# CS 0007 RECITATION – 10/7/21

LIN ROJTAS – 10:00A – 10:50A

## OVERVIEW

- Midterm debrief
- Loops!
  - While loops
  - Do while loops
  - For loops
- Lab 4 hits and Q&A

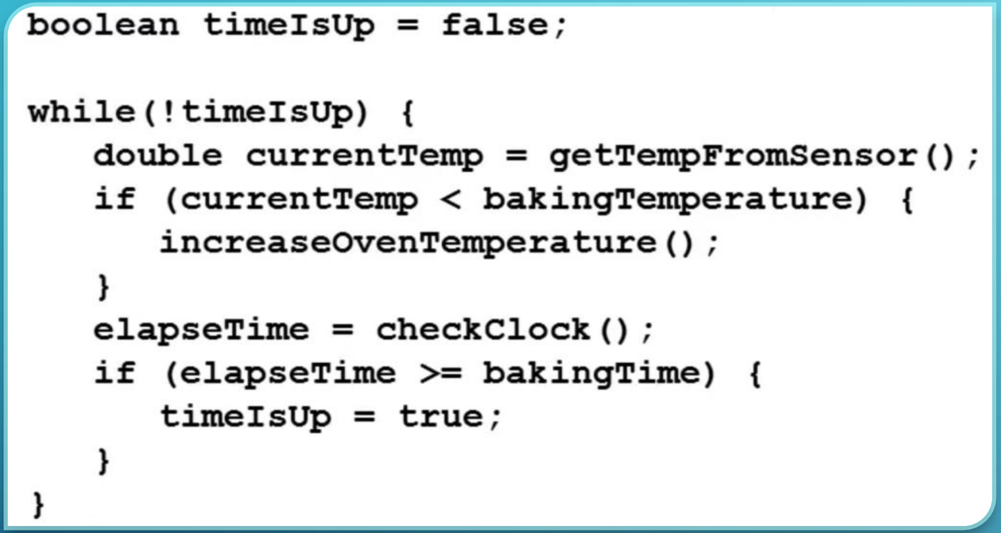
## MIDTERM…

- How was it?
- I'm not sure when they'll be graded, but the grader is in the process! There's a good amount of you, so be patient
- When you get your grades back, direct questions to the grader (Mike Neumann) or Paulo

## LOOPS

- Exactly what it sounds like!
- Avoids the need to write the same block of code over and over again
- Loops will check to see if a condition is true and execute the code inside of it any number of times depending on whether the condition stays true
  - Almost like an if statement if it ran over and over…
- Three types
  - While loop
  - Do-while loop
  - For loop

## WHILE LOOPS



```
boolean timeIsUp = false;

while(!timeIsUp) {
   double currentTemp = getTempFromSensor();
   if (currentTemp < bakingTemperature) {
      increaseOvenTemperature();
   }
   elapseTime = checkClock();
   if (elapseTime >= bakingTime) {
      timeIsUp = true;
   }
}
```

- In plain English: “While this condition is true, do this action”
- To unpack this example from the videos
  - Our condition is checking whether the time on the oven is up or not
  - **While** it isn’t, we elapse time and check to see if the elapsed time is equal to the baking time
    - If they’re equal, then we set timeIsUp to true.
    - The loop will stop when it goes back to check whether the while clause is true; since timeIsUp is true, !timeIsUp will be false, so we exit the loop

## DO-WHILE LOOPS

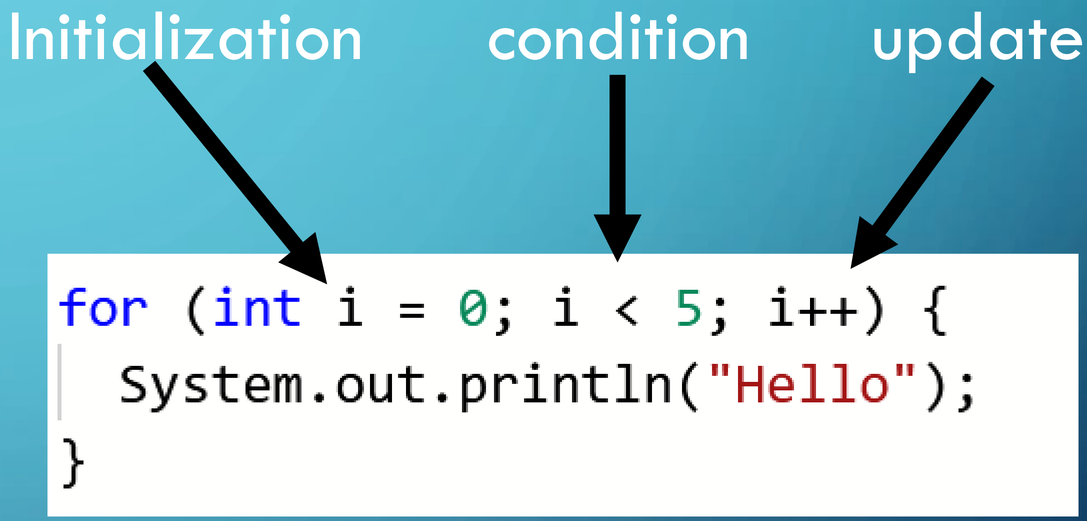
- In plain English: “Do this action, and while the condition is true, do it again”
- Essentially… the same as a while loop, except the code inside of the loop always runs **at least once**
- The loop on the left will print the numbers 0 and 1 (because it will always run at least once, and the second time it runs, the condition at the end is true), but the loop on the right will print nothing because the condition at the beginning is false

```java
i = 0;
do {
  System.out.println(i);
  i = i + 1;
} while (i == 1);
```

```java
i = 0;
while (i == 1) {
  System.out.println(i);
  i = i + 1;
}
```

# FOR LOOPS

- These are a little trickier…
- There are three clauses in a for loop:
    - Initialization
    - Condition
    - Update

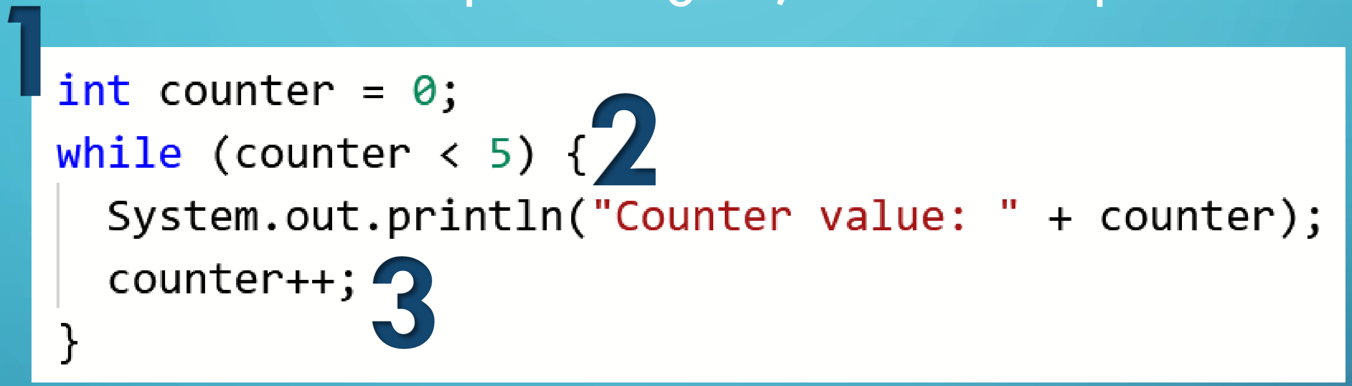


```
for (int i = 0; i < 5; i++) {
  System.out.println("Hello");
}
```

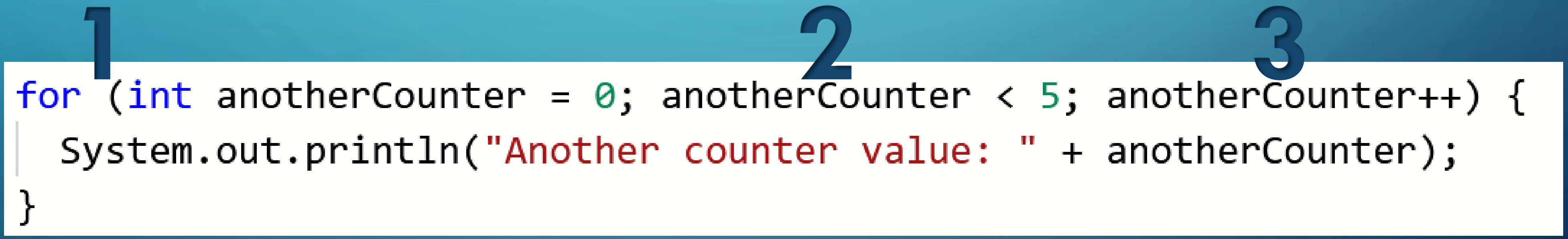
- The convention when initializing a variable is to use the letter `i` as your variable name – it's essentially the only time you should be using single-letter variables!
- The above for loop will print "Hello" five times

## FOR LOOPS

- A bit harder to translate into plain English, so let's compare to a while loop



```java
int counter = 0;
while (counter < 5) {
  System.out.println("Counter value: " + counter);
  counter++;
}
```



```java
for (int anotherCounter = 0; anotherCounter < 5; anotherCounter++) {
  System.out.println("Another counter value: " + anotherCounter);
}
```

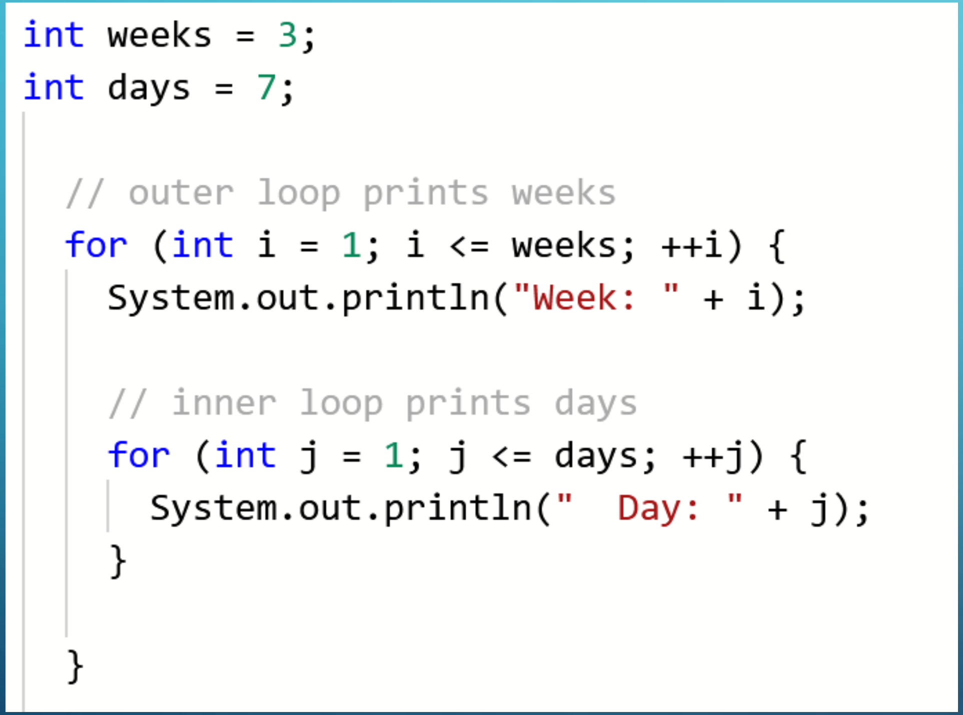
- Initialize a variable (1), create a Boolean condition (2), increment the variable (3)

## NESTED LOOPS

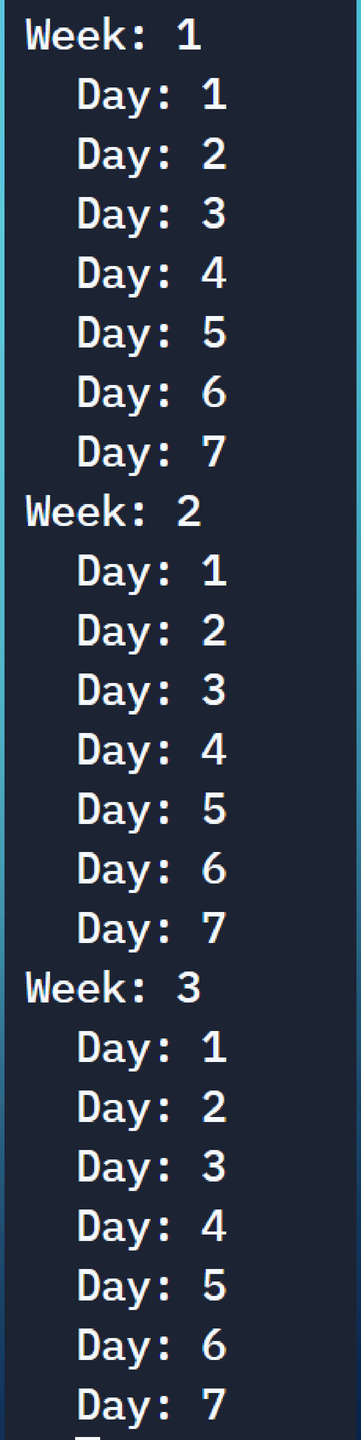
- Think nested if statements, but with loops!
  - A loop within a loop (within a loop, within a loop…)
- These tend to be harder to debug since it's two loops that can potentially go haywire and never stop… more on infinite loops in a bit
- The convention for nested for loops specifically is to have your outer for loop's variable be `i` and your inner for loop's variable be `j`

## NESTED LOOPS

- An example and its output…



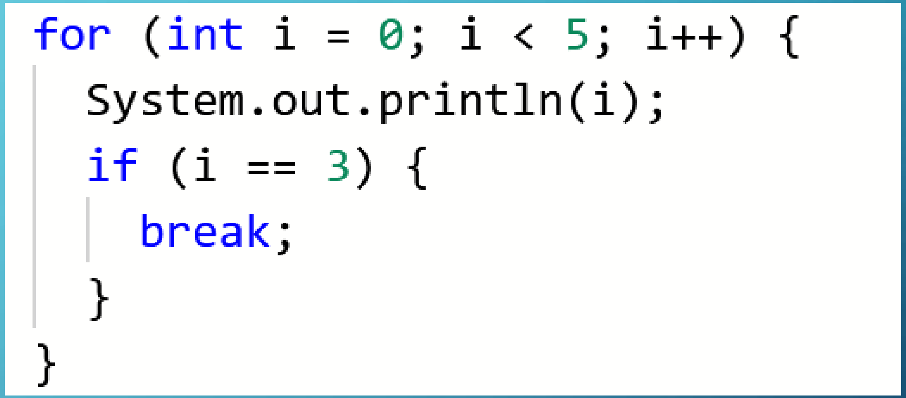
```java
int weeks = 3;
int days = 7;

  // outer loop prints weeks
  for (int i = 1; i <= weeks; ++i) {
    System.out.println("Week: " + i);

    // inner loop prints days
    for (int j = 1; j <= days; ++j) {
      System.out.println("  Day: " + j);
    }

  }
```



```
Week: 1
  Day: 1
  Day: 2
  Day: 3
  Day: 4
  Day: 5
  Day: 6
  Day: 7
Week: 2
  Day: 1
  Day: 2
  Day: 3
  Day: 4
  Day: 5
  Day: 6
  Day: 7
Week: 3
  Day: 1
  Day: 2
  Day: 3
  Day: 4
  Day: 5
  Day: 6
  Day: 7
```

## THE BREAK KEYWORD

- Remember this from switch cases?
- The break keyword essentially “breaks” the loop
  - It stops the loop exactly where it is in its tracks; code will continue to run, but outside of the for loop
  - Very helpful in debugging infinite loops!



```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
  if (i == 3) {
    break;
  }
}
```

Outputs 0 1 2 3 AND THEN STOPS!

## LAB 4 PART 1

- This is a longer one… don't freak out if you end up with 100+ lines and lots of print statements!
- You will **not** use loops in this part of the lab, just if-structures
- Pay attention to how the user could possibly input things; String methods will be your friend here
- DO NOT hard-code in the user's answer to the quiz. You'll need to use the scanner class to actually take user input!

## LAB 4 PART 2

- Loop time!
- Each activity in this lab builds upon itself
  - first you'll randomly generate a number and print it
  - Then you'll prompt the user to guess the secret number
  - Then you'll check to see if the input is valid (i.e. only accept numbers 1 through 10)
  - Then you'll add a maximum number of guesses
- Try not to get trapped in infinite loops… (the `break` keyword will be useful!)

## FOR NEXT WEEK

- Next time: possibly more on loops, reading from text files, arrays…?
- These labs are hefty; start early and do not hesitate to reach out (email or Discord) if you need help with anything regarding the labs!
- **Please email me or message me on Discord if you're having issues with submitting labs on time**
- We made a Discord server! https://discord.gg/23weGMFk
  - Joining is optional, but it'll be a good point of contact with us and hopefully a place to make some friends ☺
- WEAR A MASK AND BE SAFE!