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 favourite dessert?

Music: 121 25wi lecture playlist ❄️

Instructor: Matt Wang

TAs:	Ailsa	Alice	Chloë	Christopher
	Ethan	Hanna	Hannah	Hibbah
	Janvi	Judy	Julia	Kelsey
	Lucas	Luke	Maitreyi	Merav
	Ruslana	Samrutha	Sam	Shayna
	Sushma	Vivian		

Announcements, Reminders

- C1 is out, due Tuesday January 28th
- Resubmission Cycle 0 (R0) released, due Thursday Jan 30th
  - Eligible for resubmission: C0 & P0
- Quiz 0 is on Thursday, February 6th (in your registered quiz section)
  - can't make it? email Matt ASAP
- Observation: the course picks up pace a bit in this next week!
- Support reminders:
  - Matt's OHs: Mon 2:30 – 3:20, Wed 3:30 – 4:20, Fri 1:30 – 2:20
  - IPL: Mon-Thu 12:30 – 9:30, Fri 12:30 – 5:30, Sat 1:30 – 3:30
  - Async via Ed & email!

Wed PCM Review: 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)
}
```

Wed PCM Review: String Traversals

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

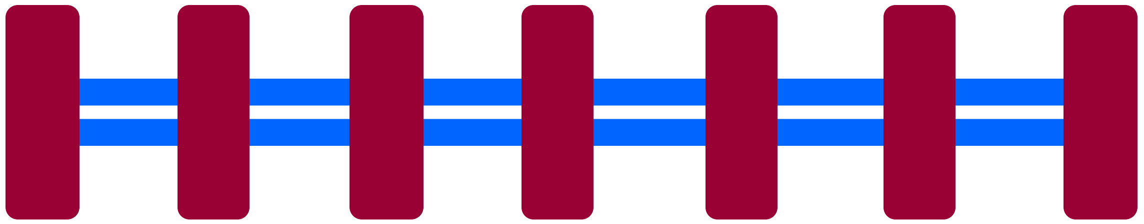
Go Huskies?

```
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

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



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, "outer 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);
}
```

PCM: Nested for loops, “inner loop”

```java
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
```

New: Scope

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

- General rule: from its **declaration to the next closing brace, }**
- a variable declared in a `for` loop only exists in that loop!
- exception: a loop variable's scope ends at that loop's closing brace

```
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

Pseudo-randomness

Having a computer generate truly random numbers is hard!

(CS folks use natural processes, e.g. atmospheric noise or lava lamps)

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 behaviour – very important for testing!

Aside: why randomness?

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

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

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

LavaRand: CloudFlare's Wall of Lava Lamps

PCM Review: Random

A `Random` **object** generates pseudo-random numbers.

- the Random **class** is found in the java.util **package**; to use, need import java.util.*;
- we can "seed" the generator to make it behave deterministically

Method	Description
`nextInt()`	Returns a random integer
`nextInt(`*`max`*`)`	**Returns a random integer in the range [0, *max*), or in other words, 0 to *max*-1 inclusive**
`nextDouble()`	Returns a random double in the range [0.0, 1.0)

Practice: Think

sli.do #cse121

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

Which of these best models picking a random card? (1-13 inclusive)

A. `randy.nextInt()`

B. `randy.nextInt(13)`

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

D. `randy.nextInt(14)`

Practice: Pair

sli.do #cse121

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

Which of these best models picking a random card? (1-13 inclusive)

A. `randy.nextInt()`

B. `randy.nextInt(13)`

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

D. `randy.nextInt(14)`

PCM Review: Math

Calling:

`Math.<method>(…)`

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

Announcements, Reminders (again)

- C1 is out, due Tuesday January 28th
- Resubmission Cycle 0 (R0) released, due Thursday Jan 30th
  - Eligible for resubmission: C0 & P0
- Quiz 0 is on Thursday, February 6th (in your registered quiz section)
  - can't make it? email Matt ASAP
- Observation: the course picks up pace a bit in this next week!
- Support reminders:
  - Matt's OHs: Mon 2:30 – 3:20, Wed 3:30 – 4:20, Fri 1:30 – 2:20
  - IPL: Mon-Thu 12:30 – 9:30, Fri 12:30 – 5:30, Sat 1:30 – 3:30
  - Async via Ed & email!