

Building Java Programs

Chapter 3

Lecture 3-2: Return values, `Math`, and `double`

reading: 3.2, 2.1 - 2.2

Java's `Math` class

Method name	Description
<code>Math.abs (value)</code>	absolute value
<code>Math.ceil (value)</code>	rounds up
<code>Math.floor (value)</code>	rounds down
<code>Math.log10 (value)</code>	logarithm, base 10
<code>Math.max (value1, value2)</code>	larger of two values
<code>Math.min (value1, value2)</code>	smaller of two values
<code>Math.pow (base, exp)</code>	<i>base</i> to the <i>exp</i> power
<code>Math.random ()</code>	random double between 0 and 1
<code>Math.round (value)</code>	nearest whole number
<code>Math.sqrt (value)</code>	square root
<code>Math.sin (value)</code>	sine/cosine/tangent of an angle in radians
<code>Math.cos (value)</code>	
<code>Math.tan (value)</code>	
<code>Math.toDegrees (value)</code>	convert degrees to radians and back
<code>Math.toRadians (value)</code>	

Constant	Description
<code>Math.E</code>	2.7182818...
<code>Math.PI</code>	3.1415926...

No output?

- Simply calling these methods produces no visible result.

```
public class TestMath {  
    public static void main(String[] args) {  
        Math.pow(3, 4); // no output!  
    }  
}
```

3

No output!

- Math method calls use a feature called *return values* that cause them to be treated as expressions.
- The program runs the method, computes the answer, and then "replaces" the call with its computed result value.

```
• Math.pow(3, 4); // no output  
  81.0; // no output
```

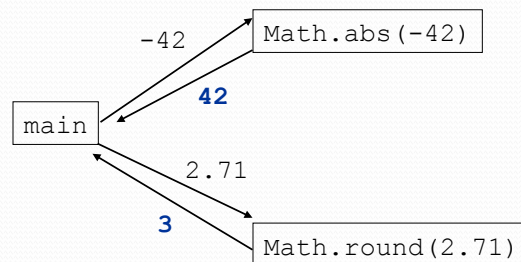
- To see the result, we must print it or store it in a variable.

```
• double result = Math.pow(3, 4);  
• System.out.println(result); // 81.0
```

4

Return

- **return:** To send out a value as the result of a method.
 - The opposite of a parameter
 - Parameters send information *in* from the caller to the method.
 - Return values send information *out* from a method to its caller.
 - A call to the method can be used as part of an expression.



5

Why return and not print?

- It might seem more useful for the `Math` methods to print their results rather than returning them. Why don't they?

- **Answer:** Returning is more flexible than printing.

- We can compute several things before printing:

```
double pow1 = Math.pow(3, 4);  
double pow2 = Math.pow(10, 6);  
System.out.println("Powers are " + pow1 + " and " + pow2);
```

- We can combine the results of many computations:

```
double k = 13 * Math.pow(3, 4) + 5 - Math.sqrt(17.8);
```

6

Math questions

- Evaluate the following expressions:
 - `Math.abs(-1.23)`
 - `Math.pow(3, 2)`
 - `Math.pow(10, -2)`
 - `Math.sqrt(121.0) - Math.sqrt(256.0)`
 - `Math.round(Math.PI) + Math.round(Math.E)`
 - `Math.ceil(6.022) + Math.floor(15.9994)`
 - `Math.abs(Math.min(-3, -5))`
- `Math.max` and `Math.min` can be used to bound numbers. Consider an `int` variable named `age`.
 - What statement would replace negative ages with 0?
 - What statement would cap the maximum age to 40?

7

Incompatible types

- Some `Math` methods return `double` or other non-`int` types.

```
int x = Math.pow(10, 3); // ERROR: incompatible types
```
- What if you wanted to store a `double` in an `int` variable? (maybe you don't care about the decimal part)

8

Type casting

- **type cast:** A conversion from one type to another.
 - To promote an `int` into a `double` to get exact division from /
 - To truncate a `double` from a real number to an integer

- Syntax:

(*<type>*) *<expression>*

Examples:

```
double result = (double) 19 / 5;      // 3.8
int result2 = (int) result;          // 3
int x = (int) Math.pow(10, 3);       // 1000
```

9

More about type casting

- Type casting has high precedence and only casts the item immediately next to it.

- `double x = (double) 1 + 1 / 2; // 1.0`
- `double y = 1 + (double) 1 / 2; // 1.5`

- You can use parentheses to force evaluation order.

- `double average = (double) (a + b + c) / 3;`

- A conversion to `double` can be achieved in other ways.

- `double average = 1.0 * (a + b + c) / 3;`

10

Returning a value

```
public static <type> <name> (<parameters>) {  
    <statement(s)>;  
    ...  
    return <expression>;  
}
```

- Example:

```
// Returns the slope of the line between the given points.  
public static double slope(int x1, int y1, int x2, int y2) {  
    double dy = y2 - y1;  
    double dx = x2 - x1;  
    return dy / dx;  
}
```

- slope(5, 11, 1, 3) returns 2.0

11

Return examples

```
// Converts degrees Fahrenheit to Celsius.  
public static double fToC(double degreesF) {  
    double degreesC = 5.0 / 9.0 * (degreesF - 32);  
    return degreesC;  
}
```

```
// Computes triangle hypotenuse length given its side lengths.  
public static double hypotenuse(int a, int b) {  
    double c = Math.sqrt(a * a + b * b);  
    return c;  
}
```

- You can shorten the examples by returning an expression:

```
public static double fToC(double degreesF) {  
    return 5.0 / 9.0 * (degreesF - 32);  
}
```

12

Common error: Not storing

- Many students incorrectly think that a `return` statement sends a variable's name back to the calling method.

```
public static void main(String[] args) {
    slope(0, 0, 6, 3);
    System.out.println("The slope is " + result);
}
// ERROR: result not defined

public static double slope(int x1, int y1, int x2, int y2) {
    double dy = y2 - y1;
    double dx = x2 - x1;
    double result = dy / dx;
    return result;
}
```

13

Fixing the common error

- Instead, returning sends the variable's *value* back.
 - The returned value must be stored into a variable or used in an expression to be useful to the caller.

```
public static void main(String[] args) {
    double s = slope(0, 0, 6, 3);
    System.out.println("The slope is " + s);
}

public static double slope(int x1, int y1, int x2, int y2) {
    double dy = y2 - y1;
    double dx = x2 - x1;
    double result = dy / dx;
    return result;
}
```

14

Common error variation

- Particularly confusing is conflating the return variable with a variable in the calling method.
 - Your program will compile, but you won't get the right result!

```
public class ReturnExample {
    public static void main(String[] args) {
        int x = 1;
        addOne(x);
        System.out.println("x = " + x);
    }

    public static int addOne(int x) {
        x = x + 1;
        return x;
    }
}
```

15

Don't ignore the return value!

- Just because the return variable in the called method has the same name as the variable in the calling method, they are **NOT** the same. Think scope!

```
public class ReturnExample {
    public static void main(String[] args) {
        int x = 1;
        addOne(x);
        System.out.println("x = " + x);
    }

    public static int addOne(int x) {
        x = x + 1;
        return x;
    }
}

public class ReturnExample {
    public static void main(String[] args) {
        int x = 1;
        x = addOne(x);
        System.out.println("x = " + x);
    }

    public static int addOne(int x) {
        x = x + 1;
        return x;
    }
}
```

16

Exercise

- In physics, the *displacement* of a moving body represents its change in position over time while accelerating.
 - Given initial velocity v_0 in m/s, acceleration a in m/s^2 , and elapsed time t in s, the displacement of the body is:
 - Displacement = $v_0 t + \frac{1}{2} a t^2$
- Write a method `displacement` that accepts v_0 , t , and a and computes and returns the change in position.
 - Example: `displacement(3.0, 4.0, 5.0)` returns 52.0

17

Exercise solution

```
public static double displacement(double v0, double t, double a) {  
    double d = v0 * t + 0.5 * a * Math.pow(t, 2);  
    return d;  
}
```

18

Exercise

- If you drop two balls, which will hit the ground first?
 - Ball 1: height of 600m, initial velocity = 25 m/sec downward
 - Ball 2: height of 500m, initial velocity = 15 m/sec downward
- Write a program that determines how long each ball takes to hit the ground (and draws each ball falling).
- Total time is based on the force of gravity on each ball.
 - Acceleration due to gravity $\cong 9.81 \text{ m/s}^2$, downward
 - Displacement = $v_0 t + \frac{1}{2} a t^2$

19

Ball solution

```
// Simulates the dropping of two balls from various heights.
import java.awt.*;

public class Balls {
    public final static int PANEL_HEIGHT = 600;

    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(600, 600);
        Graphics g = panel.getGraphics();

        int ball1x = 100,    initialBall1y = 600,    v01 = 25;
        int ball2x = 200,    initialBall2y = 500,    v02 = 15;

        // draw the balls at each time increment
        for (double t = 0; t <= 10.0; t = t + 0.1) {
            double height1 = initialBall1y - displacement(v01, t, 9.81);
            g.fillOval(ball1x, PANEL_HEIGHT - (int)height1, 10, 10);
            double height2 = initialBall2y - displacement(v02, t, 9.81);
            g.fillOval(ball2x, PANEL_HEIGHT - (int)height2, 10, 10);

            panel.sleep(50);    // pause for 50 ms
            panel.clear();
        }
    }
    ...
}
```

20