

# CSE 11

# Accelerated Intro to Programming

## Lecture 4

Greg Miranda, Spring 2021

# Announcements

- Quiz 4 due Wednesday @ 8am
- PA1 due Wednesday @ 11:59pm
- Survey 2 due Friday @ 11:59pm
- • PA0.5 resubmission due Friday, April 16<sup>th</sup> @ 11:59pm
  - Late add/did not do
  - Grading in progress...←
    - Up to 100% credit

# Example from Lecture Quiz 3

```
class Example {  
    int average(int n, int m, int o) {  
        return (n + m + o) / 3;  
    }  
    String withDotAtTheEnd(int n) {  
        return n + ".";  
    }  
    String ans = this.withDotAtTheEnd(this.average(3, 5, 7));  
}
```

have no effect  
until we call these methods

this.withDotAtTheEnd ( this.average(3, 5, 7) );

$\Rightarrow$  this.average(3, 5, 7)  $\Rightarrow$

$(n+m+o)/3$	$n=3$ $m=5$ $o=7$
-------------	-------------------------

 $\Rightarrow$ 

$15/3$
--------

 $\Rightarrow$ 

5
---

$\Rightarrow$  this.withDotAtTheEnd ( 5 );

$n + "."$	$n=5$
-----------	-------


 $\Rightarrow$ 

$5 + "."$
-----------

$\Rightarrow$  "5."

# JShell

- The Java Shell tool (JShell) is an interactive tool for learning the Java programming language and prototyping Java code.
- The way to think about the environment of JShell:
  - Sort of inside a class
    - Can start writing field definitions and trying things out
  - Good tool for experimentation
    - Can write one field definition or method definition at a time


- Make a String
    - String h = "hello";
      - JShell immediately prints out the string
    - String h2 = "he" + "llo";
      - Evaluates the expression, shows us the value
  - Methods already defined by Java that we can use
    - String – is built-in Java class (i.e. already defined in Java)
      - Defines many methods
- 

- String myName = "Greg";
- int nameLen = myName.length();
  - Note: these method calls are using something other than this
    - We can call methods on many different kinds of values in Java
    - When we define a method within a class and call that method from within the class
      - Then we use this. to refer to methods within the class *→ optional*
    - When call a method that's in another class
      - We use a particular value and then use that method
        - That method is going to be able to use information about that class to get its answer- length() – does something different depending on which value it's called from

- Other String methods:

- String myFullName = "Gregory Joseph Miranda";

- String middle = myFullName.substring(8, 14);


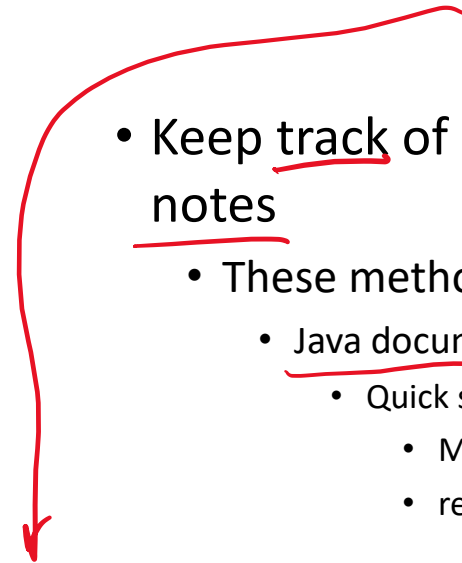
- What did the method substring() do? 

"Joseph"




- length() and substring()
  - 2 methods defined on Java's built-in String class
  - Can use them to do different types of calculations with String
- A bunch more String methods to come...
- Main point:
  - String value – can use these existing methods to do this calculations
- 2<sup>nd</sup> big lesson:
  - Indexes – indexing into Strings to access the characters
    - Something we will be working with as we go forward

→ starts at 0



- Another String method:
    - `String myWeirdName = myFullName.replace("e", "WEIRD");`
  - What did replace() do? 
  - What's the value of myFullName after calling replace()?
  - Keep track of the String methods you learned about in your own notes
    - These methods are all written down online
      - Java documentation – we would be able to see all these methods
        - Quick search: Java string documentation
          - Many String methods we could use
          - `repeat()`
- 

*Strings are immutable → cannot change a String*

- String helloTwice = h.repeat(2);
- String manyHello = h.repeat(20);
- What if we want to find if another String appears in a String, like a search?   

- int index = myFullName.indexOf("Joseph");
- What if the String is not in my name?
  - int anotherIndex = myFullName.indexOf("Orange");
  - What happened?

- 0+ – index where we found the String → *first occurrence*
- -1 – didn't find the String

- Just a few more String methods

- Working with the idea that there is built-in stuff in Java that we are going to be able to use
  - This will help us write interesting programs that work with and manipulate text

- String example program

```
class StringExamples {  
}
```

- Write a method called firstHalf that:

- Takes a String and returns a new String that has just the first half of the characters from the input String *(rounding down # of chars)*

- When writing a method:

- Think about what some examples are and what we expect the results to be:
    - We can write these down as fields
    - Then we can easily check if we are right after running the program
  - Examples first – then build up into the implementation
    - Do on paper/whiteboard first – then type them in

- One of the first things to think about is:
  - What method (or methods) out of the methods we saw on strings is going to be useful here
    - We will be able to accomplish this only with methods we have seen so far

- This showed us how to implement a method from a word problem prompt
- We thought through some examples
  - Which helped us to refine our understanding
- We experimented a little bit
  - Figured out we are okay with this empty String result
- This is the process we should use when implementing methods
  - i.e. Programming Assignments