Building Java Programs

Chapter 3
Lecture 3-2: Return; double; System.out.printf

reading: 3.2, 3.5, 4.4
videos: Ch. 3 #2, 4

Return Values

reading: 3.2

self-check: #7-11
exercises: #4-6
videos: Ch. 3 #2
Java's Math class

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

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.E</td>
<td>2.7182818...</td>
</tr>
<tr>
<td>Math.PI</td>
<td>3.1415926...</td>
</tr>
</tbody>
</table>

Calling Math methods

Math. **methodName** (parameters)

- Examples:
  ```java
  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.

```
main
    Math.abs(42)   -42
    Math.round(2.71)   2.71
        42
    Math.round(2.71)   3
```

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.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?
Returning a value

```java
public static type name(parameters) {
    statements;
    ...
    return expression;
}
```

- Example:
  ```java
  // 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;
  }
  ```

Return examples

```java
// Converts 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:
  ```java
  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.

```java
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.

```java
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;
}
```
Quirks of real numbers

• Some Math methods return double or other non-int types.
  ```java
  int x = Math.pow(10, 3); // ERROR: incompat. types
  ```

• Some double values print poorly (too many digits).
  ```java
  double result = 1.0 / 3.0;
  System.out.println(result); // 0.3333333333333333
  ```

• The computer represents doubles in an imprecise way.
  ```java
  System.out.println(0.1 + 0.2);
  ```
  Instead of 0.3, the output is 0.3000000000000004

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:
  ```java
  (type) expression
  ```

Examples:
  ```java
  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;

System.out.printf

*an advanced command for printing formatted text*

System.out.printf("format string", parameters);

- A format string contains placeholders to insert parameters into it:
  - %d an integer
  - %f a real number
  - %s a string

- Example:
  int x = 3;
  int y = 2;
  System.out.printf("(%d, %d)\n", x, y); // (3, 2)
A placeholder can specify the parameter's width or precision:

- `%8d` an integer, 8 characters wide, right-aligned
- `%8d` an integer, 8 characters wide, left-aligned
- `%4f` a real number, 4 characters after decimal
- `%6.2f` a real number, 6 characters wide, 2 after decimal

Examples:
```
int age = 45;
double gpa = 1.2345678;
System.out.printf("%-8d %4f\n", age, gpa);
System.out.printf("%8.3f %.1f %.5f", gpa, gpa, gpa);
```

Output:
```
45     1.23
1.234 1.2 1.23457
```