

LEC 05

CSE 121

Nested Loops, Random, Math

Questions during Class?

Raise hand or send here

sli.do #cse121



BEFORE WE START

*Talk to your neighbors:**What's your favorite dessert?*Music:  [CSE 121 26wi Lecture Tunes](#) **Instructor:** Miya Natsuhara

TAs:	Amogh	Hayden	Anum	Sam	Shayna
	William	Aki	Abdul	Ethan	Jesse
	Johnathan	Spencer	Janvi	Jessica	Minh
	Anant	Savannah	Navya	Paul	Cayden
	Reese	Tamsyn	Ruslana	Carson	

Agenda

- **Announcements, Reminders**
- String Traversal Review
- Nested for Loop Practice
- Random, Math Practice

Announcements, Reminders

- **C1** is out, due Tuesday January 27th
- **Resubmission Cycle 0 (R0)** released, due Thursday January 29th
 - Eligible for resubmission: C0 & P0
- **Quiz 0** is on Thursday, January 27th (in your registered quiz section)
 - Read the [quiz logistics post](#) on Ed!
 - can't make it? email mnats@cs.washington.edu ASAP
- Observation: the course picks up pace a bit in this next week!
 - Go to section!
- Support reminders:
 - Miya's OH: Mon 10am-11:30am, Fri 1pm-2pm
 - IPL: Mon-Thu 12:30 – 9:30, Fri 12:30 – 6:30
 - Async via Ed & email!

Agenda

- Announcements, Reminders
- **String Traversal Review**
- Nested for Loops
- Random, Math

Wed PCM: String Traversals

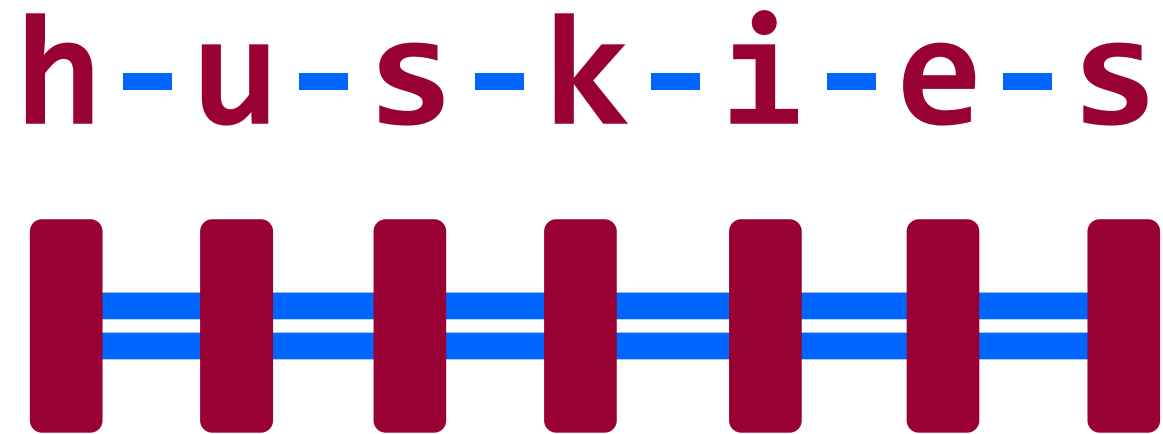
```
// For some String s  
for (int i = 0; i < s.length(); i++) {  
    // do something with s.charAt(i)  
}
```

Go Huskies?

h-u-s-k-i-e-s

The Fencepost Pattern

Some task where one piece is repeated n times, and another piece is repeated $n-1$ times and they alternate



Agenda

- Announcements, Reminders
- String Traversal Review
- **Nested for Loops**
- Random, Math

Wed PCM: for Loops!

For loops are our first **control structure**: a syntax *structure* that *controls* the execution of other statements.

```
for ( initialization ; test ; update ) {  
    body (statements to be repeated)  
}
```

PCM: Nested for Loops

```
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);  
}
```

PCM: Nested for Loops, “Inner Loop”

```
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);  
}
```



Practice: Think

[sli.do](#)

#cse121

What output is produced by the following code?

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(i);  
    }  
    System.out.println();  
}
```

A. 1
12
123
1234
12345

B. i
ii
iii
iiii
iiiii

C. 1
22
333
4444
55555

D. 1
11
111
1111
11111



Practice: Pair

[sli.do](#)

#cse121

What output is produced by the following code?

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(i);  
    }  
    System.out.println();  
}
```

A. 1
12
123
1234
12345

B. i
ii
iii
iiii
iiiii

C. 1
22
333
4444
55555

D. 1
11
111
1111
11111



Practice: Think

sli.do

#cse121

What code produces the following output?

A.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(i);  
    }  
    System.out.println();  
}
```

B.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

C.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; i <= j; j++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

D.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; i++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

```
1  
12  
123  
1234  
12345
```



Practice: Pair

sli.do

#cse121

What code produces the following output?

A.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(i);  
    }  
    System.out.println();  
}
```

B.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

C.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; i <= j; j++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

D.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= i; i++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

```
1  
12  
123  
1234  
12345
```

New: Scope

Scope: the part of a program where a variable exists (and can thus be referenced, modified, or used).

- General Rule: a variable 'exists' from its **declaration** to the **closing brace of that indentation level** }

```
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);  
}
```

innerloop's scope

outerloop's scope

Agenda

- Announcements, Reminders
- String Traversal Review
- Nested for Loops
- **Random, Math**

Randomness

Having a computer generate truly random numbers is hard!

Instead, computers generate numbers that “look random” in a predictable way, using mathematical formulas

Pseudo-Randomness

Having a computer generate truly random numbers is hard!

Instead, computers generate numbers that “look random” in a predictable way, using mathematical formulas

- can use “external” variables like time, mouse position, etc.
- if we “fix” these variables, we can reproduce the same behavior
 - very important for testing!

Aside: Why Randomness?

Randomness is core to computer science. It powers (among others):

- cryptography
- computer security
- machine learning (LLMs!)

True randomness is important: if we just use math, someone can “reverse” the formula.



[LavaRand](#): CloudFlare's Wall of Lava Lamps

PCM: Random

The Random class helps us generate pseudo-random numbers.

- We create a new Random number generator by calling `new Random()`
- First, we need to **import** the Random class with `import java.util.*;`
- We can “seed” the generator to make it behave *deterministically*

Method	Description
<code>nextInt()</code>	Returns a random integer
<code>nextInt(<i>max</i>)</code>	Returns a random integer in the range <code>[0, <i>max</i>)</code> , or in other words, 0 to <i>max</i> -1 inclusive
<code>nextDouble()</code>	Returns a random double in the range <code>[0.0, 1.0)</code>



Practice: Think

[sli.do](#)[#cse121](#)

Assuming you've declared: `Random randy = new Random();`

Which of these best models picking a random card? (number between 1 and 13 inclusive)

A. `randy.nextInt()`

B. `randy.nextInt(13)`

C. `randy.nextInt(13) + 1`

D. `randy.nextInt(13 + 1)`



Practice: Pair

sli.do

#cse121

Assuming you've declared: `Random randy = new Random();`

Which of these best models picking a random card? (number between 1 and 13 inclusive)

A. `randy.nextInt()`

B. `randy.nextInt(13)`

C. `randy.nextInt(13) + 1`

D. `randy.nextInt(13 + 1)`

PCM: Math

Math.<method>(...)

Method	Description
<code>Math.abs(<i>value</i>)</code>	Returns the absolute value of <i>value</i>
<code>Math.ceil(<i>value</i>)</code>	Returns <i>value</i> rounded up
<code>Math.floor(<i>value</i>)</code>	Returns <i>value</i> rounded down
<code>Math.max(<i>value1</i>, <i>value2</i>)</code>	Returns the larger of the two values
<code>Math.min(<i>value1</i>, <i>value2</i>)</code>	Returns the smaller of the two values
<code>Math.round(<i>value</i>)</code>	Returns <i>value</i> rounded to the nearest whole number* note: need to cast result to int (it's complicated!)
<code>Math.sqrt(<i>value</i>)</code>	Returns the square root of <i>value</i>
<code>Math.pow(<i>base</i>, <i>exp</i>)</code>	Returns <i>base</i> raised to the <i>exp</i> power