Welcome to CSE 121!

Simon Wu

Summer 2024

Use this QR code as one way to ask questions!

sli.do #cse121

TAs:	Trey	Hannah	Mia	Vivian	Jolie	Colton	Ziao







Announcements & Reminders

- **Congrats on finishing quiz 0 yesterday!**
  - **the first CS quiz you take can definitely be challenging**
  - **we will try to be kinder on quiz 0 grading**
  - **reminder that we drop the lowest 3 exam grades**
  - **quiz grades will be released next Friday**
  - **quizzes are one small part of your overall grade**

- **P1 released later today, due next Thursday 7/18 @ 11:59**

P1 Reminders

- **<u>Please start early!</u>** P1 is a big step up from previous assignments
  - P1 has a *long spec*, worth reading it tonight or tomorrow (and completing the spec quiz to test your understanding!)
  - The dev slides are there to help you (particularly for P1)!
  - Please check out our **Golf Game example**
- We strongly recommend you to **not** use methods for P1
- Ask for help early!

Getting Extra Help (part 1)

- Course structure
  - Less lectures in Summer, so lecture moves faster
  - Reminder: lecture is designed with PCMs in mind
  - Reminder: section will review lecture content
- Extra resources and practice with (nested) for loops:
  - Pre-class readings (and the pre-class walkthrough video)
  - (Starred) section problems
  - Simon will record extra nested for loops walkthrough video for P1, releasing on Monday

Getting Extra Help (part 2)

Office Hours

- TA Office Hours at the IPL (has been very quiet so far!)
- Virtual office hours
  - Thursdays and Weekends
  - found on the same OH schedule linked on our course website
- Simon's office hours in CSE1 214
  - Wednesdays from 2:30 – 4:30
  - Fridays from 3:30 – 4:30
  - or by appointment (email me: simonswu@uw.edu)

Getting Extra Help (part 3)

Mid Quarter check-ins

- meet 1-on-1 with Simon to discuss goals and struggles with class
- Google form to request check-ins will be posted to Ed later tonight

Bottom line: **please ask for help if you need it!**

- It's expected that students struggle with content and find support through our course resources
- Ask sooner than later, since all course knowledge builds on itself

(Last Time) Methods

Writing our own **methods** allow us to define our own statements / commands in Java!

- Naming conventions for methods are the same as variables: `camelCased`

```
public static void myMethod() {
    /***
    Your code here
    **/
}
```

(Last Time) Example Method

```
public static void main(String[] args) {
  lineOf5();
}

public static void lineOf5() {
   for (int i = 1; i <= 5; i++) {
      System.out.print("*");
   }
   System.out.println();
}
```

(Last Time) Parameters

Definition: A value passed to a method by its caller

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

Calling a method with a parameter…

```
myMethod("Poke"); // Poke is the best!
```

(Last Time) Example Method with Params

```
public static void main(String[] args) {
  lineOfStars(10);
}

public static void lineOfStars(int numStars) {
   for (int i = 1; i <= numStars; i++) {
      System.out.print(“*”);
   }
   System.out.println();
}
```

(Review) Scope in For Loops

- Definition: The part of a program where a variable exists (and can thus be referenced/modified/used).
  - From its **declaration to the end of the { } braces**
  - Ex: a variable declared in a for loop only exists <u>in that loop</u>!

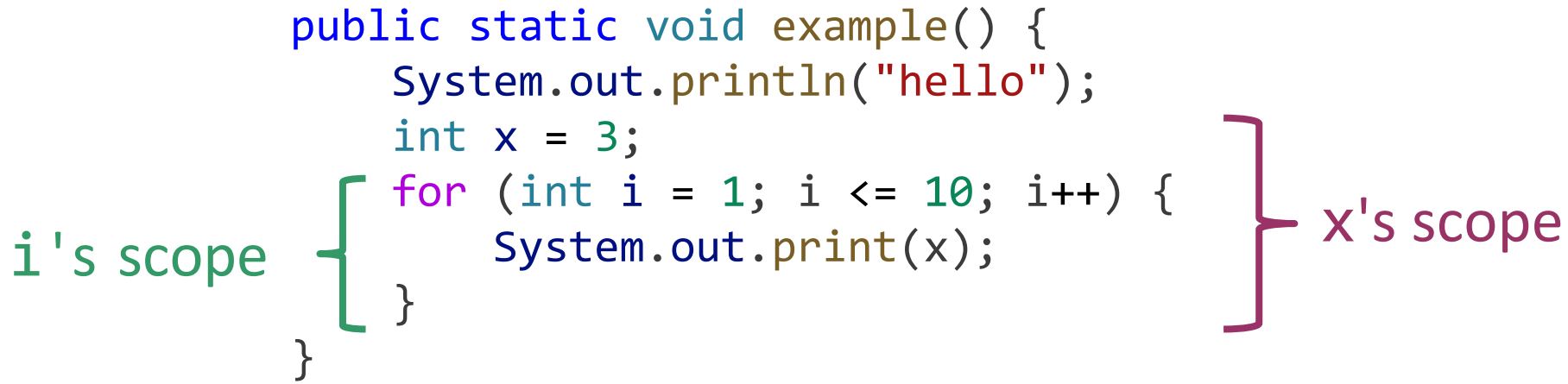
innerloop's scope

```
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

Scope in Methods

- Definition: The part of a program where a variable exists (and can thus be referenced/modified/used).
  - From its **declaration to the end of the { } braces**
  - Ex: a variable declared in a method exists only in that method!



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)

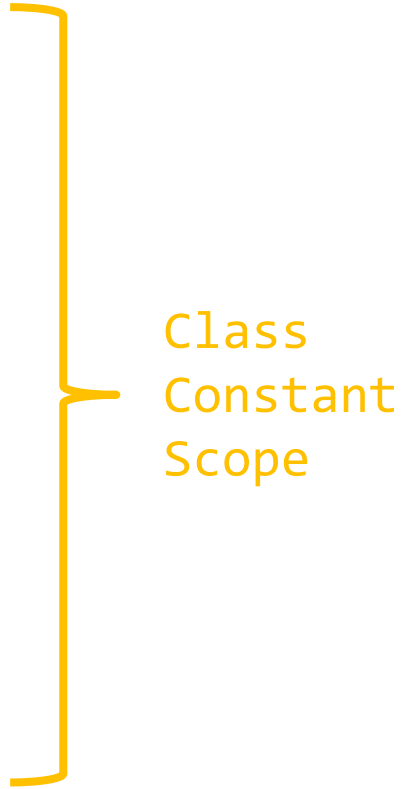
```
public static final type NAME_OF_CONSTANT = expression;
```

Class Constant (example)

```
public static final String VOWELS = “aeiou”;
public static final int START_LOWER_ASCII = 97; // (int) ‘a’
public static final int END_LOWER_ASCII = 122; // (int) ‘z’

public static void main(String[] args) {
    printVowels();
}

public static void printVowels() {
    System.out.println(“vowels are:”);
    for (int i = 0; i < VOWELS.length(); i++) {
        System.out.println(“ - ” + VOWELS.charAt(i));
    }
    System.out.println(“(and sometimes y)”);
}
```



Method Comments!

- Now that we know how to write methods, we have a new form of documentation (using comments) to write.
- Each method you write (except for main) should be accompanied by a short comment that describes what it does.
- **Be sure to comment on method behavior, and all parameters and returns of a method!**

```
// Randomly generates an addition problem where the
// operands are in the range 1-10 (inclusive), and prints the result
// rounded to two decimal places.
public static void addTwoRandomNumbers() {
    Random randy = new Random();
    int num1 = randy.nextInt(10) + 1;
    int num2 = randy.nextInt(10) + 1;
    int sum = num1 + num2;
    ...
}
```

Poll in with your answer!

sli.do #cse121

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`

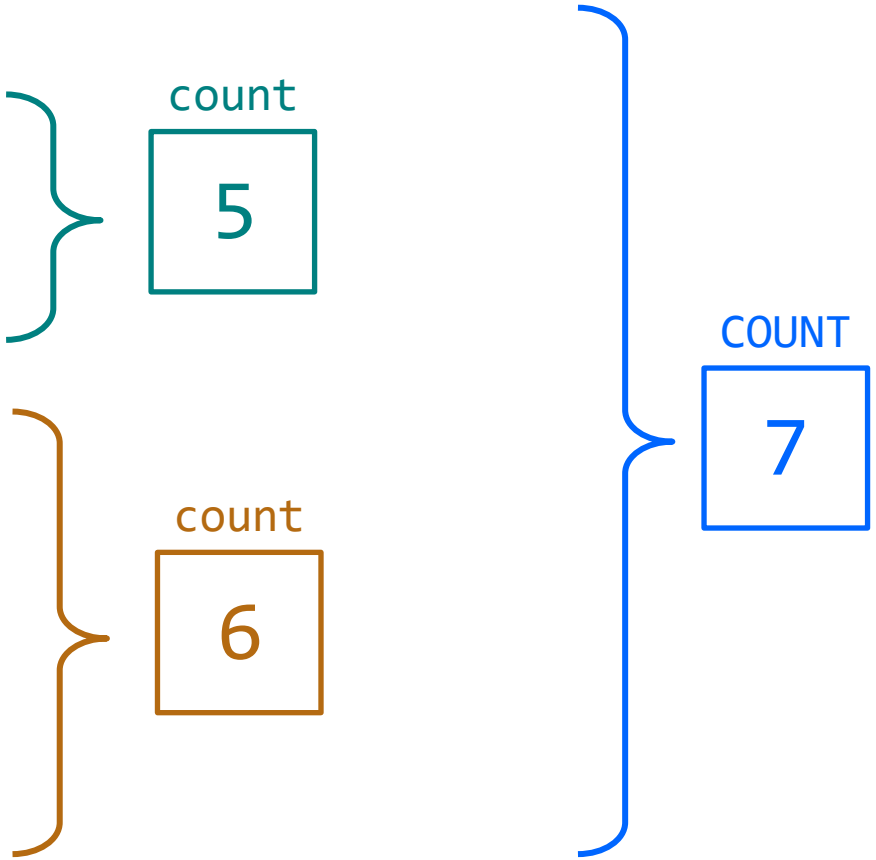
C. `count is: 6`

D. `count is: 7`

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();
}
```



Poll in with your answer!

sli.do #cse121

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

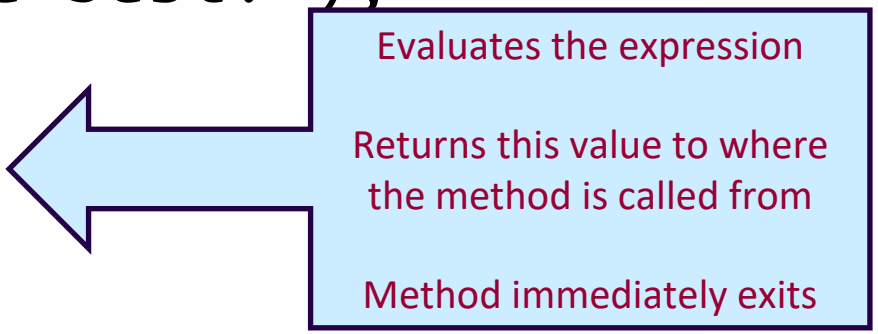
C. 9 and -3
5 and -7

D. I'm lost

(Last Time) 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);
```

Returns (example)

```
public static void main(String[] args) {
    funkyMath(1, 2); // calls the function, but we lose the return D:

    int funkyMathResult = funkyMath(1, 2); // 1 * 2 + 2 * 3 = 2 + 6 = 8
    System.out.println(funkyMathResult); // 8
}

public static void funkyMath(int a, int b) {
    return a * 2 + b * 3;
    // java will throw an error if any more code is written here!
}
```

(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";
```

Poll in with your answer!

sli.do #cse121

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

Poll in with your answer!

sli.do #cse121

What will be the **output** when the following code is executed?

```
public static void main(String[] args) {
  int startNum = 2;
  favoriteNumber(startNum);
  System.out.println(startNum);
}

public static int favoriteNumber(int startNum) {
  int newNum = 0;
  for (int i = 1; i <= startNum; i++) {
    newNum += startNum;
  }
  return newNum;
}
```

A. 0

B. 2

C. 4

D. None of the above

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;
}
```