Welcome to CSE 121!

Simon Wu

Summer 2024

Use this QR code as one way to ask questions!

sli.do #cse121

TAs:	Trey	Hannah	Mia	Vivian	Jolie	Colton	Ziao

Announcements, Reminders

- P0 was released on Wed and is due Tues, July 2nd
- Quiz 0 scheduled for July 11th (about 2 weeks away)
  - More details will be released in the coming week!
  - Prep includes practice quizzes, sections, etc.
- No July 4th section or in-person class on July 5th!
  - Enjoy your holiday weekend!
  - Lecture recording will be posted instead

Announcements, Reminders

Resubmission 0 is out!

- R0 is a “free” chance to submit C0 late
- Can submit past 3 assignments for each resub
  - Resubmit the ENTIRE assignment (including reflections!)
  - New grade REPLACES old grade
- Generally, use resubs to implement feedback on the latest assignment, but can be used to turn in late work

(PCM) Precedence (updated)

Parentheses

Logical not

Multiplication**, M**odulo**, D**ivision

Addition (and Concatenation)**, S**ubtraction

Relational operators

Equality operators

Logical and

Logical or

(Review) Variables

- Now that we know about different types and data, we can learn about how to store it!
- Java allows you to create variables within a program. A variable has
  - A type
  - A name
  - (Potentially) a value it is storing

Declaration: `int x;`

Initialization: `x = 30;`

Or all in one line:

```
int x = 30;
```

(Review) New Operators! (1/3)

```
myFavoriteNumber = myFavoriteNumber + 3;
```

This pattern is so common, we have a shorthand for it!

```
myFavoriteNumber += 3;
```

Note: this works for both numeric addition and string concatenation!

(Review) New Operators! (2/3)

The shorthands `-=`, `*=`, `/=`, and `%=` exist too!

```
myFavoriteNumber /= 3;
```

Should this work for integers? Doubles? Strings?

(Review) New Operators! (3/3)

There are even shorter operators for “incrementing” and “decrementing”!

```
myFavoriteNumber++; // equal to myFavoriteNumber += 1;
myFavoriteNumber--; // equal to myFavoriteNumber -= 1;
```

Should this work for integers? Doubles? Strings?

(PCM) Type Casting in Java

Casting – a way to convert one data type to another!

Implicit casting: integer to double (example: `30 * 1.0`)

Explicit casting: using (*typename*) syntax:

```
double pi = 3.14;
int piTrunc = (int) pi; // stores 3
```

Casting Reminders!

implicit casting can only go from "simpler" data types to more generic data types

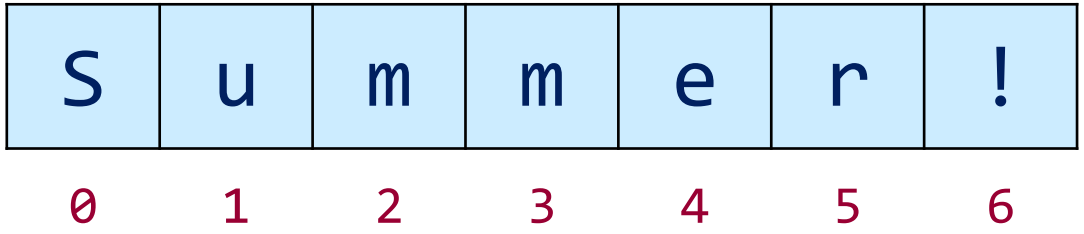
- *e.g. int to double, String or double to String*

explicit casting is a "promise" to Java that you know what you're getting yourself into!

- may be used to cast double into an int
- can also cast between int and char

(PCM) Strings and chars

- String = sequence of characters treated as one, yet can be indexed to get individual parts
- Zero-based indexing 



- **Side note**: new data type!
  `char`, represents a single character, so we use single quotes
  Strings are made up of `char`s!

```
char letter = 'c';
char num = '1';
char symbol = '%';
```

(PCM) String Methods

Usage: `<string variable>.<method>(…)`

Method	Description
`length()`	Returns the length of the string.
`charAt(i)`	Returns the character at index *i* of the string
`indexOf(s)`	Returns the index of the first occurrence of *s* in the string; returns -1 if *s* doesn't appear in the string
`substring(i, j)` or `substring(i)`	Returns the characters in this string from *i* (inclusive) to *j* (exclusive); if *j* is omitted, goes until the end of the string
`contains(s)`	Returns whether or not the string contains *s*
`equals(s)`	Returns whether or not the string is equal to *s* (case-sensitive)
`equalsIgnoreCase(s)`	Returns whether or not the string is equal to *s* ignoring case
`toUpperCase()`	Returns an uppercase version of the string
`toLowerCase()`	Returns a lowercase version of the string

(PCM) for loops!

For loops are our first *control structure*

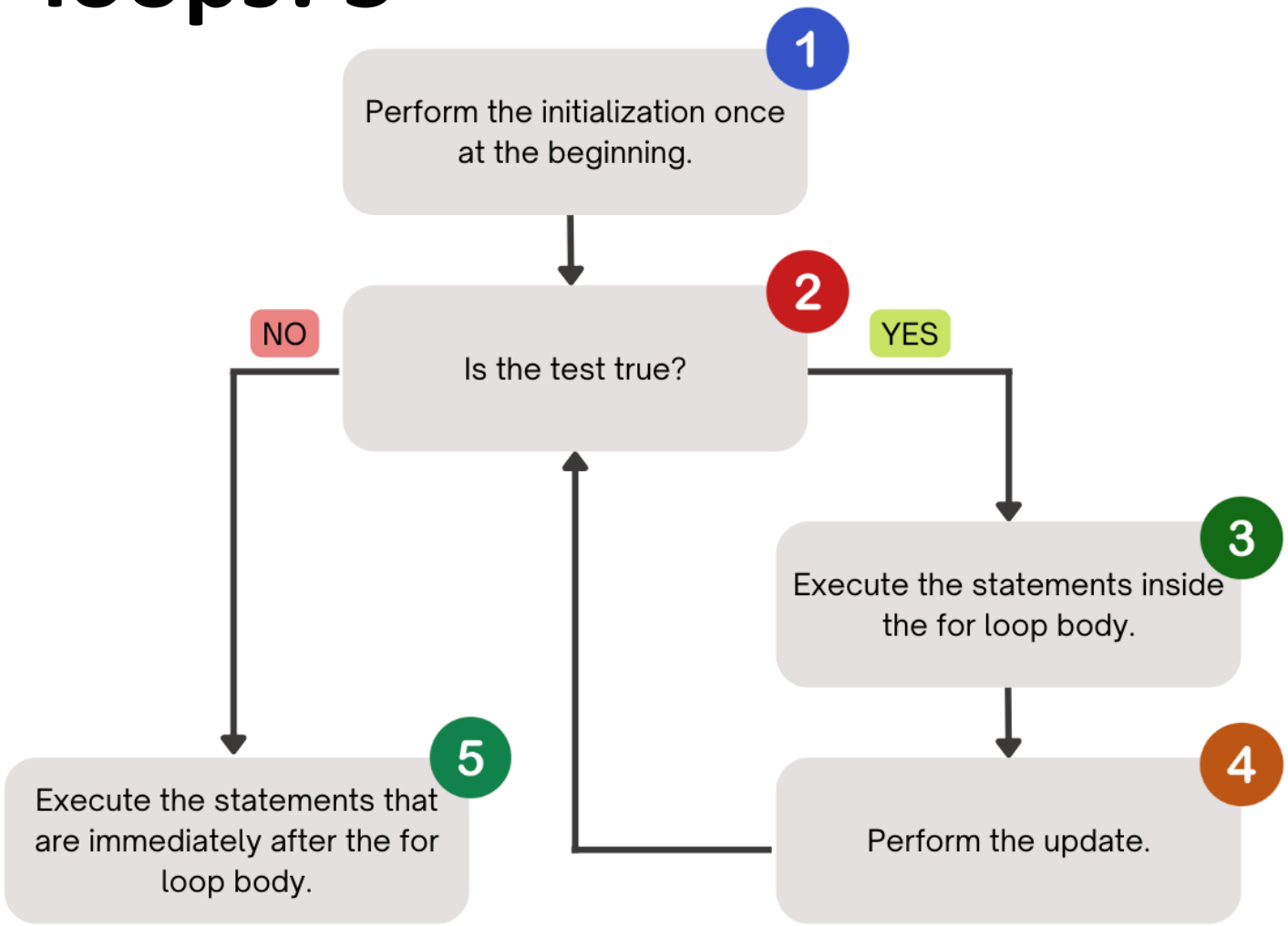
A syntactic structure that *controls* the execution of other statements.

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

(PCM) for loops! 2

```
for (int counter = 1; counter <= 5; counter++) {
    System.out.println("I love CSE 121!");
}
```

(PCM) for loops! 3



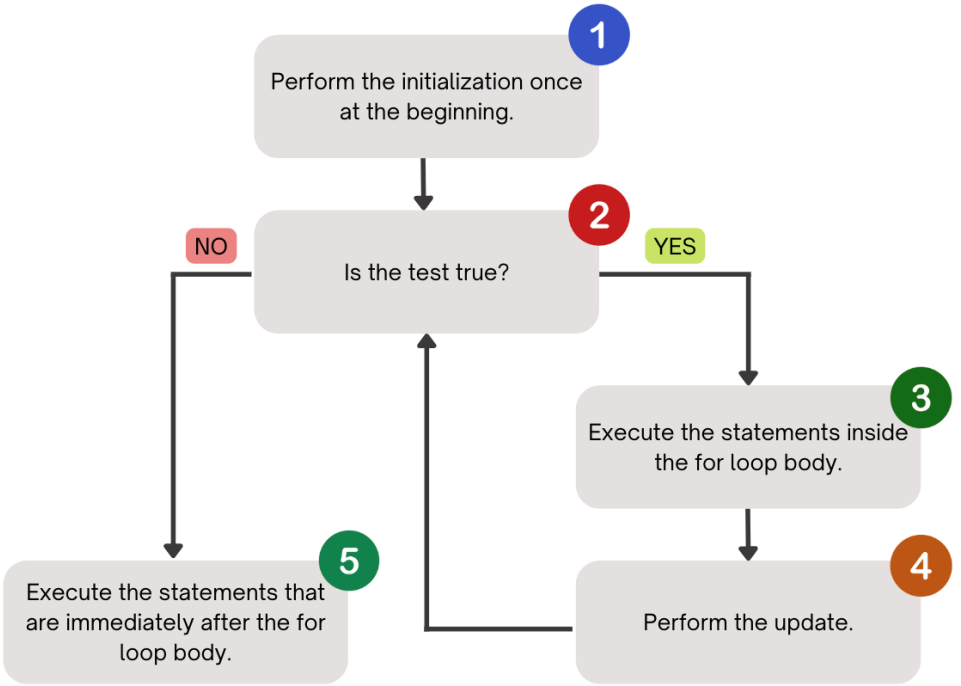
(PCM) for loops! 4

counter

6

```
for (int counter = 1; counter <= 5; counter++) {
    System.out.println("I love CSE 121!");
}
```

```
I love CSE 121!
I love CSE 121!
I love CSE 121!
I love CSE 121!
I love CSE 121!
```



Poll in with your answer!

sli.do #cse121

What output does the following code produce?

```
for (int i = 1; i <= 6; i++) {
  System.out.println(i + " squared = " + i * i);
}
```

A.

```
i squared = i * i
i squared = i * i
i squared = i * i
i squared = i * i
i squared = i * i
i squared = i * i
```

B.

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
5 squared = 25
6 squared = 36
```

C.

```
2 squared = 4
3 squared = 9
4 squared = 16
5 squared = 25
6 squared = 36
7 squared = 49
```

D.

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
5 squared = 25
6 squared = 36
7 squared = 49
```

(PCM) String traversals

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

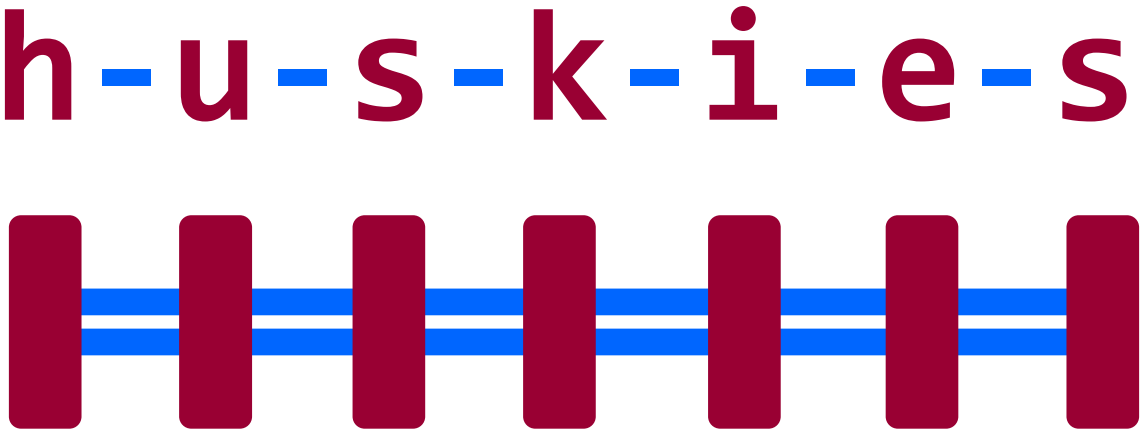
Fencepost Pattern 1

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`

Fencepost Pattern 2

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



Reflection Feedback (summarized)

C0 meant to be an introduction to reflections (graded more leniently, and feedback will be provided)

Reflections will be graded on your ideas (critical thinking), not your writing!

Reflection Megathreads (new!) moving forward

Challenge Question

In your own words, please define what an UStopia is, and what that might look like in a computing context. For instance, what types of computing technologies or practices might be challenged in an UStopia? How might communities use technology to empower humans as opposed to harm them?

Challenge Responses (paraphrased)

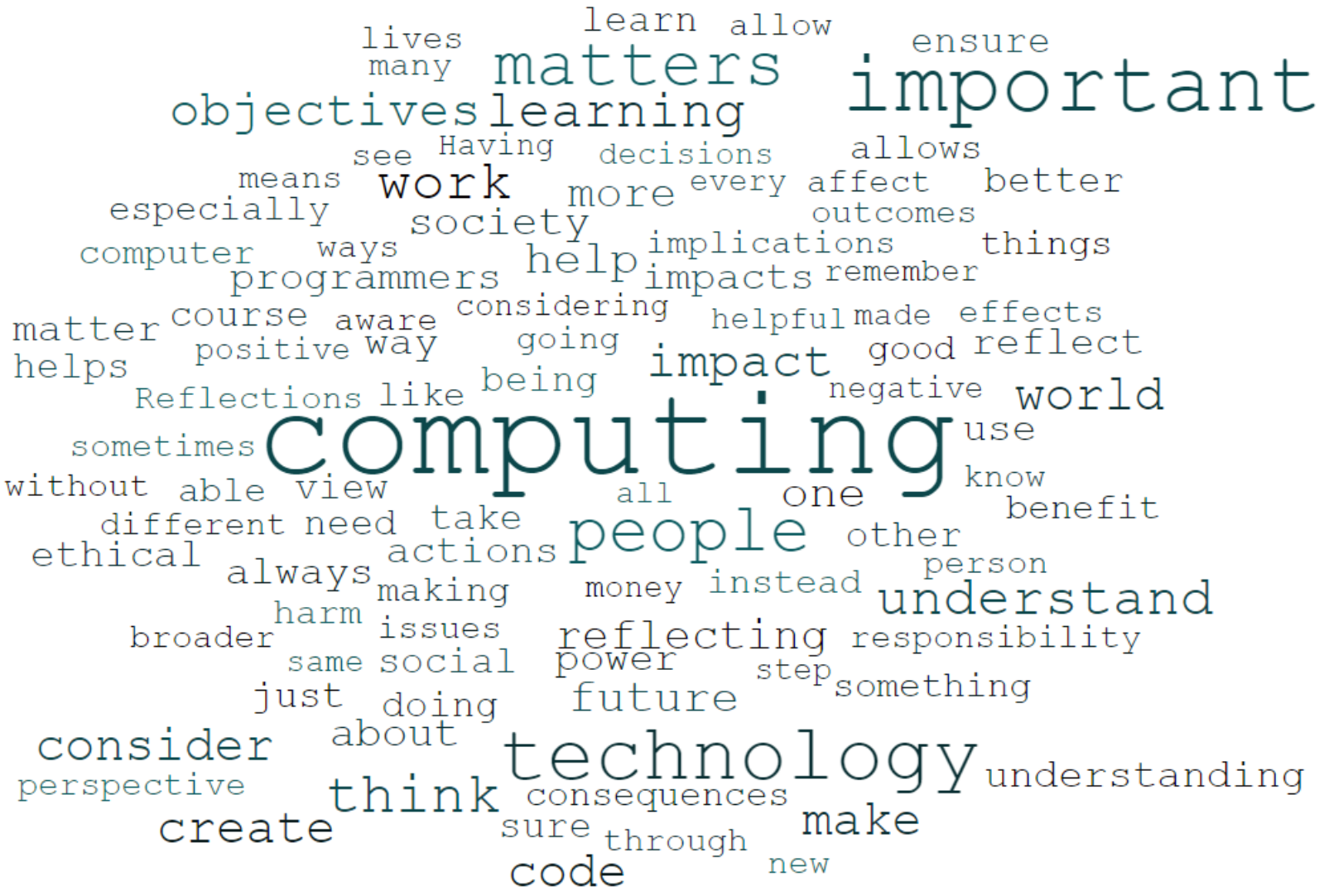
“an UStopia is a society where technology serves humans instead of profit”

“communities use technology to advocate for their needs, such as organizing the distribution of food and clothing”

“practices like artificial intelligence might be challenged”

Reflection Objectives

- Describe the ethical and social impacts of technology and explain how our choices as programmers can influence these impacts
- Challenge dominant assumptions, values, and goals reflected in computing and technology
- Analyze the strengths and limitations of using computing and technology to solve various problems
- Identify interdisciplinary applications of computing that can be in service of different communities
- Understand disparities in access to computing, and explain the consequences of such disparities in technologies we build


learn
allow
lives
many
matters
ensure
important
objectives
learning
see
Having
decisions
allows
means
work
more
every
affect
better
especially
society
outcomes
computer
ways
help
implications
things
programmers
impacts
remember
considering
course
aware
helpful
made
effects
matter
positive
way
going
good
reflect
helps
impact
being
negative
world
Reflections
like
computing
sometimes
use
without
able
view
all
one
know
benefit
different
need
take
people
other
ethical
actions
person
always
making
money
instead
understand
harm
broader
issues
reflecting
responsibility
same
social
power
step
just
doing
future
something
consider
about
technology
understanding
perspective
think
consequences
create
sure
through
make
code
new