CSE 121 Lesson 7:
Methods, Parameters, Returns

Matt Wang
Spring 2024

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?

sli.do #cse121-7

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)

```java
public static final type NAME_OF_CONSTANT = expression;
```
Definition: A value passed to a method by its caller

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

Calling a method with a parameter...
```java
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

```java
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);
```
(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

```java
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!

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

```java
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!

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

```java
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...

```java
<type> result = myMethod(42);
```
**String Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>length()</code></td>
<td><strong>Returns</strong> the length of the string.</td>
</tr>
<tr>
<td><code>charAt(i)</code></td>
<td><strong>Returns</strong> the character at index <code>i</code> of the string.</td>
</tr>
<tr>
<td><code>indexOf(s)</code></td>
<td><strong>Returns</strong> the index of the first occurrence of <code>s</code> in the string; returns <code>-1</code> if <code>s</code> doesn't appear in the string.</td>
</tr>
<tr>
<td><code>substring(i, j)</code> or <code>substring(i)</code></td>
<td><strong>Returns</strong> the characters in this string from <code>i</code> (inclusive) to <code>j</code> (exclusive); if <code>j</code> is omitted, goes until the end of the string.</td>
</tr>
<tr>
<td><code>contains(s)</code></td>
<td><strong>Returns</strong> whether or not the string contains <code>s</code>.</td>
</tr>
<tr>
<td><code>equals(s)</code></td>
<td><strong>Returns</strong> whether or not the string is equal to <code>s</code> (case-sensitive).</td>
</tr>
<tr>
<td><code>equalsIgnoreCase(s)</code></td>
<td><strong>Returns</strong> whether or not the string is equal to <code>s</code> ignoring case.</td>
</tr>
<tr>
<td><code>toUpperCase()</code></td>
<td><strong>Returns</strong> an uppercase version of the string.</td>
</tr>
<tr>
<td><code>toLowerCase()</code></td>
<td><strong>Returns</strong> a lowercase version of the string.</td>
</tr>
</tbody>
</table>
# Example of returns: Math class

<table>
<thead>
<tr>
<th>Methods</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.abs(value)</td>
<td>Absolute value of value</td>
</tr>
<tr>
<td>Math.ceil(value)</td>
<td>value rounded up</td>
</tr>
<tr>
<td>Math.floor(value)</td>
<td>value rounded down</td>
</tr>
<tr>
<td>Math.max(value1, value2)</td>
<td>Larger of the two given values</td>
</tr>
<tr>
<td>Math.min(value1, value2)</td>
<td>Smaller of the two given values</td>
</tr>
<tr>
<td>Math.round(value)</td>
<td>value rounded to the nearest whole number</td>
</tr>
<tr>
<td>Math.sqrt(value)</td>
<td>Square root of value</td>
</tr>
<tr>
<td>Math.pow(base, exp)</td>
<td>base to the exp power</td>
</tr>
</tbody>
</table>
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;
}
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
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: not tested content. Just food for thought :)

Lesson 7 - Spring 2024
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)

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”