CSE 121 Lesson 7: Methods, Parameters, Returns

Matt Wang & Brett Wortzman

Autumn 2024

sli.do #cse121

TAs:

Abby	Afifah	Ailsa	Alice	Aliyan	Arohan
Chloë	Christopher	Dalton	Derek	Elizabeth	Ethan
Hanna	Hannah	Heather	Hibbah	Janvi	Jasmine
Judy	Julia	Kelsey	Lucas	Luke	Mahima
Maitreyi	Maria	Merav	Minh	Neha	Ronald
Ruslana	Sahej	Sam	Samrutha	Sushma	Vivian
Yijia	Zachary				

Today's playlist:
121 24au lecture tunes

Announcements, Reminders

- P1 is out, due next Tuesday October 22nd
  - Start early! This one is tough!
  - Doing P1 is *also* studying for the quiz!
- R1 released yesterday, due Thursday October 24
- Post-section work grades now on Canvas (!!)
  - in future: weekly-ish updates (but not instant)
- No PCM for next Wed :)

Quiz 0 (1/2)

- Quiz 0 is Thursday, Oct 24!
  - Big review opportunity: section on Tuesday, October 22nd
  - Practice Quiz is out!
  - stay tuned for potentially more resources
- General notes:
  - taken on your computer, in your quiz section
  - broadly: mostly focused on concepts, reading, and debugging code
  - covers material up to Wednesday's lecture (methods & parameters), but no further (e.g. no returns)

Quiz 0 (2/2)

- Policy notes
  - **Please read the policies & procedures in the practice quiz. You are responsible for following these rules.**
  - (live in lecture, let's go over some of these right now!!)
- Advice
  - do the practice quiz in an environment similar to the quiz: time yourself, only used allowed resources, etc.
  - organize your notes – open book doesn't mean "no notes required"!
  - go to section!!

(Review) Class Constants

A fixed value visible to the whole program (the entire *class*).

- Value can be set only at declaration; **cannot** be reassigned (so the value is constant)

```
public static final type NAME_OF_CONSTANT = expression;
```

(Review) Parameters

Definition: A value passed to a method by its caller

```
public static void myMethod(String musicalAct) {
    System.out.print(musicalAct + " is the best!");
    ...
}
```

Calling a method with a parameter…

```
myMethod("Laufey"); // Laufey is the best!
```

(Review) Scope – in for loops

The part of a program where a variable exists.

- From its declaration to the end of the { } braces
- Ex: a variable declared in a for loop only exists in that loop

innerloop's scope

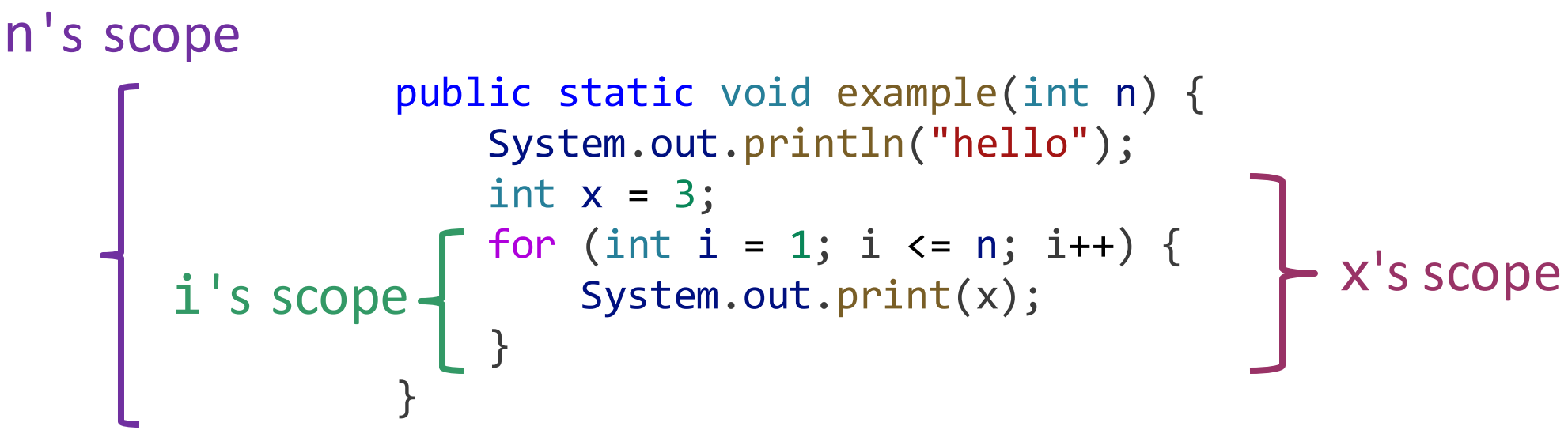
```
for (int outerLoop = 1; outerLoop <= 5; outerLoop++) {
    System.out.println("outer loop iteration #" + outerLoop);
    for (int innerLoop = 1; innerLoop <= 3; innerLoop++) {
        System.out.println("    inner loop iteration #" + innerLoop);
    }
    System.out.println(outerLoop);
}
```

outerloop's scope

(Review) Scope – in methods

The part of a program where a variable exists.

- From its declaration to the end of the { } braces
- Ex: a variable declared in a method exists only in that method



Poll in with your answer!

What will be the **last line of output** from this code?

```
public static final int COUNT = 7;
public static void main(String[] args) {
  int count = 5;
  line(count);
  System.out.println("count is: " + count);
}

public static void line(int count) {
  for (int i = 1; i <= count; i++) {
    System.out.print("*");
  }
  count++;
  System.out.println();
}
```

sli.do #cse121

A. `count is: 1`

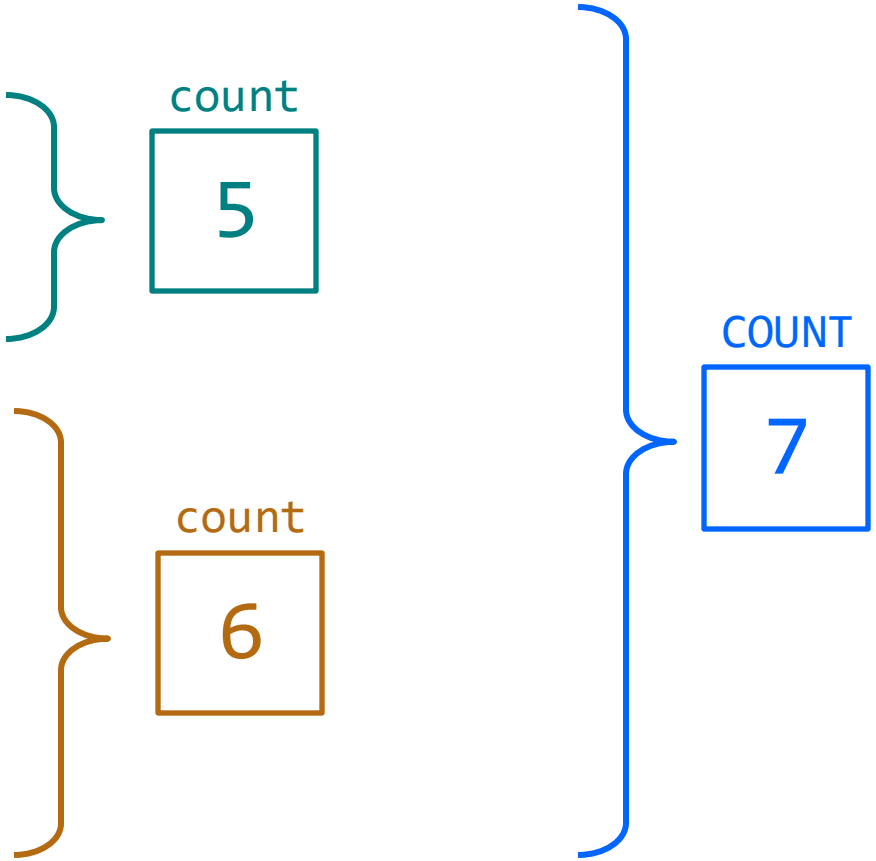
B. `count is: 5`

C. `count is: 6`

D. `count is: 7`

Walkthrough: Counting Counts

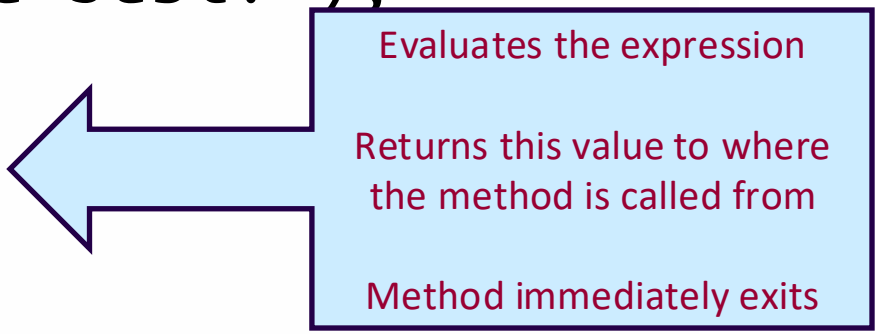
```
public static final int COUNT = 7;
public static void main(String[] args) {
  int count = 5;
  line(count);
  System.out.println("count is: " + count);
}

public static void line(int count) {
  for (int i = 1; i <= count; i++) {
    System.out.print("*");
  }
  count++;
  System.out.println();
}
```



(PCM) Returns

Returns allow us to send values *out of a method*

```
public static <type> myMethod(int num) {
    System.out.print(num + " is the best!");
    ...
    return <value of correct type>
}
```



Calling a method that returns a value…

```
<type> result = myMethod(42);
```

(Recall) String Methods

Usage: `<string variable>.<method>(…)`

Method	Description
`length()`	**Returns** the length of the string.
`charAt(i)`	**Returns** the character at index *i* of the string
`indexOf(s)`	**Returns** the index of the first occurrence of *s* in the string; returns -1 if *s* doesn't appear in the string
`substring(i, j)` or `substring(i)`	**Returns** the characters in this string from *i* (inclusive) to *j* (exclusive); if *j* is omitted, goes until the end of the string
`contains(s)`	**Returns** whether or not the string contains *s*
`equals(s)`	**Returns** whether or not the string is equal to *s* (case-sensitive)
`equalsIgnoreCase(s)`	**Returns** whether or not the string is equal to *s* ignoring case
`toUpperCase()`	**Returns** an uppercase version of the string
`toLowerCase()`	**Returns** a lowercase version of the string

Returns & String Methods

```
String s = "bubblegum";
s = s.substring(7, 8).toUpperCase() + s.substring(8) + "ball";

s =                   "g".toUpperCase() + s.substring(8) + "ball";

s =                                     "G" + s.substring(8) + "ball";

s =                                           "G" + "um" + "ball";
```

Poll in with your answer!

sli.do #cse121

To go from Celsius to Fahrenheit, you multiply by 1.8 and then add 32. Which of these correctly implements this logic as a method?

A.

```
public static void celsiusToF(double celsius) {
  double fahrenheit = celsius * 1.8 + 32;
  return fahrenheit;
}
```

B.

```
public static void celsiusToF(double celsius) {
  double fahrenheit = celsius * 1.8 + 32;
}
```

C.

```
public static double celsiusToF(double celsius) {
  int fahrenheit = celsius * 1.8 + 32;
  return fahrenheit;
}
```

D.

```
public static double celsiusToF(double celsius) {
  return celsius * 1.8 + 32;
}
```

Poll in with your answer!

sli.do #cse121

What value is returned from this method?

```
public static int returnExample() {
  for (int i = 0; i < 5; i++) {
    return i;
  }
  return -1;
}
```

A. -1

B. 0

C. 4

D. 5

Method Comments!

- Now that we know how to write methods, we have a new form of documentation (using comments) to write.
- Each method you write (except for main) should be accompanied by a short comment that describes what it does.
- **Be sure to comment on method behavior, and all parameters and returns of a method!**

```
// Randomly generates an addition problem where the
// operands are in the range 1-10 (inclusive), and prints the result
// rounded to two decimal places.
public static void addTwoRandomNumbers() {
    Random randy = new Random();
    int num1 = randy.nextInt(10) + 1;
    int num2 = randy.nextInt(10) + 1;
    int sum = num1 + num2;
    ...
}
```