CSE 121 Lesson 8 : 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

- Creative Project 2 (C2) releasing later today (due Tuesday, Oct 29th)
- R1 due tomorrow; R2 opens tomorrow (due Thursday, Oct 31st 🎃 👻)
  - R2 eligible assignments: C0, P0, C1, P1
- Friday, Nov 1st: Mid-quarter Formative Feedback in class
- Quiz 0 is tomorrow in your quiz section!
  - Quiz logistics announced on Ed
- Quiz review reminders
  - Priority materials: practice quizzes, starred section problems!
  - Ed Class megathreads (great while reviewing lecture)

(Recall) Methods & Parameters

Definition: A value passed to a method by its caller; sending information into a method

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

Calling a method with a parameter…

```
myMethod("Rush"); // Prints out
                  // "Rush is the best!"
```

(Recall) Returns 1

Returns allow us to send values out of a method

```
public static <type> myMethod(<zero or more params>) {
    ...
    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(...); // catching what is returned!
```

(Recall) Returns 2

Returns allow us to send values out of a method

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

Calling a method with a parameter…

```
String s = myMethod("Rush"); // Prints and returns
                             // "Rush is the best!"
```

(Recall) Returns 3

Returns allow us to send values out of a method

```
public static String myMethod(String musicalAct) {
    ...
    return musicalAct + " is the best!”
}
```

Calling a method with a parameter…

```
String s = myMethod("Rush"); // Returns
                             // "Rush is the best!"
```

(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

String example

```
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";
```

Example of returns: Math class

Methods	Returns
`Math.abs(`*value*`)`	Absolute value of *value*
`Math.ceil(`*value*`)`	*value* rounded up
`Math.floor(`*value*`)`	*value* rounded down
`Math.max(`*value1*`, `*value2*`)`	Larger of the two given values
`Math.min(`*value1*`, `*value2*`)`	Smaller of the two given values
`Math.round(`*value*`)`	*value* rounded to the nearest whole number
`Math.sqrt(`*value*`)`	Square root of *value*
`Math.pow(`*base*`, `*exp*`)`	*base* to the *exp* power

Math example

```
double value = 823.577564893;
double roundedValue = (double) Math.round(value * 100) / 100;
                    = (double) Math.round( 82357.7564893 ) / 100;
                              = (double) 82358.0 / 100;
                              = 823.58
```

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.

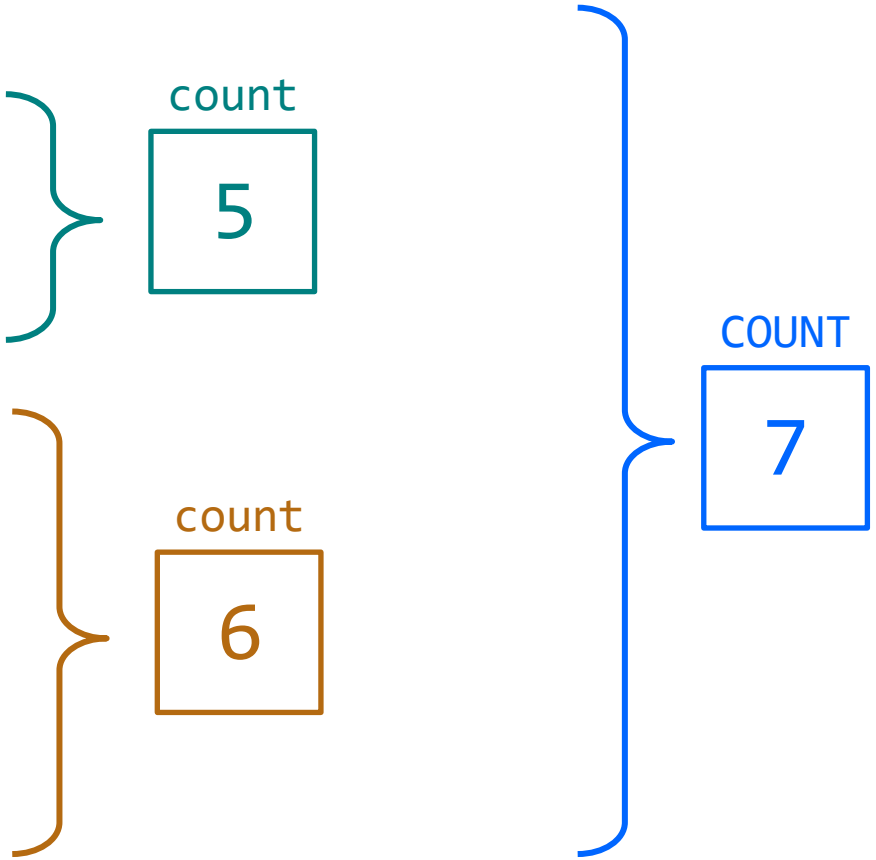
```
public static double celsiusToF(double celsius) {
  int fahrenheit = celsius * 1.8 + 32;
  return fahrenheit;
}
```

D.

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

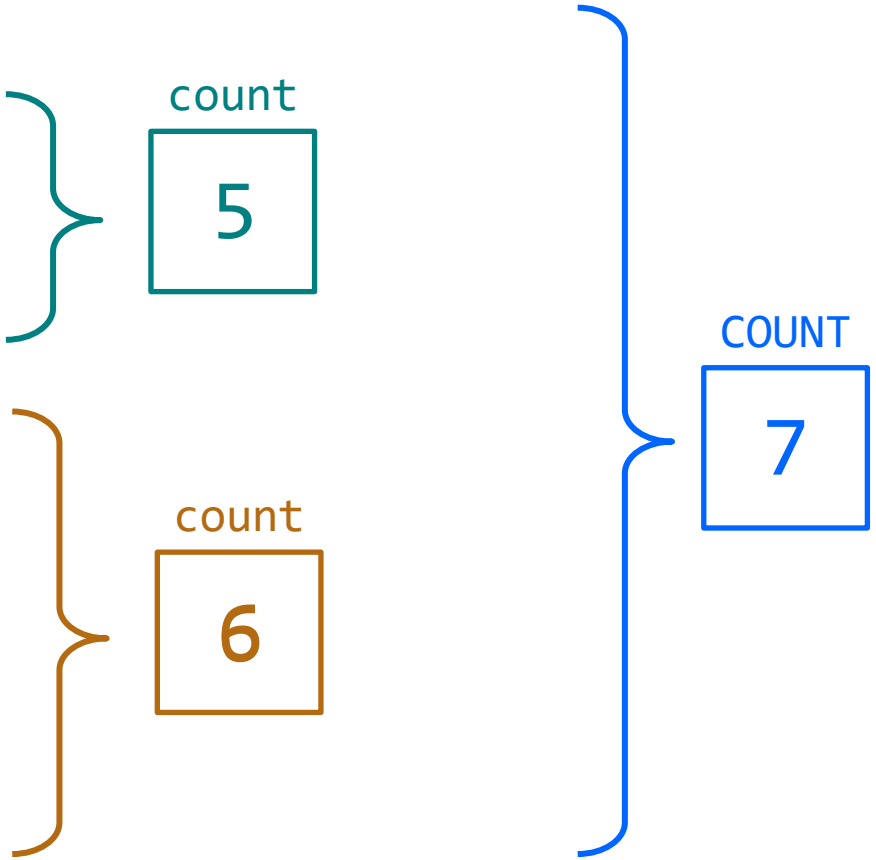
(Recall) Tricky Poll: Last line printed?

```
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();
}
```



Tricky Poll: Returnable

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

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



Poll in with your answer!

sli.do #cse121

What value is returned from this method?

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

A. `-1`

B. `0`

C. `max-1`

D. `max`

Common Problem-Solving Strategies

- **Analogy** – Is this similar to another problem you've seen?
- **Brainstorming** – Consider steps to solve problem before jumping into code
  - Try to do an example "by hand" → outline steps
- **Solve sub-problems** – Is there a smaller part of the problem to solve?
- **Debugging** – Does your solution behave correctly?
  - What is it doing?
  - What do you expect it to do?
  - What area of your code controls that part of the output?
- **Iterative Development** – Can we start by solving a different problem that is easier?