LEC 05

CSE 121

Nested Loops, Random, Math

BEFORE WE START

Talk to your neighbors:

What's your favorite dessert?

Music: CSE 121 26sp Lecture Tunes

Instructor: Matt Wang

TAs:

Abdul	Amogh	Anant	Anum	Cayden
Dalton	Ethan	Hayden	Jesse	Jessica
Johnathan	Minh	Navya	Paul	Reese
Ruslana	Sam	Savannah	Spencer	Shayna
Tamsyn	TJ	Trey		

Questions during Class?

Raise hand or send here

sli.do #cse121

Agenda (1/4)

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

Announcements, Reminders

- **C1** is out, due Tuesday April 21st
- **Resubmission Cycle 0 (R0)** released, due Thursday April 23rd
  - Eligible for resubmission: C0 & P0
- **Quiz 0** is on Thursday April 23rd (in your registered quiz section)
  - Read the quiz logistics post on Ed! (includes practice quizzes)
  - can't make it? email mxw@cs.washington.edu ASAP
- Observation: the course picks up pace a bit in this next week!
  - Go to section!
- Support reminders:
  - Matt's OH: Mon 11:30 – 12:20, 3:30 – 4:20; Wed 12:30 – 1:20
  - IPL: Tue-Thu 12:30 – 9:30, Mon/Fri 12:30 – 6:30
  - Async via Ed & email!

Agenda (2/4)

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

Wed PCM: String Traversals

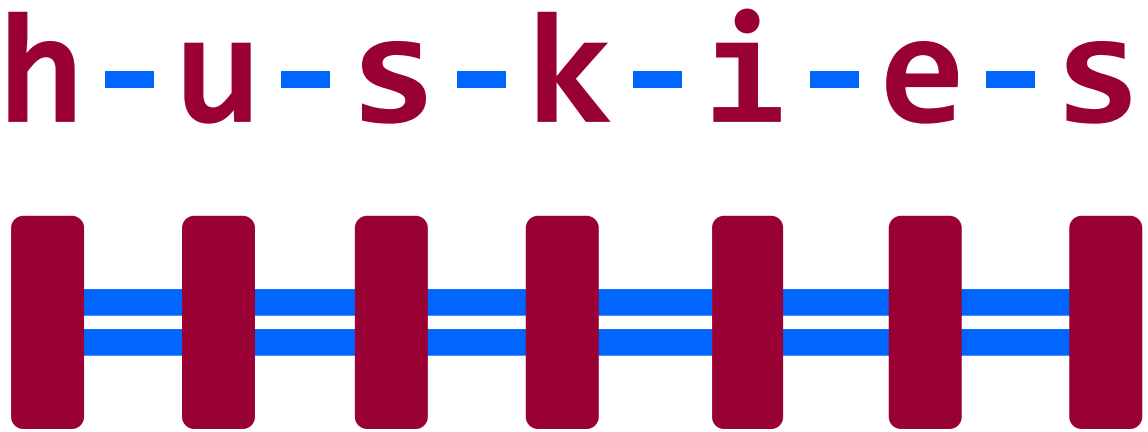
```
// For some String s
for (int i = 0; i < s.length(); i++) {
    // do something with s.charAt(i)
}
```

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 (3/4)

- 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);
}
```

inner loop

outer loop

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
```

Agenda (4/4)

- 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
`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)

PCM: Math

`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

Pseudocode

- Quite literally: “fake” code
- Goal: writing that is
  - **for humans**
  - structured *like* code
  - “easy” to convert *to* code
  - *not* beholden to the same preciseness as code

```
// Example for “name” demo
// find the space in the name
// use substring to get first, last name
// get first char of first name w/ charAt
// concatenate all to get name
```