Java Numbers

V102

by Dave Slemon

Data Types

- data types: `int`, `double`, `boolean`, `char` `byte`, `short`, `long`, `float`
- `boolean` has values `true` and `false`
- declarations look like this,
  - `double x, y;`
  - `int count = 0;`

Expressions

- Assignment statements you can use `=`, `+=`, `*=` etc.
- Arithmetic uses the familiar `+ - * / %`
- Java also has `++` and `--`
- Java has boolean operators `&& || !`
- Java has comparisons `< <= == != >= >`

Comments Within the Code in Java

```
/* This kind of comment
can span multiple lines
*/


// This kind of comment, concludes at the end of the line


/**
  * This kind of comment is a special
  * javadoc style comment
  */
```

Data Type Ranges

5

Data Type	Characteristics	Range
`byte`	8 bit signed integer	-128 to 127
`short`	16 bit signed integer	-32768 to 32767
`int`	32 bit signed integer	-2,147,483,648 to 2,147,483,647
`long`	64 bit signed integer	-9,223,372,036,854,775,808 to-9,223,372,036,854,775,807
`float`	32 bit floating point number	± 1.4E-45 to ± 3.4028235E+38
`double`	64 bit floating point number	± 4.9E-324 to ± 1.7976931348623157E+308
`boolean`	`true` or `false`	NA, note Java booleans cannot be converted to or from other types
`char`	16 bit, Unicode	Unicode character, `\u0000` to `\uFFFF` Can mix with integer types

Math Class's Static Methods

Examples:

```
double squareRoot = Math.sqrt(144.0);
System.out.println(squareRoot);            // 12.0

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 ("returns") a numeric result.
- The results are used in expressions

7

Java's `Math` class

Method name	Description
`Math.abs(`*value*`)`	absolute value
`Math.ceil(`*value*`)`	moves up to ceiling
`Math.floor(`*value*`)`	moves down to floor
`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.rint(`*value*`)`	Round int, 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...

Real number Issues

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

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

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

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

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

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

Instead of 0.3, the output is `0.30000000000000004`

Casting

type cast: a conversion from one type to another.

- to promote an `int` into a `double` for floating point division
- 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 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;
```