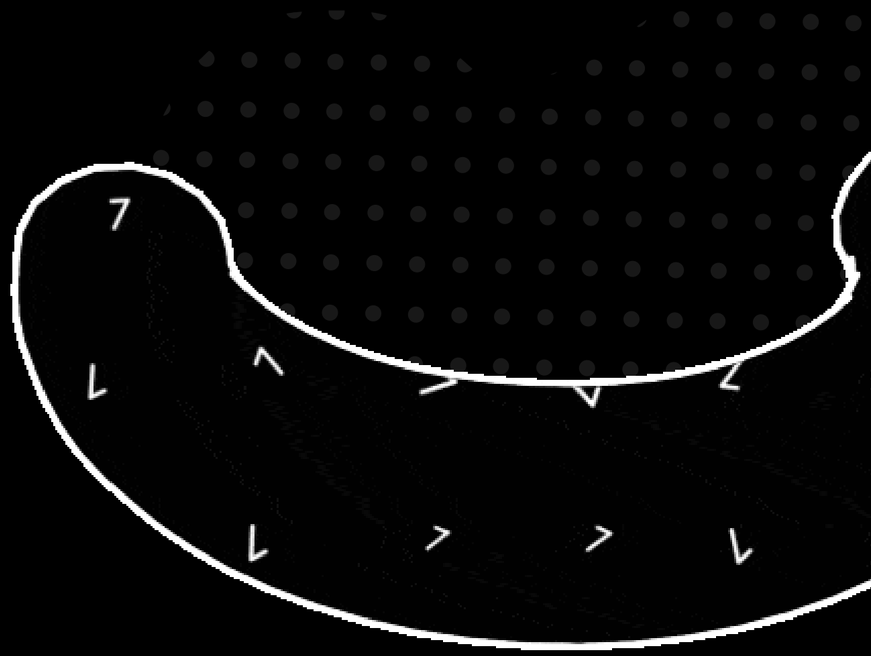
Discussion Section #2

Omeed Tehrani

CS 312 - Section 52195

University of Texas at Austin

February 1st, 2021



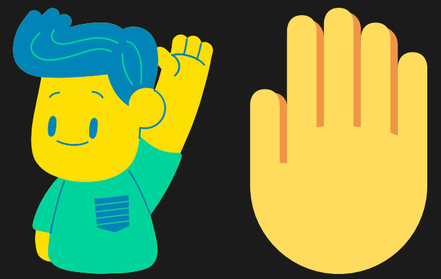
Order of Business

- Office Hour Logistics / Updates
- Participation Quiz #2
- Review Quiz as a group
- Best Practices for Style
- Assignment #2 Overview
- Remaining available time:
  - Open-Ended Discussion
  - Q&A
  - Advice

Updates / Information 

- Reminder on how to use office hour queue:
  - Navigate to => https://cs312.utcshelphours.com/
  - Join the Zoom meeting corresponding to the TA's time when you are in line or near the front of the line.
  - Make sure to remove yourself from the queue if you do not plan on showing up.
- **My Office Hours:** Tuesday and Thursday from 6:00 pm to 7:00 pm.
- Meetings by appointment
- Mental Health and Wellness
- Answering Chat Questions - Addressing Discussion #1!

Participation Quiz #2



Instructions:

- Please navigate to **Canvas --> Assignments --> Participation Quiz 2**
- Press and view the PDF.
- Read thoroughly and write down solutions
  - Recommendation: Writing the code on paper would be good practice for exams in this course and 314 (Data Structures).
  - Typing is permissible
- Upload to Canvas before the deadline
  - Preferably today (so you don't forget!)
  - GeniusScan, AdobeScan, etc.
- Additional Notes: When submitting, please try to avoid extensive blurriness and difficult to read angles. Please make sure you are submitting proper file types.

You will have about **25 minutes** before we come back together as a group.

Good luck!

Generally operators evaluate left-to-right.

// problem #1

Precedence:

1. **()**
2. ***, /, %**
3. **+, -**

```
System.out.println( 3 * 4 / 5 + 2 * 3 );
```

Solution:

// problem #1

3 * 4 = 12

2 * 3 = 6

Now, let's start evaluating from left to right:
- 12 / 5 = 2
- 2 + 6 = **8**

```
System.out.println( 3 * 4 / 5 + 2 * 3 );
```

Generally operators evaluate left-to-right.

Precedence:

1. **()**
2. ***, /, %**
3. **+, -**

// problem #2

```
System.out.println( 2.5 * 2 / 4 + 2 + "CS" + 1 + 2);
```

Solution:

// problem #2

2.5 * 2 = 5.0

5.0 / 4 = 1.25

1.25 + 2 = 3.25

3.25CS12

```
System.out.println( 2.5 * 2 / 4 + 2 + "CS" + 1 + 2);
```

Generally operators evaluate left-to-right.

Precedence:

1. **()**
2. ***, /, %**
3. **+, -**

// problem #3

```
System.out.println( 1.5 + 7 % 4 + 1.5 * 4 / 3 );
```

Solution:

// problem #3

7 % 4 = 3

1.5 * 4 = 6.0 6.0 / 3 = 2.0

1.5 + 3 = 4.5

4.5 + 2.0 = 6.5

```
System.out.println( 1.5 + 7 % 4 + 1.5 * 4 / 3 );
```

What is the output by the following code?

// problem #4

```java
int x = 3;
int y = 2;
x++;
x = x * 4;
y = y * 2 + y - 1;
x = x % 7;
System.out.println("x is : " + x + " y is : " + y);
```

// problem #4

```java
int x = 3; // Ok, we know x = 3.

int y = 2; // Ok, we know y = 2.


x++; // aha! Let's increment x. x is now 4.

x = x * 4;
// hmm. looks like x is being updated.
// lets take the previous x = 4, then
// multiply it by 4. x = 4 * 4 = 16.

y = y * 2 + y - 1;
// alright, now lets update our y. Don't forget precedence!
// y = 2 * 2 + 2 - 1
// 4 + 2 = 6 - 1 = 5
// y is now equal to 5!

x = x % 7;
// lets take that previous x (which was 16), and plug in!
// x = 16 % 7 = 2
// x is now equal to 2!

// final product: x = 2 and y = 5.
// lets print it!
System.out.println("x is : " + x + " y is : " + y);
// x is : 2 y is : 5
```

What is the output by the following code?

// problem #5

```
for (int i = 0; i < 5; i++) {
    System.out.print(i * 5 + 1 + " ");
}
```

```java
// How would we interpret this loop?
    // 1. We start at 0.
    // 2. Bounds are zero inclusive, 5 exclusive.
        // i = 0, 1, 2, 3, 4
    // 3. Incrementing by one.
for (int i = 0; i < 5; i++) {

    // hmm.. looks like this isn't a println.
    System.out.print(i * 5 + 1 + " ");

    // i = 0? => 0 * 5 = 0 + 1 = 1
    // i = 1? => 1 * 5 = 5 + 1 = 6
    // i = 2? => 2 * 5 = 10 + 1 = 11
    // i = 3? => 3 * 5 = 15 + 1 = 16
    // i = 4? => 4 * 5 = 20 + 1 = 21
    // i = 5? => hmm. Looks like 5 < 5 is false. Exit loop.

    // Final output: 1 6 11 16 21
}
```

// problem #5

Can anyone tell me what "" vs. " " is?

What is the output by the following code?

// problem #6

```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 5; j++) {
        for (int k = 1; k <= 8; k++) {
            System.out.print("*");
        }
        System.out.print("*");
    }
    System.out.print("*");
}
System.out.print("*");
```

```java
// NOTE: PAY ATTENTION TO LOOP BRACKETS! (aka. the bounds)
// executes from 1 inclusive to 10 inclusive.
// i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
for (int i = 1; i <= 10; i++) {
    // executes from 1 inclusive to 5 inclusive.
    // j = 1, 2, 3, 4, 5
    for (int j = 1; j <= 5; j++) {
        // executes from 1 inclusive to 8 inclusive.
        // k = 1, 2, 3, 4, 5, 6, 7, 8
        for (int k = 1; k <= 8; k++) {
            System.out.println("*");
        }
        System.out.println("*");
    }
    System.out.println("*");
}
System.out.println("*");

// Let's lay out the order of loop operations here:
    // The third loop prints 8 astericks.
    // Another asterick is printed for the second loop.
    // This is repeated 5 times because of the second loop.
    // Then this is repeated 10 times because of the outer loop.
// 8 + 1 = 9
// 9 * 5 = 45
// 45 * 10 = 450
// 450 + 10 = 460
// 460 + 1
// ANSWER: 461
```

// problem #6

Tip on practicing nested loops: In your IDE, place println()'s and mark them to represent the loops corresponding position in the nesting. Write it on paper and then check your answer in the output.

Would anyone like to share their approach or solution?

// problem #7

7. Write a Java method to produce the following figure. The figure relies on a program constant named `SIZE`. `SIZE` is an `int`.
When `SIZE = 1`, the output is:

```
**
****
```

When `SIZE = 2` the output is

```
**
****
******
********
```

When `SIZE = 3` the output is

```
**
****
******
********
**********
************
```

Look for patterns before you design your algorithm!
Hmm... line number is changing?

// problem #7

```
// we are working off the assumption that size has already been declared.
public static void figure() {
    final int NUM_LINES = SIZE * 2;
    for (int line = 1; line <= NUM_LINES; line++) {
        final int NUM_STARS = line * 2;
        for (int i = 0; i < NUM_STARS; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

KEY POINTS:

- It is all about looking for patterns and building algorithms to represent those patterns!
- We did "final int" for a reason! Avoid having magic numbers, this is a common way that students lose points on Assignments.

Assignment #1 Update / Common Areas for Improvement

- Assignment #1 Grades will be released either tonight or tomorrow morning.
  - Lots of feedback on this first assignment!
- Common stylistic mistakes:
  - Not filling slip days or incomplete header.
  - Not leaving space between methods or inconsistent spacing. This is incredibly important in terms of readability.
  - Method names lacking descriptiveness.
  - No commenting throughout the entire code or partial commenting.
  - Autoformatter will be your best friend!
- Overall, fantastic job across the board! It was clear that everyone put in a great deal of effort. :)

Assignment #2 - Tips

Credits to Santiago Moreno for some of these tips!

- Start early & come to Office Hours!
- Read instructions VERY carefully on the website.
- Use class constants:
- public static final int YOUR_CONSTANT = 10;
- Use for loops to repeat code a specific number of times
- Structural decomposition is important! Break things down to components
- Figure out patterns and possible methods on paper before writing code!
- Comments should add value, not be redundant
- Comments should now extend to for loops, this will help you debug later
- If you can’t describe it, you don’t fully understand what your code is doing!

Assignment #2 - Common Style Tips

Credits to Santiago Moreno for some of these tips!

- Variable names for your for-loops should be descriptive, not just one letter:

```
for (int rowNum = 0; rowNum < NUM_ROWS; rowNum++) {

}
```

- Have spaces between your mathematical operators, not all bunched together:

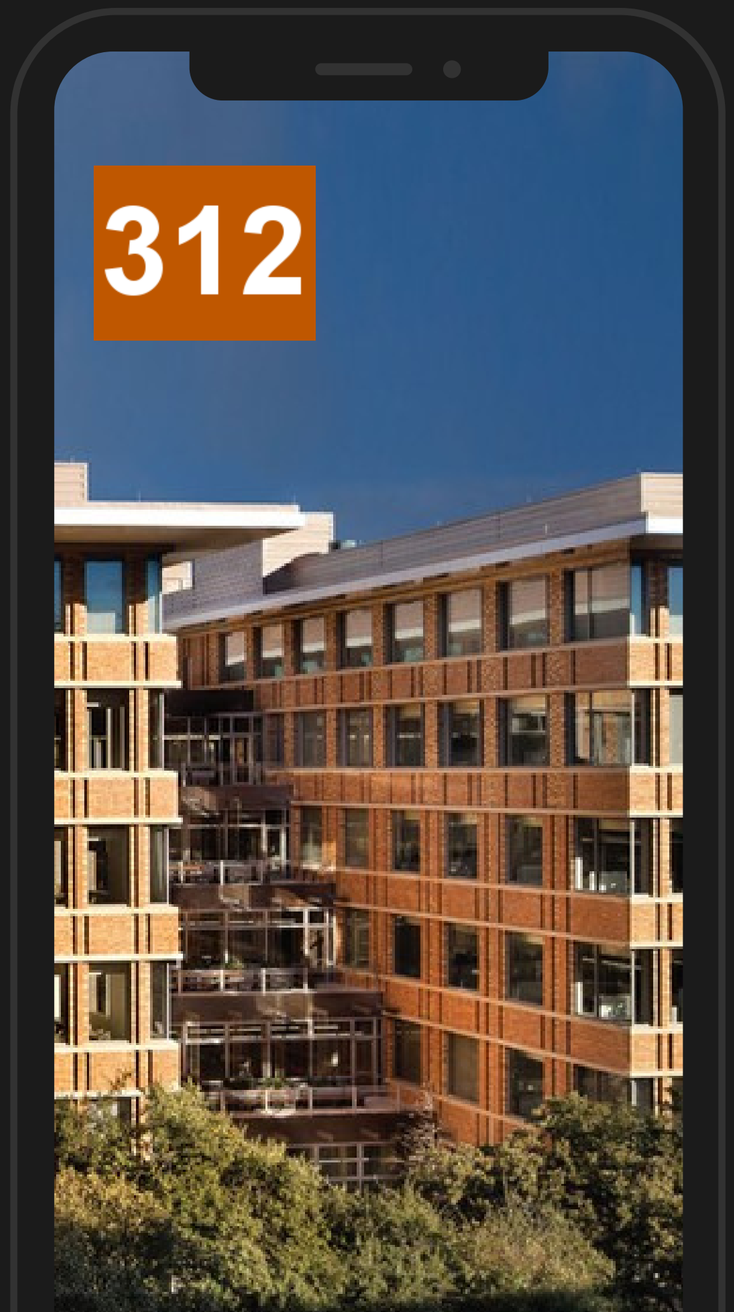
```
int fahrenheitGood = ((celsius * 9) / 5) + 32; // good!
int fahrenheitBad = ((celsius*9)/5)+32; // bad!
```

- Give it a descriptive variable name and use the appropriate constant type!
- Class constant vs. method constant scope
- Multiple println()'s to construct the tower is not sufficient.
- This is a fantastic source for referencing when you are coding: https://www.cs.utexas.edu/~scottm/cs312/handouts/IntroJavaStyleGuide.pdf

Have a great day!

I'm not a great programmer; I'm just a good programmer with great habits.

- Kent Beck (American software engineer)



Q&A?