LEC 07

CSE 121

Methods, Parameters, Returns

Questions during Class?

Raise hand or send here

sli.do #cse121

BEFORE WE START

Talk to your neighbors:

What are your weekend (or Weeknd) plans?

Music: 121 25wi lecture playlist

Instructor: Matt Wang

TAs: Ailsa, Alice, Chloë, Christopher, Ethan, Hanna, Hannah, Hibbah, Janvi, Judy, Julia, Kelsey, Lucas, Luke, Maitreyi, Merav, Ruslana, Samrutha, Sam, Shayna, Sushma, Vivian

a brief note on, **gestures wildly**

Wanted to acknowledge that:

- laws (and our interpretation of them) are changing rapidly
- there's a lot of uncertainty, and many questions don't have answers
- this can be <u>extraordinarily</u> stressful (and makes it hard to do 121!)

Materially,

- if this is affecting your ability to engage in CSE 121, please let me know – I'm more than happy to chat
- I'm also happy to refer you to on-campus resources that are helpful

Announcements, Reminders

- P1 is out, due next Tuesday, Feb 4th
  - Start early – this one is tough!
  - Doing P1 is *also* studying for the quiz
- R1 released yesterday, due Thursday Feb 6th
- Quiz 0 is on Thursday, February 6 (in your registered quiz section)
  - can't make it? email Matt ASAP
- As you potentially rewatch lectures – Ed megathreads reminder!
- No pre-class work for next Wed :)

Even more about Quiz 0…

Quiz 0 is Thursday, Feb 6th!

Please read the policies & procedures.
You are responsible for following these rules!

General advice:

- do the practice quiz in an environment like an actual quiz: time yourself, only used allowed resources, etc.
- organize your notes – open book *doesn't* mean "no notes required"!
- get some sleep – you won't do well with no sleep (or food)

On "studying deliberately"

Biggest advice: **study deliberately**; as a sketch,

- catch up to methods & parameters
- then, do the first practice quiz
- **reflect**: what went well, what topics were shaky?
- **review those topics** (with practice problems, office hours, …)
- then, do the second practice quiz – do you see the same problems?

Targeted, deliberate practice gives you a better return!

New: Class Constants

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

Value is set at declaration, **cannot** be reassigned – value is ***constant***.

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

Practice: Think

sli.do #cse121

```
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();
}
```

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

A. `count is: 1`

B. `count is: 5`

C. `count is: 6`

D. `count is: 7`

Practice: Pair

sli.do #cse121

```
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();
}
```

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

A. `count is: 1`

B. `count is: 5`

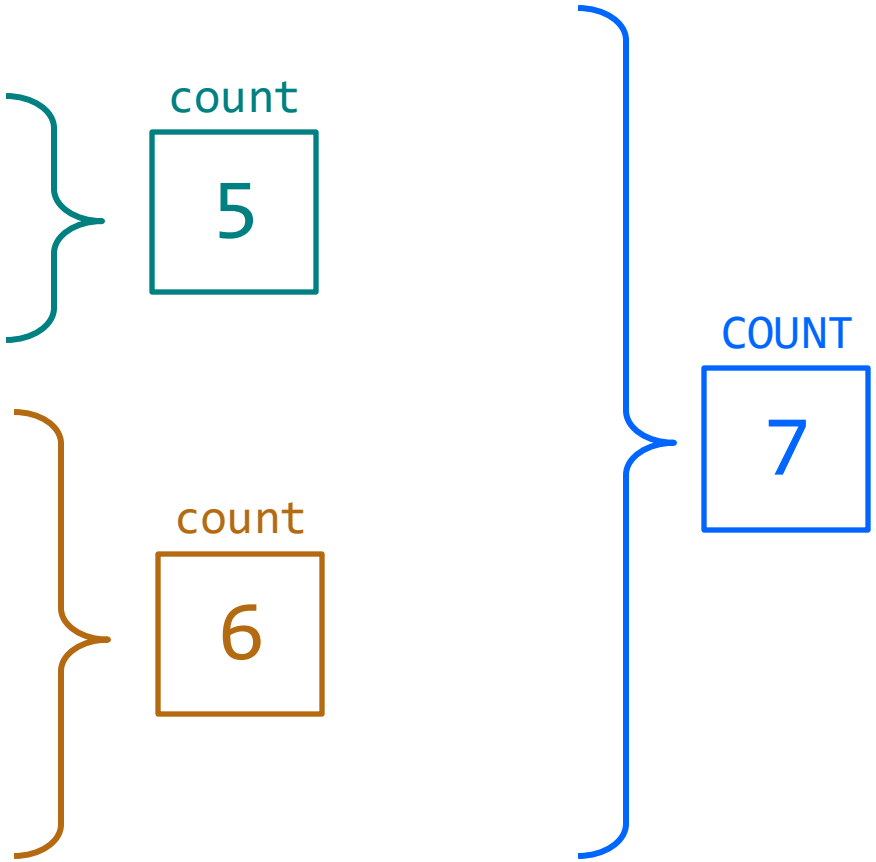
C. `count is: 6`

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>
}
```

Evaluates the expression

Returns this value to where the method is called from

Method immediately exits

Calling a method that returns a value…

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

Recall: Math

Method	Description
`Math.abs(value)`	**Returns** the absolute value of *value*
`Math.ceil(value)`	**Returns** *value* rounded up
`Math.floor(value)`	**Returns** *value* rounded down
`Math.max(value1, value2)`	**Returns** the larger of the two values
`Math.min(value1, value2)`	**Returns** the smaller of the two values
`Math.round(value)`	**Returns** *value* rounded to the nearest whole number* note: need to cast result to int (it's complicated!)
`Math.sqrt(value)`	**Returns** the square root of *value*
`Math.pow(base, exp)`	**Returns** *base* raised to the *exp* power

Recall: String Methods

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

Reminder: Gumball & Strings

```
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";
```

Practice: Think

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;
}
```

Practice: Pair

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;
}
```

Announcements, Reminders (again)

- P1 is out, due next Tuesday, Feb 4th
  - Start early – this one is tough!
  - Doing P1 is *also* studying for the quiz
- R1 released yesterday, due Thursday Feb 6th
- Quiz 0 is on Thursday, February 6 (in your registered quiz section)
  - can't make it? email Matt ASAP
- As you potentially rewatch lectures – Ed megathreads reminder!
- No pre-class work for next Wed :)