Building Java Programs

Chapter 3
Lecture 3-2: Return values; `double`; `Point`

reading: 3.2, 3.3, 3.5

self-check: #7-11

exercises: #4-6

videos: Ch. 3 #2, 4

Java's `Math` class

Method name	Description
`Math.abs(`*value*`)`	absolute value
`Math.ceil(`*value*`)`	rounds up
`Math.floor(`*value*`)`	rounds down
`Math.log10(`*value*`)`	logarithm, base 10
`Math.max(`*value1*`, `*value2*`)`	larger of two values
`Math.min(`*value1*`, `*value2*`)`	smaller of two values
`Math.pow(`*base*`, `*exp*`)`	*base* to the *exp* power
`Math.random()`	random `double` between 0 and 1
`Math.round(`*value*`)`	nearest whole number
`Math.sqrt(`*value*`)`	square root

`Math.sin(`*value*`)`
`Math.cos(`*value*`)`
`Math.tan(`*value*`)`
	sine/cosine/tangent of an angle in radians

`Math.toDegrees(`*value*`)` `Math.toRadians(`*value*`)`	convert degrees to radians and back

Constant	Description
`Math.E`	2.7182818...
`Math.PI`	3.1415926...

Calling `Math` methods

```
Math.methodName(parameters)
```

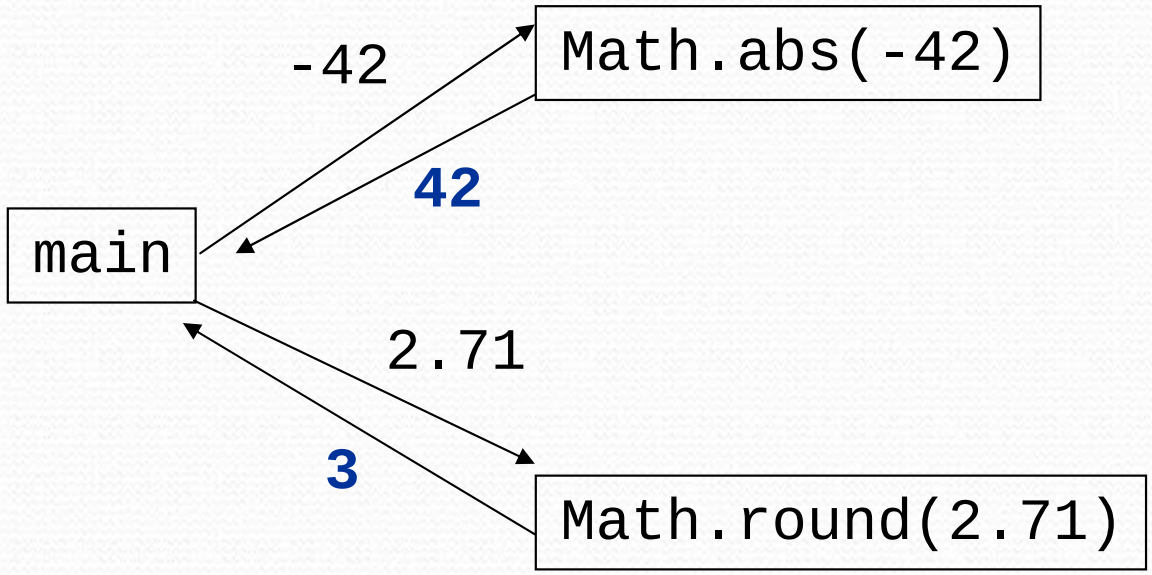
- Examples:

```
double squareRoot = Math.sqrt(121.0);
System.out.println(squareRoot);             // 11.0

int absoluteValue = Math.abs(-50);
System.out.println(absoluteValue);          // 50

System.out.println(Math.min(3, 7) + 2);     // 5
```

- The `Math` methods do not print to the console.
  - Each method produces ("returns") a numeric result.
  - The results are used as expressions (printed, stored, etc.).

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.



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?

Quirks of real numbers

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

```
int x = Math.pow(10, 3);   // ERROR: incompat. types
```

- Some `double` values print poorly (too many digits).

```
double result = 1.0 / 3.0;
System.out.println(result);    // 0.3333333333333
```

- The computer represents `double`s in an imprecise way.

```
System.out.println(0.1 + 0.2);
```

  - Instead of 0.3, the output is `0.30000000000000004`

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
```

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`
  - `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;`

Exercise

- If you drop three balls, which will hit the ground first?
  - Ball 1: height of 600m, initial velocity = 25 m/sec downward
  - Ball 2: height of 400m, initial velocity = 0
  - Ball 3: height of 500m, initial velocity = 15 m/sec downward
- Write a program that determines how long each ball takes to hit the ground (or draws each ball falling).
- Total time is based on the force of gravity on each ball.
  - Acceleration due to gravity $\cong 9.81$ m/s^2, downward
  - Displacement = $v_0 t + \frac{1}{2} a t^2$

Returning a value

```
public static type name(parameters) {
    statements;
    ...
    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(1, 3, 5, 11)` returns `2.0`

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);
}
```

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

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

Point objects

```
import java.awt.*;
...
Point p1 = new Point(5, -2);
Point p2 = new Point();          // the origin (0, 0)
```

- Data:

Name	Description
x	the point's x-coordinate
y	the point's y-coordinate

- Methods:

Name	Description
`setLocation(`**`x`**`, `**`y`**`)`	sets the point's x and y to the given values
`translate(`**`dx`**`, `**`dy`**`)`	adjusts the point's x and y by the given amounts
`distance(`**`p`**`)`	how far away the point is from point *p*

Using Point

```
public class PointMain {
    public static void main(String[] args) {
        // create two Point objects
        Point p1 = new Point();
        p1.y = 8;
        Point p2 = new Point(5, 7);
        p2.x = 4;

        System.out.println(p1.x + ", " + p1.y);   // 0, 8

        // move p2 and then print it
        p2.x += 2;
        p2.y++;
        System.out.println(p2.x + ", " + p2.y);   // 6, 8

        // move p1 and then print it
        p1.translate(4, -5);
        System.out.println(p1.x + ", " + p1.y);   // 4, -3
    }
}
```

Ball solution

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

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

        Point ball1 = new Point(100, (600 - 600));  // height of 600
        Point ball2 = new Point(200, (600 - 400));  // height of 400
        Point ball3 = new Point(300, (600 - 500));  // height of 500

        // draw the balls at each time increment
        for (double t = 0; t <= 10.0; t = t + 0.1) {
            drawBall(g, ball1, 25, t);   // initial velocity of 25
            drawBall(g, ball2,  0, t);   // initial velocity of  0
            drawBall(g, ball3, 15, t);   // initial velocity of 15
            panel.sleep(50);   // pause for 50 ms
        }
    }

    ...
```

Ball solution, cont'd.

```
    ...

    // Draws the given ball point with the given initial velocity
    // after the given amount of time has elapsed.
    public static void drawBall(Graphics g, Point ball, double v0, double t) {
        double disp = displacement(v0, t, 9.81);
        g.fillOval(ball.x, ball.y + (int) disp, 10, 10);
    }

    // Computes the displacement of a moving ball
    // with the given initial velocity, acceleration, and time.
    // displacement = v0 t + 1/2 a t^2
    public static double displacement(double v0, double t, double a) {
        double d = v0 * t + 0.5 * a * Math.pow(t, 2);
        return d;
    }
}
```