ) / Math.PI );
        System.out.println( "atan/PI is "
            + Math.atan( value ) / Math.PI );
    }
}
```

sin, cos, tan
Trigonometry

```java
// examples of trigonometric functions
public class MathTrigonometry {
    public static void main( String[] args ) {
        double angle = - Math.PI / 6;
        System.out.println( "angle is " + angle + " (in radian)" );
        System.out.println( "sin is " + Math.sin( angle ) );
        System.out.println( "cos is " + Math.cos( angle ) );
        System.out.println( "tan is " + Math.tan( angle ) );
        double value = - 0.5;
        System.out.println( "value = " + value );
        System.out.println( "asin/PI is "
            + Math.asin( value ) / Math.PI );
        System.out.println( "acos/PI is "
            + Math.acos( value ) / Math.PI );
        System.out.println( "atan/PI is "
            + Math.atan( value ) / Math.PI );
    }
}
```

asin, acos, atan
Result of MathTrigonometry

```java
% 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

```java
// examples of math functions with two parameters
public class MathTwoParameters {
    public static void main(String[] args) {
        double real1 = 5.5, real2 = 12.0;
        System.out.print("real1 is " + real1);
        System.out.println(" , real2 is " + real2);
        System.out.println(" pow(real1, real2 ) is " + Math.pow(real1, real2));
        System.out.println(" max(real1, real2 ) is " + Math.max(real1, real2));
        System.out.println(" min(real1, real2 ) is " + Math.min(real1, real2));

        int int1 = -3, int2 = -4;
        System.out.print("int1 is " + int1);
        System.out.println(" , int2 is " + int2);
        System.out.println(" pow(int1, int2 ) is " + Math.pow(int1, int2));
        System.out.println(" max(int1, int2 ) is " + Math.max(int1, int2));
        System.out.println(" min(int1, int2 ) is " + Math.min(int1, int2));
    }
}
```
Two Parameter Methods

```java
// examples of math functions with two parameters
public class MathTwoParameters {
    public static void main( String[] args ) {
        double real1 = 5.5, real2 = 12.0;
        System.out.print( "real1 is " + real1 );
        System.out.println( " , real2 is " + real2 );
        System.out.println( "pow( real1, real2 ) is "
            + Math.pow( real1, real2 ) );
        System.out.println( "max( real1, real2 ) is "
            + Math.max( real1, real2 ) );
        System.out.println( "min( real1, real2 ) is "
            + Math.min( real1, real2 ) );

        int int1 = -3, int2 = -4;
        System.out.print( "int1 is " + int1 );
        System.out.println( " , int2 is " + int2 );
        System.out.println( "pow( int1, int2 ) is "
            + Math.pow( int1, int2 ) );
        System.out.println( "max( int1, int2 ) is "
            + Math.max( int1, int2 ) );
        System.out.println( "min( int1, int2 ) is "
            + Math.min( int1, int2 ) );
    }
}
```
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