CSE 121 Lesson 7:
Methods, Parameters, Returns

Matt Wang

Spring 2024

sli.do #cse121-7

TAs:	Andy	Anju	Archit	Arkita	Autumn	Christian
	Hannah H	Hannah S	Heather	Hibbah	Janvi	Jessie
	Jonus	Julia	Luke	Maria	Mia	Ritesh
	Shayna	Simon	Trey	Vidhi	Vivian	Gumball?

Today's playlist:
CSE 121 lecture beats 24sp

Announcements, Reminders

- Programming Assignment 1 is out, due Tuesday April 23rd
  - Start early! This one is tough!
  - Make use of the “development slides” and example code
  - Doing P1 is *also* studying for the quiz!
- R1 released yesterday, due Thursday April 25
- Quiz 0 is Thursday, Apr 25!
  - Big review opportunity: section on Tuesday, April 23
  - Optional review session on Tuesday, April 23 from 4:30-6:30 in JHN 102

(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

```
public static void example() {
    System.out.println("hello");
    int x = 3;
    for (int i = 1; i <= 10; i++) {
        System.out.print(x);
    }
}
```

i's scope

x's scope

Poll in with your answer!

sli.do #cse121-7

What will be the last line of output after this code has executed?

```
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();
}
```

A. `count is: 1`

B. `count is: 5`

C. `count is: 6`

D. `count is: 7`

Poll in with your answer!

sli.do #cse121-7

What is the output of this program?

```
public static void main(String[] args) {
  int x = 9;
  int y = 2;
  int z = 5;

  mystery(z, y, x);

  mystery(y, x, z);
}

public static void mystery(int x, int z, int y) {
  System.out.println(z + " and " + (y - x));
}
```

A. 2 and 4
9 and 3

B. 5 and -7
5 and -7

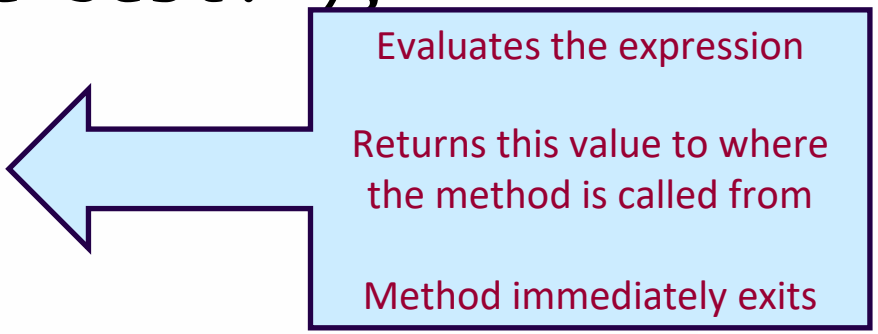
C. 9 and -3
5 and -7

D. I'm lost

(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

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

Poll in with your answer!

sli.do #cse121-7

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 #cse121-7

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

We loved your C1 reflections!

I read (skimmed?) all 199 of your responses! Some themes:

- generally, not knowing how blind people programmed
- “one minor addition and effort into making a program accessible can greatly impact the daily experience of those who need it”
- debugging is already hard – debugging without accessible error messages sounds even harder!

… with great points on "`accessiblePrinting`"

Printing out what the Turtle does is better than nothing.

But, y'all said:

- it is a ton of information – especially for complicated drawings
- the information provided might not be the "right" type
  (not precise, not high-level enough, not aware of shapes)
- **does not describe what the drawing actually is**

Food for Thought

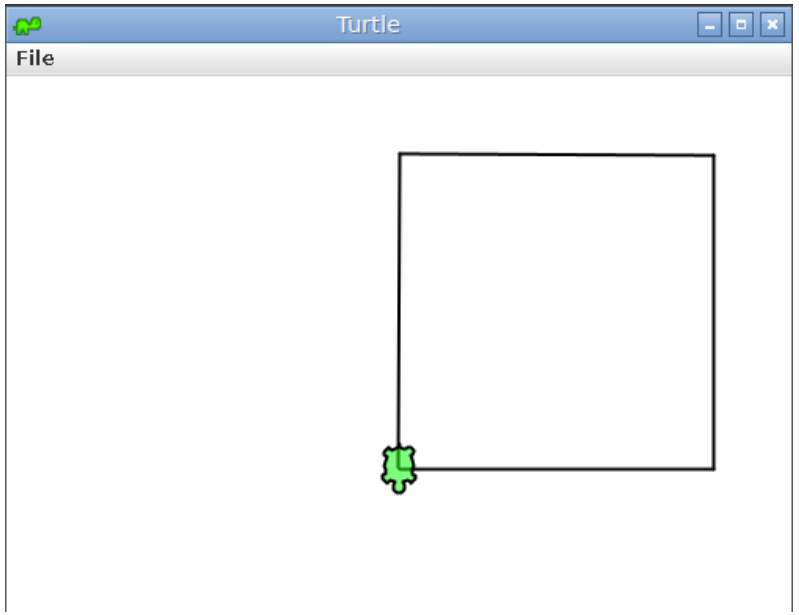
A weekly section where I introduce open problems related to our lecture topic(s) of the week.

Goals:

1. give you “conversational familiarity” with CS terminology
2. see how CS interacts with other fields and people!
3. point you in the direction of more CSE (or adjacent) classes

Note: <u>not tested content.</u> Just food for thought :)

Describing images…



Which of these best describes this image?

1. A black square drawn by a Turtle
2. An image with a green cartoon turtle overlapping with a square
3. A screenshot of the Turtle library; the toolbar says “Turtle” and has a minimize, full-screen, and close buttons. The main canvas has a 200 by 200 square, drawn by a Turtle.
4. Instruction: MOVE FORWARD 200.0 Current Pos: (200.000, 0.000) …

Describing images requires context

“A black square drawn by a Turtle”

- brief overview for someone who knows Turtle, doesn’t care about the art

“An image with a green cartoon turtle overlapping with a square”

- short description focused on the core image – no Java-Turtle context assumed

“A screenshot of the Turtle library; the toolbar says “Turtle” and has a minimize, full-screen, and close buttons. The main canvas has a 200 by 200 square, drawn by a Turtle.”

- longer caption, perhaps useful in a user design textbook or case study

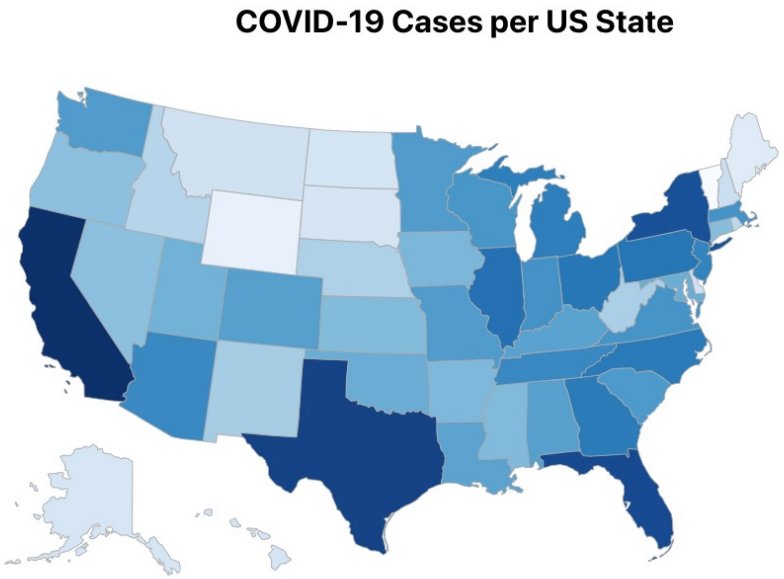
And, many other reasonable alternative texts & captions!

Active research – at UW!

Describing visualizations can be even harder!

How would you describe the visualization shown on the right?

- could read out the data as a table – 50 rows!!
- could summarize key points – loses data!
- from UW CSE + iSchool: let users decide – and ask questions about data (min, max, average)



Paper: VoxLens: Making Online Data Visualizations Accessible with an Interactive JavaScript Plug-In., Ather Sharif, Olivia H. Wang, Alida T. Muongchan, Katharina Reinecke, and Jacob O. Wobbrock. CHI '22.



Desserts for Thought

Interested in accessibility? UW is an amazing place to be!

- stellar professors in CSE – e.g. Jen Mankoff, who teaches CSE 493E: Accessibility
- amazing folks across campus – e.g. Jacob Wobbrock (iSchool) from the paper!
- people here do research, build tools, advocate for policy, and create community!

Many students mentioned that AI could be helpful. It's … complicated.

- "'Without these tools, I'd be lost': how generative AI aids in accessibility"
- "No, 'AI' Will Not Fix Accessibility"