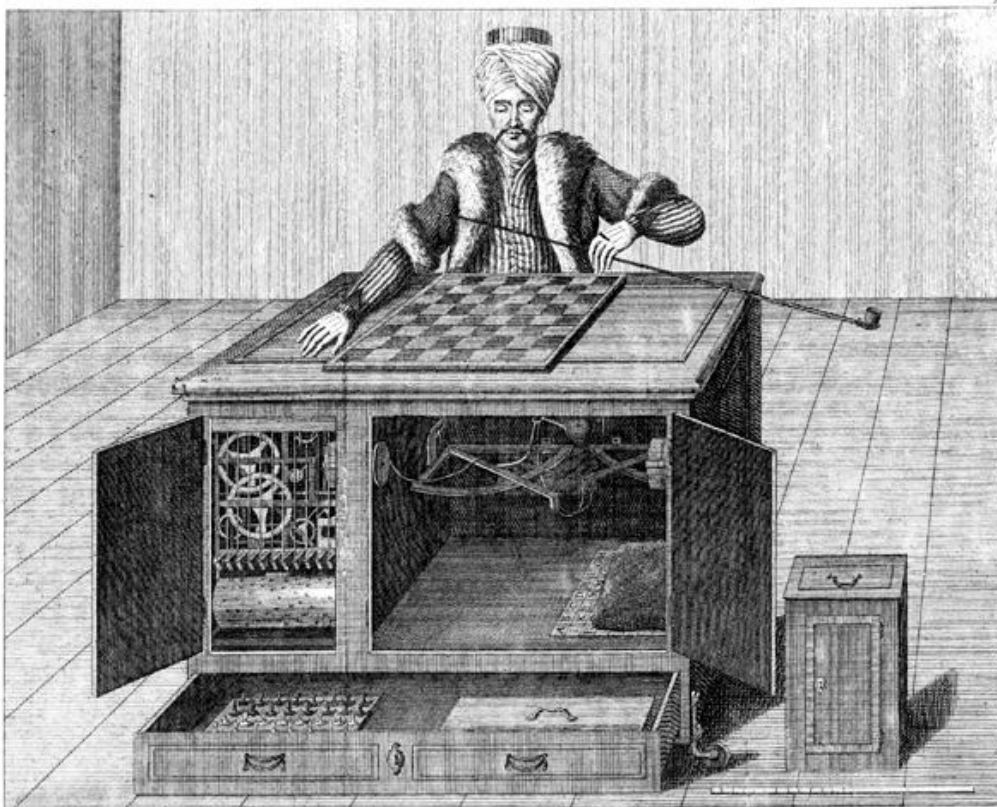


CSE 142, Summer 2015

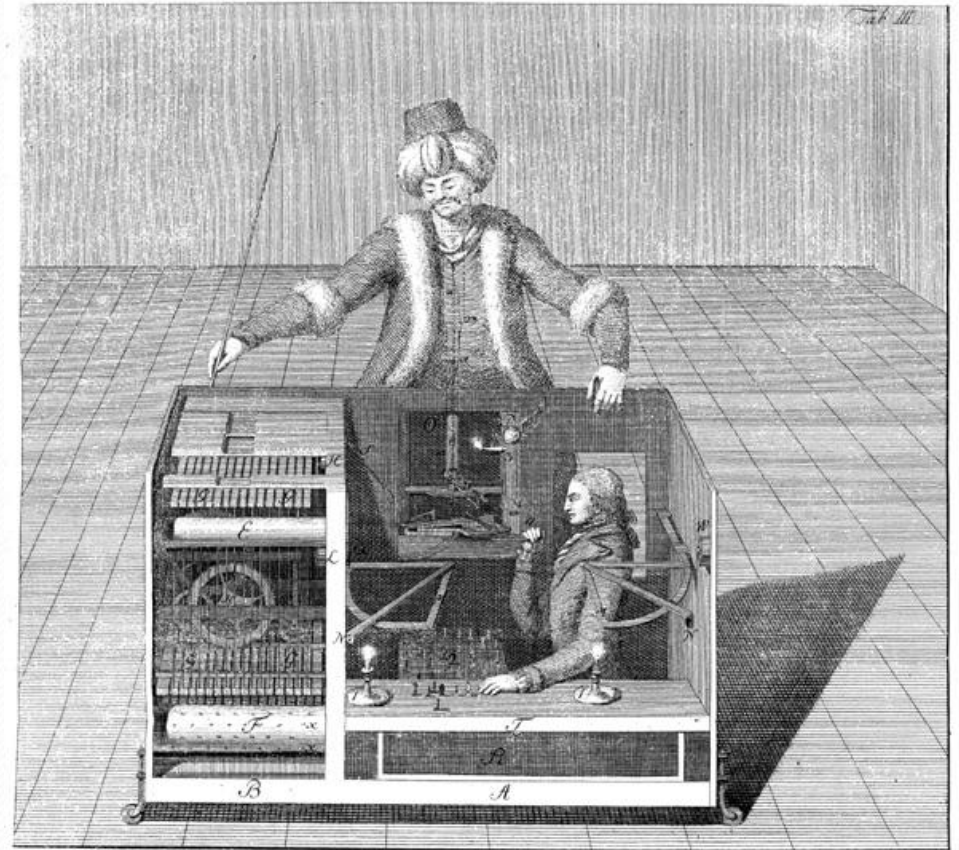
Lecture 2: Static Methods
Expressions

reading: 1.4 – 2.1

The Mechanical Turk

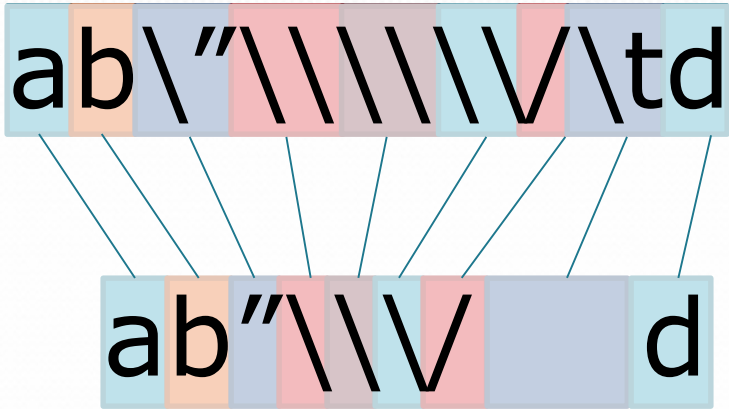


W. de Kempelen del. *Ch. a Mechel exend. Basilea.* P. G. Pinty fecit.
Der Schach-Spieler wie er vor dem Spiele zu sehen wird vor dem L. Loucard's Chess, tel qu'on le montre avant le jeu par devant.



Escape Characters

```
System.out.println("ab\"\\\"\\\"\\V\\td");
```



The diagram shows the source code string "ab\"\\\"\\\"\\V\\td"; with colored boxes highlighting characters. Lines connect these boxes to the output string "ab\"\\\"\\V\\td". The boxes are: 'a' (light blue), 'b' (orange), '\"' (light blue), '\\\"' (red), '\\\"' (red), '\\V' (light blue), '\\' (red), 'd' (light blue). The output string has boxes: 'a' (light blue), 'b' (orange), '\"' (light blue), '\\\"' (red), '\\V' (light blue), '\\' (red), 'd' (light blue).

Output:

```
ab\"\\\"\\V\\td
```

Algorithms

- **algorithm**: A list of steps for solving a problem.
- Example algorithm: "Bake sugar cookies"
 1. Mix the dry ingredients.
 2. Cream the butter and sugar.
 3. Beat in the eggs.
 4. Stir in the dry ingredients.
 5. Set the oven temperature.
 6. Set the timer for 10 minutes.
 7. Place the cookies into the oven.
 8. Allow the cookies to bake.
 9. Spread frosting and sprinkles onto the cookies.



Style

Correct Recipe with Bad Style:

- Recipe Works Correctly
- Hard to understand or modify

5. Set the oven temperature. 8. Spread frosting and sprinkles onto the cookies. 6. Set the timer for 10 minutes. 3. Beat in the eggs. 1. Mix the dry ingredients. 2. Cream the butter and sugar. 7. Allow the cookies to bake. 4. Stir in the dry ingredients. 6.5. Place the cookies into the oven.

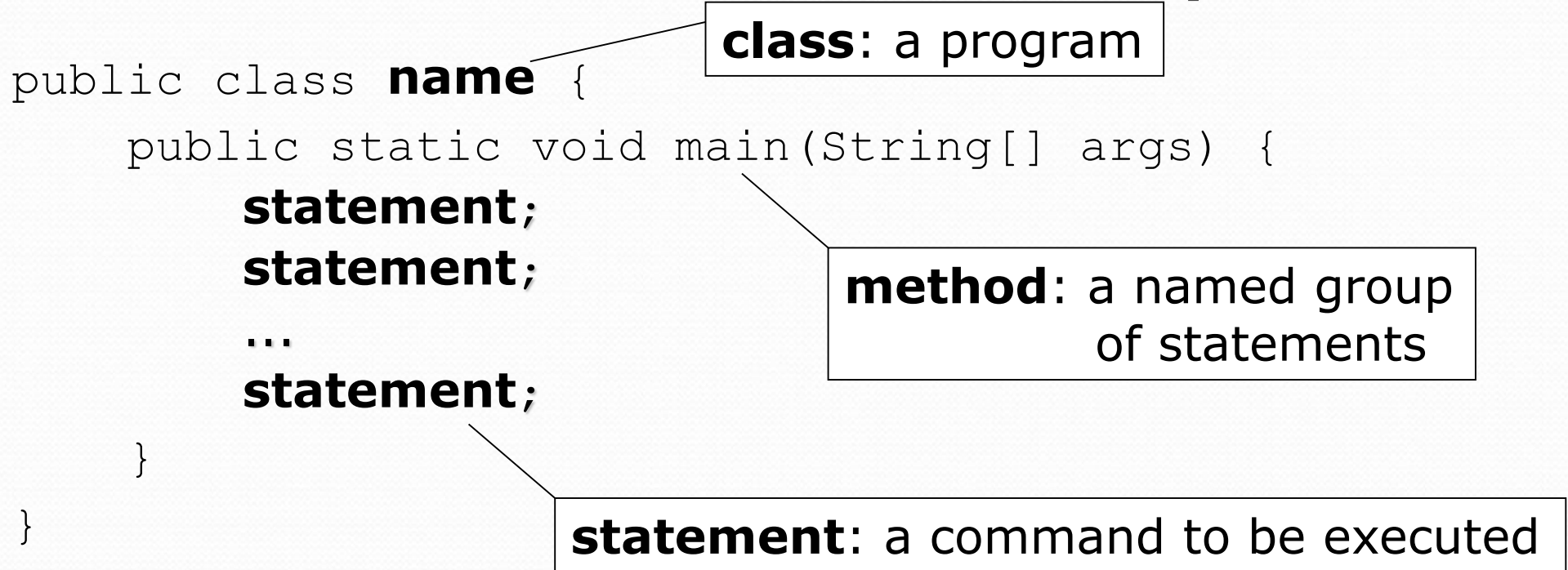


Style

- **Better Style:** Split into coherent tasks.
 - 1 Make the batter.
 - Mix the dry ingredients.
 - Cream the butter and sugar.
 - Beat in the eggs.
 - Stir in the dry ingredients.
 - 2 Bake the cookies.
 - Set the oven temperature.
 - Set the timer for 10 minutes.
 - Place the cookies into the oven.
 - Allow the cookies to bake.
 - 3 Decorate the cookies.
 - Mix the ingredients for the frosting.
 - Spread frosting and sprinkles onto the cookies.

...

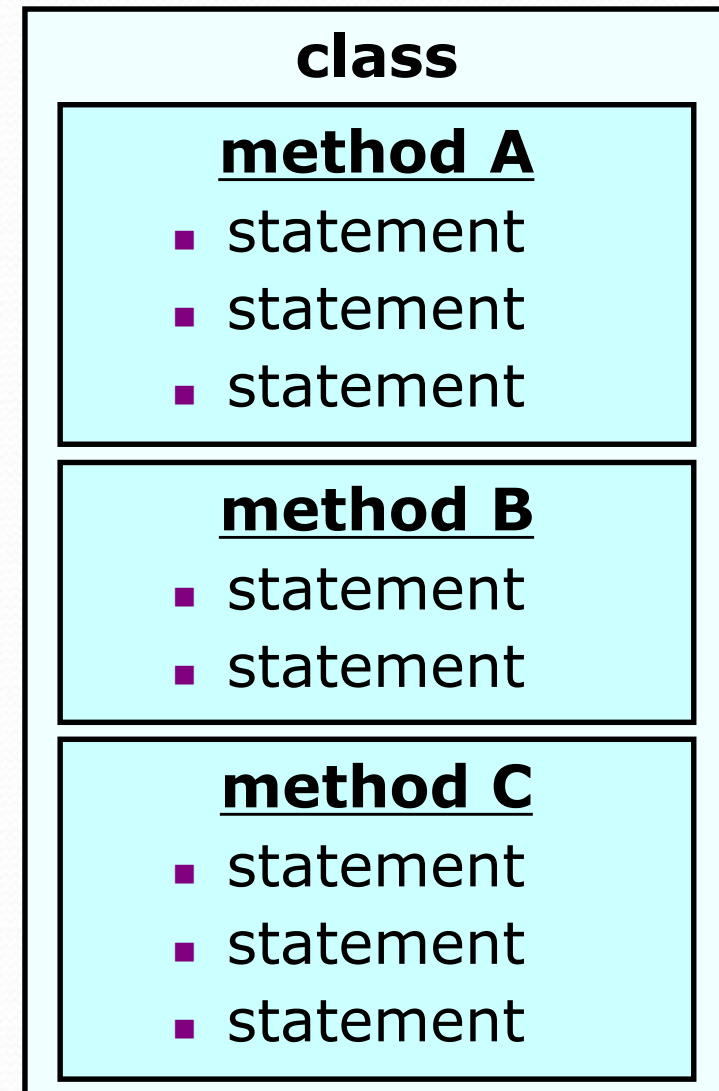
Recall: structure, syntax



- Every executable Java program consists of a **class**,
 - that contains a **method** named `main`,
 - that contains the **statements** (commands) to be executed.

Static methods

- **static method:** A named group of statements.
 - denotes the *structure* of a program
 - eliminates *redundancy* by code reuse
- **procedural decomposition:**
dividing a problem into methods
- Writing a static method is like adding a new command to Java.



Declaring a method

Gives your method a name so it can be executed

- Syntax:

```
public static void name() {  
    statement;  
    statement;  
    ...  
    statement;  
}
```

- Example:

```
public static void printWarning() {  
    System.out.println("This product causes cancer");  
    System.out.println("in lab rats and humans.");  
}
```

Calling a method

Executes the method's code

- Syntax:

name ();

- You can call the same method many times if you like.

- Example:

```
printWarning();
```

- Output:

```
This product causes cancer  
in lab rats and humans.
```

Methods calling methods

```
public class MethodsExample {
    public static void main(String[] args) {
        message1();
        message2();
        System.out.println("Done with main.");
    }

    public static void message1() {
        System.out.println("This is message1.");
    }

    public static void message2() {
        System.out.println("This is message2.");
        message1();
        System.out.println("Done with message2.");
    }
}
```

- **Output:**

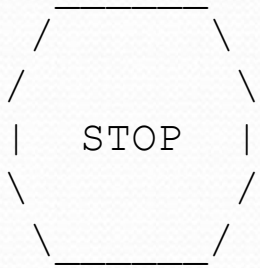
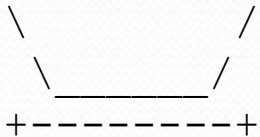
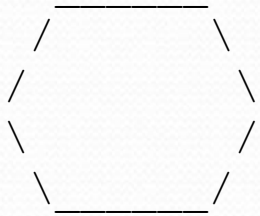
```
This is message1.
This is message2.
This is message1.
Done with message2.
Done with main.
```

When to use methods

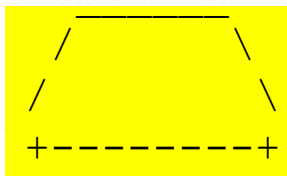
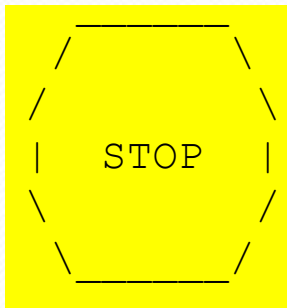
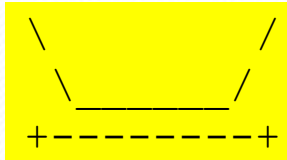
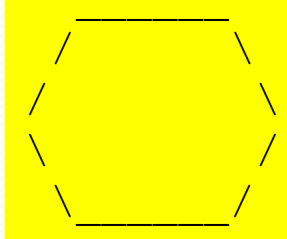
- Place statements into a static method if:
 - The statements are related structurally, and/or
 - The statements are repeated.
- You should not create static methods for:
 - An individual `println` statement.
 - Only blank lines. (Put blank `println`s in `main`.)
 - Unrelated or weakly related statements.
(Consider splitting them into two smaller methods.)

Static methods question

- Write a program to print these figures using methods.



Output structure



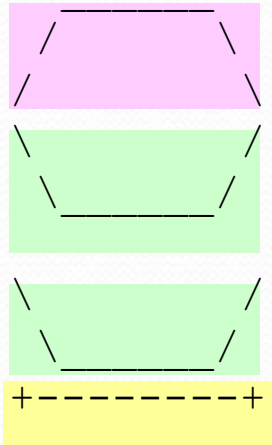
The structure of the output:

- initial "egg" figure
- second "teacup" figure
- third "stop sign" figure
- fourth "hat" figure

This structure can be represented by methods:

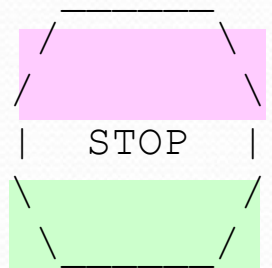
- egg
- teaCup
- stopSign
- hat

Output redundancy



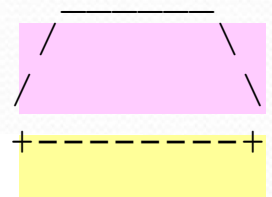
The redundancy in the output:

- egg top: reused on stop sign, hat
- egg bottom: reused on teacup, stop sign
- divider line: used on teacup, hat



This redundancy can be fixed by methods:

- `eggTop`
- `eggBottom`
- `line`



Data and expressions

reading: 2.1

Data types

- Internally, computers store everything as 1s and 0s

104 → 01101000

"hi" → 0110100001101001

h → 01101000

- How are `h` and `104` differentiated?
- **type**: A category or set of data values.
 - Constrains the operations that can be performed on data
 - Many languages ask the programmer to specify types
 - Examples: integer, real number, string

Java's primitive types

- **primitive types**: 8 simple types for numbers, text, etc.
 - Java also has **object types**, which we'll talk about later

Name	Description	Examples
<code>int</code>	integers (up to $2^{31} - 1$)	<code>42, -3, 0, 926394</code>
<code>double</code>	real numbers (up to 10^{308})	<code>3.1, -0.25, 9.4e3</code>
<code>char</code>	single text characters	<code>'a', 'X', '?', '\n'</code>
<code>boolean</code>	logical values	<code>true, false</code>

- Why does Java distinguish integers vs. real numbers?

Integer or real number?

- Which category is more appropriate?

integer (<code>int</code>)	real number (<code>double</code>)

1. Temperature in degrees Celsius
2. The population of lemmings
3. Your grade point average
4. A person's age in years
5. A person's weight in pounds
6. A person's height in meters
7. Number of miles traveled
8. Number of dry days in the past month
9. Your locker number
10. Number of seconds left in a game
11. The sum of a group of integers
12. The average of a group of integers

- credit: Kate Deibel, <http://www.cs.washington.edu/homes/deibel/CATs/>

Expressions

- **expression:** A value or operation that computes a value.
 - Examples: $1 + 4 * 5$
 $(7 + 2) * 6 / 3$
42
 - The simplest expression is a *literal value*.
 - A complex expression can use operators and parentheses.

Arithmetic operators

- **operator:** Combines multiple values or expressions.

+	addition
-	subtraction (or negation)
*	multiplication
/	division
%	modulus (a.k.a. remainder)

- As a program runs, its expressions are *evaluated*.
 - `1 + 1` evaluates to 2
 - `System.out.println(3 * 4);` prints 12
 - How would we print the text `3 * 4` ?

Integer remainder with %

- The % operator computes the remainder from integer division.

- $14 \% 4$ is 2

- $218 \% 5$ is 3

$$\begin{array}{r} \underline{3} \\ 4 \) \ 14 \\ \underline{12} \\ \mathbf{2} \end{array}$$

$$\begin{array}{r} \underline{43} \\ 5 \) \ 218 \\ \underline{20} \\ 18 \\ \underline{15} \\ \mathbf{3} \end{array}$$

What is the result?

$$45 \% 6$$

$$2 \% 2$$

$$8 \% 20$$

$$11 \% 0$$

- Applications of % operator:

- Obtain last digit of a number: $230857 \% 10$ is 7

- Obtain last 4 digits: $658236489 \% 10000$ is 6489

- See whether a number is odd: $7 \% 2$ is 1, $42 \% 2$ is 0

Precedence

- **precedence:** Order in which operators are evaluated.

- Generally operators evaluate left-to-right.

$1 - 2 - 3$ is $(1 - 2) - 3$ which is -4

- But $*$ / $\%$ have a higher level of precedence than $+$ -

$1 + 3 * 4$ is 13

$6 + 8 / 2 * 3$

$6 + 4 * 3$

$6 + 12$ is 18

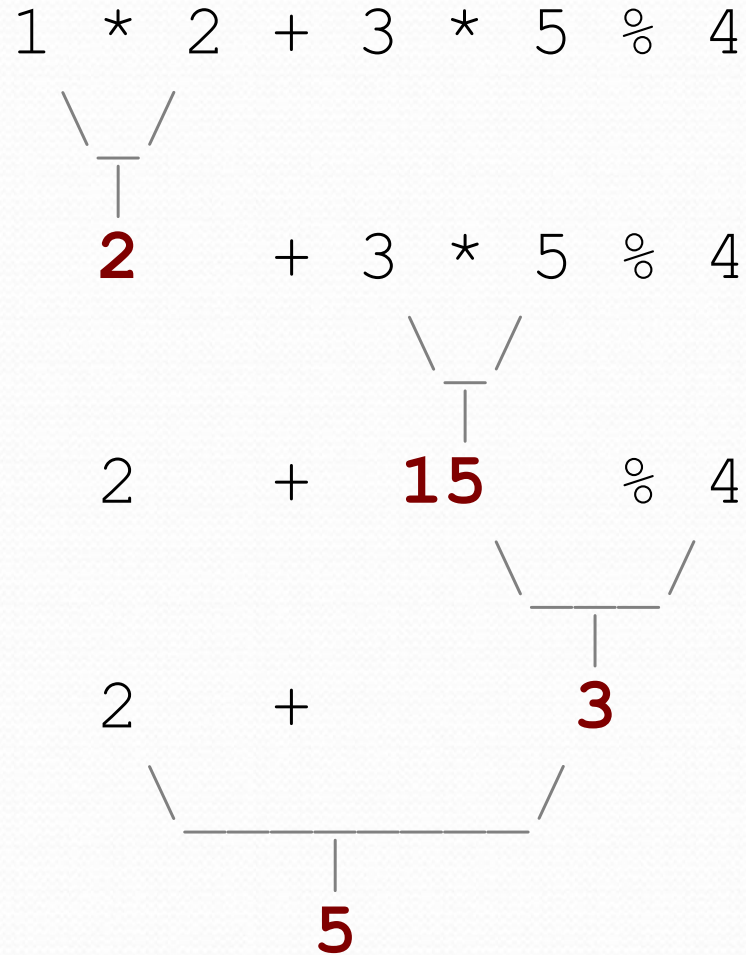
- Parentheses can force a certain order of evaluation:

$(1 + 3) * 4$ is 16

- Spacing does not affect order of evaluation

$1+3 * 4-2$ is 11

Precedence examples



Precedence questions

- What values result from the following expressions?
 - $9 / 5$
 - $695 \% 20$
 - $7 + 6 * 5$
 - $7 * 6 + 5$
 - $248 \% 100 / 5$
 - $6 * 3 - 9 / 4$
 - $(5 - 7) * 4$
 - $6 + (18 \% (17 - 12))$

Real numbers (type double)

- Examples: 6.022 , -42.0 , 2.143e17
 - Placing .0 or . after an integer makes it a double.
- The operators + - * / % () all still work with double.
 - / produces an exact answer: 15.0 / 2.0 is 7.5
 - Precedence is the same: () before * / % before + -

Real number example

$$2.0 * 2.4 + 2.25 * 4.0 / 2.0$$



4.8

$$+ 2.25 * 4.0 / 2.0$$



9.0

$$4.8 + 9.0 / 2.0$$



4.5

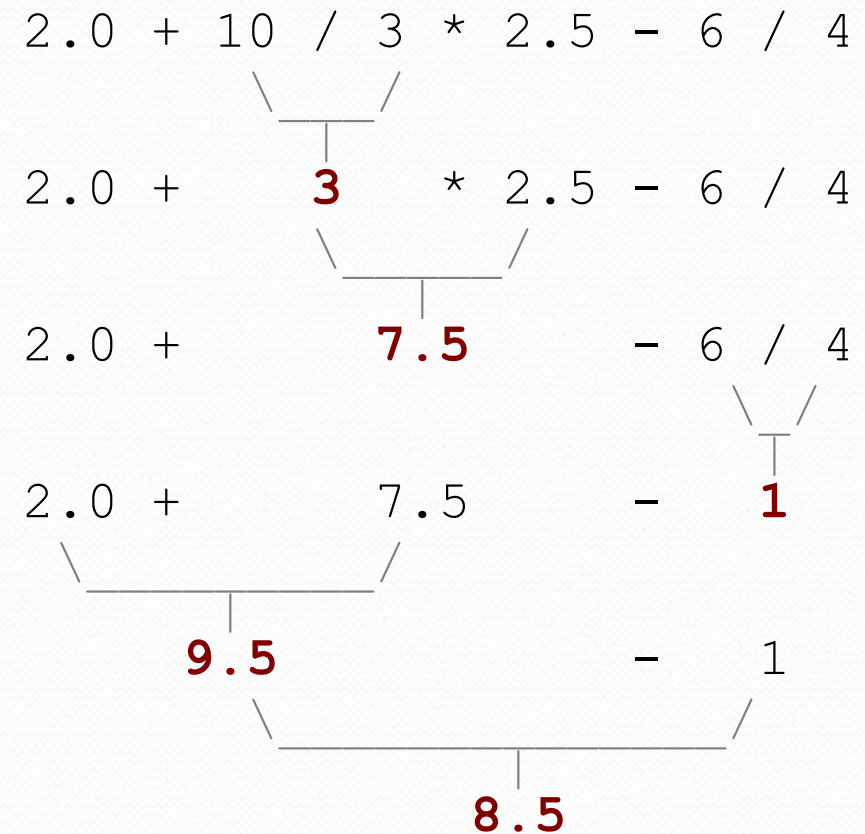
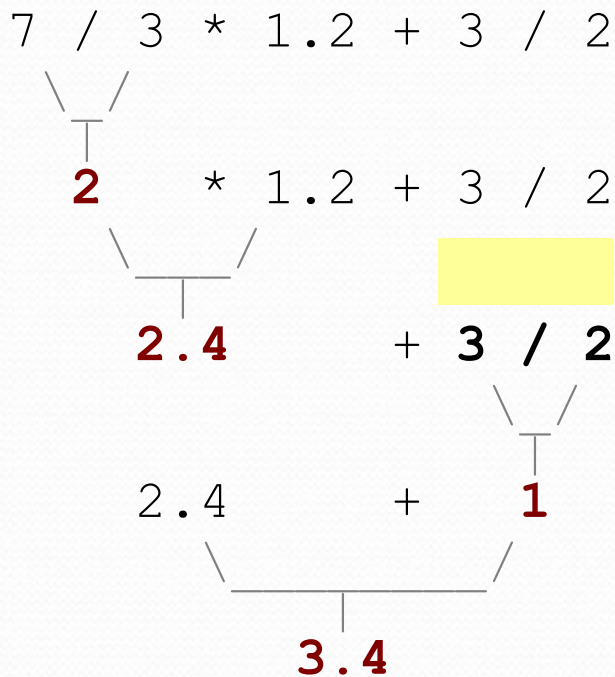
$$4.8 + 4.5$$



9.3

Mixing types

- When `int` and `double` are mixed, the result is a double.
 - `4.2 * 3` is `12.6`
- The conversion is per-operator, affecting only its operands.



- `3 / 2` is `1` above, not `1.5`.

String concatenation

- **string concatenation:** Using + between a string and another value to make a longer string.

```
"hi" + "hello"      is "hihello"  
"hello" + 42        is "hello42"  
1 + "abc" + 2       is "1abc2"  
"abc" + 1 + 2       is "abc12"  
1 + 2 + "abc"       is "3abc"  
"abc" + 9 * 3       is "abc27"  
"1" + 1             is "11"  
4 - 1 + "abc"       is "3abc"
```

- Use + to print a string and an expression's value together.
 - `System.out.println("Grade: " + (95.1 + 71.9) / 2);`
 - **Output:** Grade: 83.5