CIS 110: Introduction to Computer Programming

Lecture 8

Hey (Objects), Listen!

(§ 3.2-3.3)

Outline

- Review: what is a library?
- The Math class
- The String class
- The Story Thus Far…

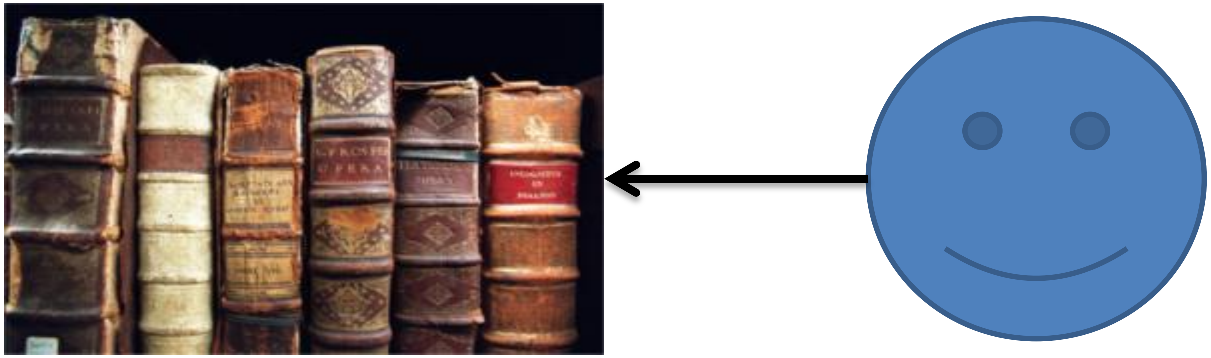
Announcements

- Homework 3 due tonight (11:59:59).
- Exam 1 on Wednesday.
  - See website for locations (based on time + name).
  - No homework or lab this week.
- Changes to exam protocol:
  - **ID required to turn in exam.**
  - Abbreviations for System.out.print(ln):
    - **S.O.PLN** and **S.O.P**
    - **All other code must be written out in full!**

What is a Library?

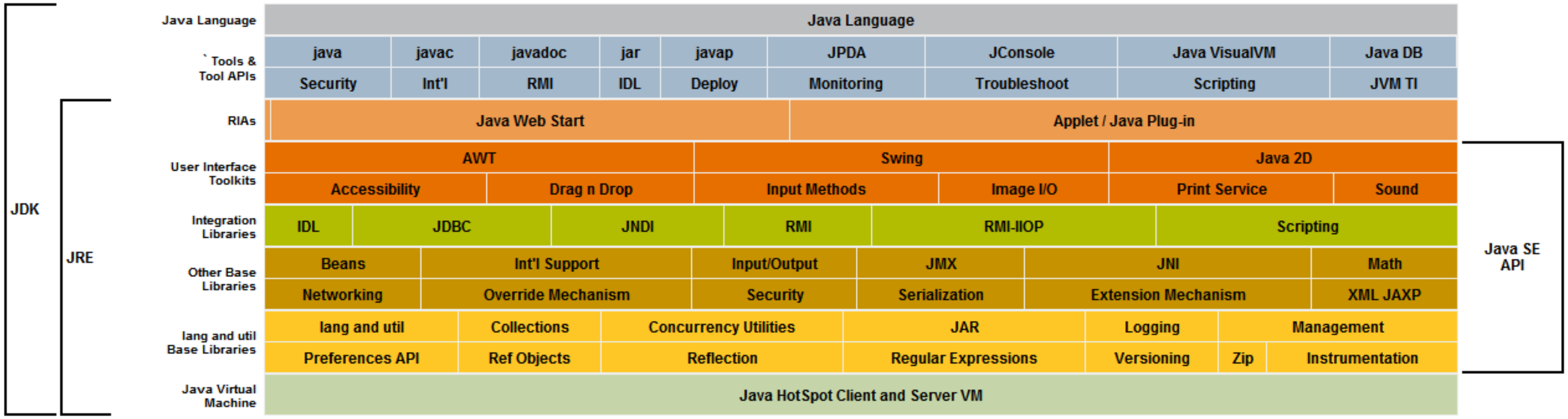
Review: Libraries

- *Libraries* are collections of classes that other people have written for us to use.
  - Motivation: avoid reinventing the wheel!
  - A particular *strength* of Java.
- Example: the DrawingPanel class



Learning a Programming Language

- Need to learn *two parts*:
  - *Language*: syntax, structure, etc.
  - *Libraries:* commonly-used functionality.
- Java's built-in library: the *Java Class Libraries*.



JDK

JRE

Java SE API

Java Language

Java Language

` Tools & Tool APIs

java, javac, javadoc, jar, javap, JPDA, JConsole, Java VisualVM, Java DB

Security, Int'l, RMI, IDL, Deploy, Monitoring, Troubleshoot, Scripting, JVM TI

RIAs

Java Web Start, Applet / Java Plug-in

User Interface Toolkits

AWT, Swing, Java 2D

Accessibility, Drag n Drop, Input Methods, Image I/O, Print Service, Sound

Integration Libraries

IDL, JDBC, JNDI, RMI, RMI-IIOP, Scripting

Other Base Libraries

Beans, Int'l Support, Input/Output, JMX, JNI, Math

Networking, Override Mechanism, Security, Serialization, Extension Mechanism, XML JAXP

lang and util Base Libraries

lang and util, Collections, Concurrency Utilities, JAR, Logging, Management

Preferences API, Ref Objects, Reflection, Regular Expressions, Versioning, Zip, Instrumentation

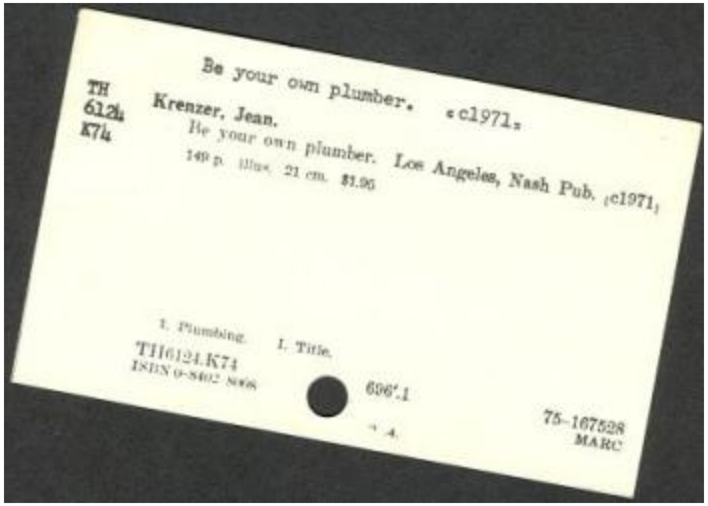
Java Virtual Machine

Java HotSpot Client and Server VM

Description of Java Conceptual Diagram

Learning a Programming Language

- In reality, few people know the entire library.
  - People know where to *look stuff up when needed*.
  - In Java, this is the *javadoc API documentation*.
- For now, we focus on two particular classes:
  - The Math class.
  - The String class.



Be your own plumber. ₍c1971₎

TH
6124
K74

Krenzer, Jean.
Be your own plumber. Los Angeles, Nash Pub. ₍c1971₎
149 p. illus. 21 cm. $1.95

1. Plumbing. I. Title.
TH6124.K74
ISBN 0-8402 [illegible]

696'.1

75-167528
MARC

The Math Class

Demo (MathExamples.java)

Math Example

- Can we write a method that rounds a decimal to an integer?
  - MyMath.round(3.75) → 4
  - MyMath.round(3.25) → 3

```
public class MyMath {
  public static int round(double d) {
    return (int) (d + 0.5);
  }
}
```

Calling Methods from Other Classes

- Can't simply call sqrt from another class!

```
public class MathTest {
  public static void main(String[] args) {
    System.out.println(sqrt(3.75)); // Bad!
  }
}
```

- sqrt is *out of scope* in MathTest.

Dot Notation for Static Methods and Constants

- Solution: clarify where sqrt comes from with *dot notation*.
  - *Syntax:* <class name>.<method name>().

```
public class MathTest {
  public static void main(String[] args) {
    System.out.println(MyMath.sqrt(3.75));
  }
}
```

- Only works for static methods and constants.
  - *Not for local variables!*

The Math class

- Java already provides round in the Math class.
  - Best to not reinvent the wheel!

```
public class MathTest {
  public static void main(String[] args) {
    System.out.println(Math.sqrt(3.75));
  }
}
```

Math Methods and Constants Sampler (See p. 150 for details)

```
Math.E
Math.PI
Math.abs(num)
Math.ceil(num)
Math.exp(num)
Math.floor(num)
Math.log(num)
Math.log10(num)
Math.max(num1, num2)
Math.min(num1, num2)
Math.pow(num1, num2)
Math.random()
Math.round(num)
```

```
Math.sin(num)
Math.sqrt(num)
Math.toDegrees(num)
Math.toRadians(num)
```

The String Class

Demo (StringExamples.java)

Recall: Objects Are Different From Primitives

- DrawingPanel is an object!
  - Different syntax for creation and method calls:

```
DrawingPanel p = new DrawingPanel(500, 500);
Graphics g = p.getGraphics();
```

- Remember objects contain *data* and *methods*.
  - e.g., DrawingPanel provides the getGraphics method.

Startling Fact: Strings Are Objects!

- Strings try to act like primitives...

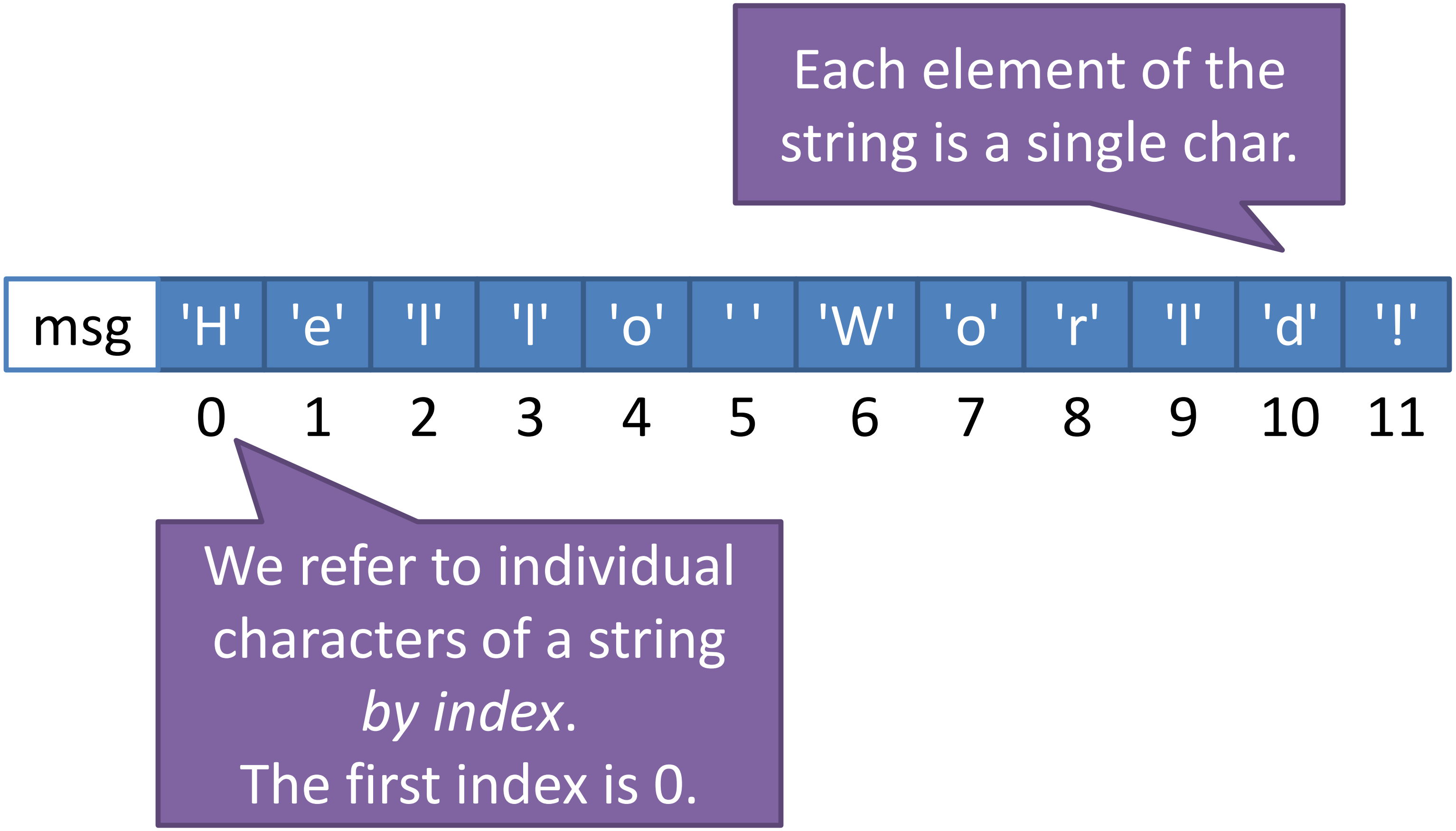
```
String msg = "hello world!"
```

- ...but they're really objects!

```
System.out.println(msg.length());
// Prints out 12 --- the length of msg!
```

- What other methods does String provide?

Strings as sequence of characters



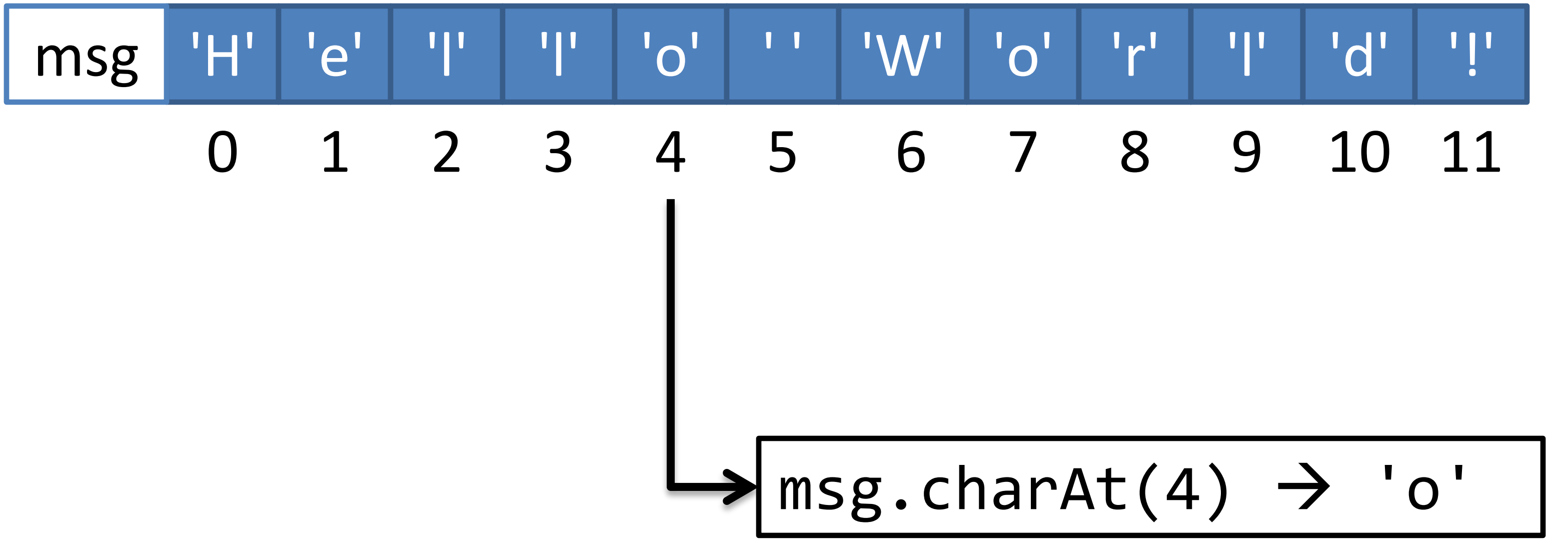
Each element of the string is a single char.

msg	'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'!'
	0	1	2	3	4	5	6	7	8	9	10	11

We refer to individual characters of a string *by index*.
The first index is 0.

Example: charAt(index)

```
System.out.println(msg.charAt(4));
// Prints out 'o' --- the char at index 4
```

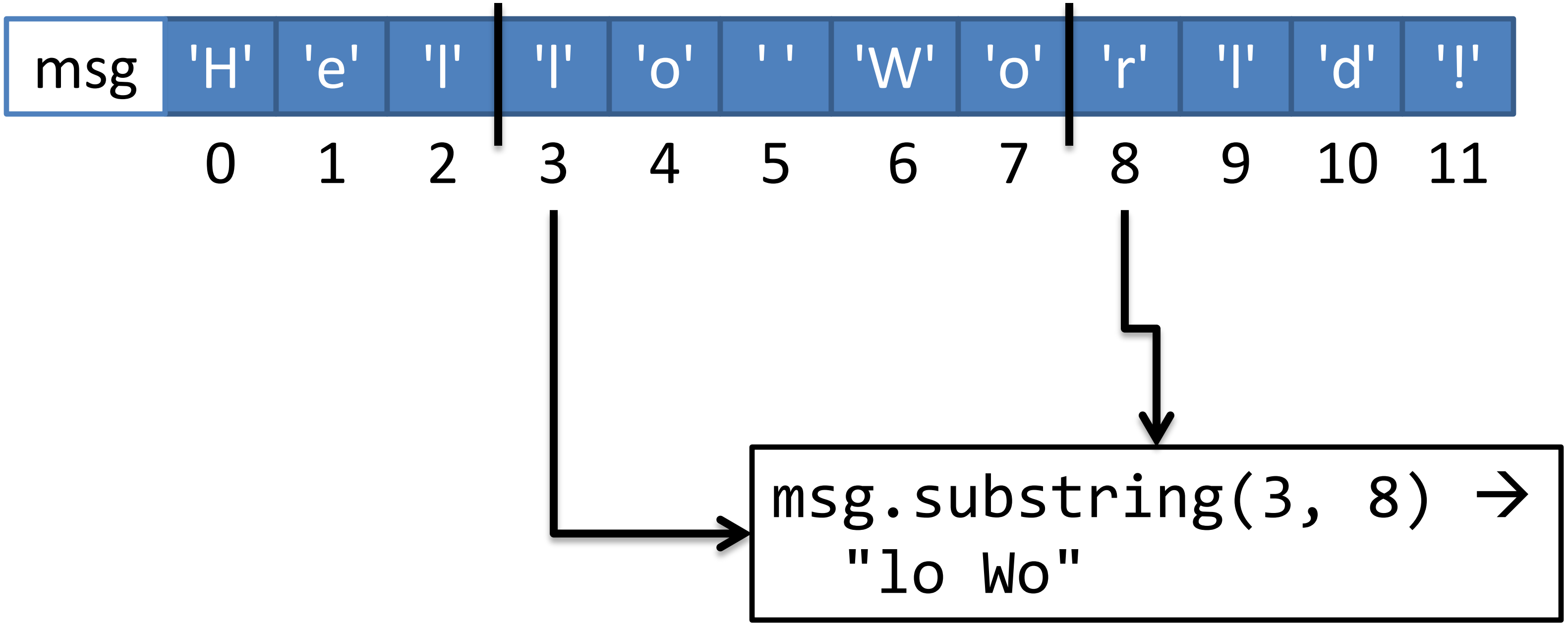


msg	'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'!'
	0	1	2	3	4	5	6	7	8	9	10	11

```
msg.charAt(4) → 'o'
```

Example: substring(start, end)

```
System.out.println(msg.substring(3, 8));
// Prints out "lo Wo": the substring starting
// at index 3 and ending at index (8-1) = 7.
```



msg	'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'!'
	0	1	2	3	4	5	6	7	8	9	10	11

```
msg.substring(3, 8) →
   "lo Wo"
```

Runtime Errors From Bad Arguments

```
System.out.println(msg.charAt(12));
// Raises an exception since 12 is not valid!
```

msg	'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'!'
	0	1	2	3	4	5	6	7	8	9	10	11

Strings Are Immutable

```
String msg = "Hello World!";
msg.toUpperCase();
System.out.println(msg);      // Prints Hello World!
```

- String are *immutable*!
  - Methods calls on a String do not change that String.
  - Instead, methods *return new Strings* that are the result of the operation.

```
String msg = "Hello World!";
msg = msg.toUpperCase();
System.out.println(msg);      // Prints HELLO WORLD!
```

String Methods Sampler
(See p. 162 for details)

```
charAt(index)
endsWith(text)
indexOf(text)
length()
startsWith(text)
substring(start, stop)
toLowerCase()
toUpperCase()
```

And The Story Thus Far…

Why Are We Here?

- Learning about *algorithmic thinking* via *computer programming!*

1. Precision
2. Decomposition
3. Abstraction

Mental Model of Computation

Static methods
For loops

Class Constants
Parameters
Return values

Syntax: Declarations

```
// Class declarations
public class <name> {
  <methods>
}

// Method declaration
public static <type> <name>(<params>) {
  <statements>
}
```

Syntax: Statements

```
// Static method call
<name>(<params>);

// Static method call (another class)
<class>.<name>(<params>);

// Method call on an object
<object>.<name>(<params>);

// Variable declaration
<type> <name> = <expr>

// Variable assignment
<name> = <expr>
<name> += <expr>
<name++>
```

```
// For loop
for (<init>; <test>; <update>) {
  <statements>
}
```

Syntax: Expressions

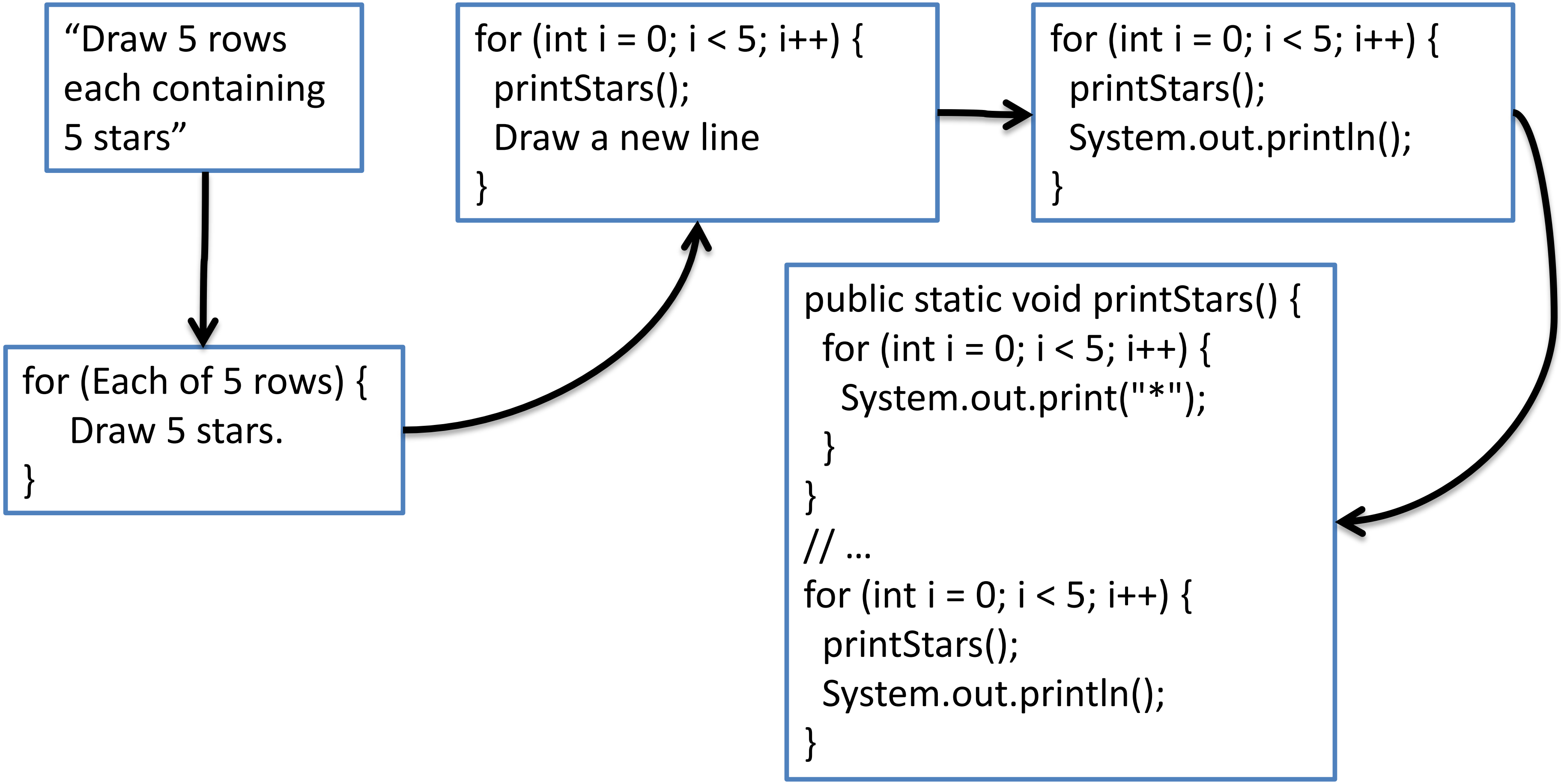
```
// All method calls if they return values (from previous slide)

// Variable use
<name>

// Literals
0  0.0  'c'  "hello!"

// Operators (+, -, *, /, %)
<expr> + <expr>
```

Decomposition: Break A Problem Into Sub-problems



“Draw 5 rows each containing 5 stars”

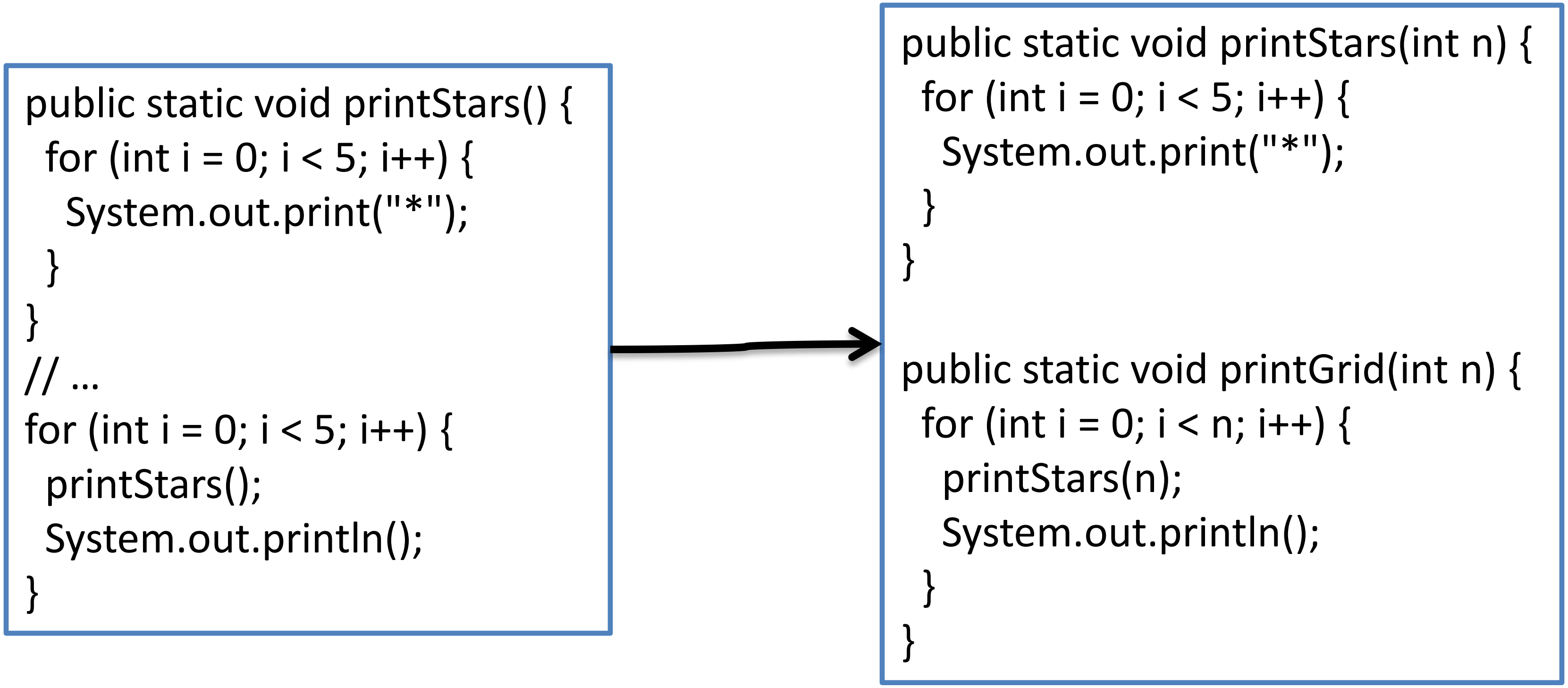
```
for (Each of 5 rows) {
     Draw 5 stars.
}
```

```
for (int i = 0; i < 5; i++) {
  printStars();
  Draw a new line
}
```

```
for (int i = 0; i < 5; i++) {
  printStars();
  System.out.println();
}
```

```
public static void printStars() {
  for (int i = 0; i < 5; i++) {
    System.out.print("*");
  }
}
// …
for (int i = 0; i < 5; i++) {
  printStars();
  System.out.println();
}
```

Generalization: Make a Piece of Code Handle More Cases



```
public static void printStars() {
  for (int i = 0; i < 5; i++) {
    System.out.print("*");
  }
}
// …
for (int i = 0; i < 5; i++) {
  printStars();
  System.out.println();
}
```

```
public static void printStars(int n) {
  for (int i = 0; i < 5; i++) {
    System.out.print("*");
  }
}

public static void printGrid(int n) {
  for (int i = 0; i < n; i++) {
    printStars(n);
    System.out.println();
  }
}
```