LEC 02

CSE 121

String methods, char, More Variables

BEFORE WE START

Talk to your neighbors:

What is your favorite emoji? 😀

Respond on sli.do!

Instructor: Hayden Feeney

TAs: Anisha Savannah William Cayden Tamsyn

Questions during Class?

Raise hand or send here

sli.do #cse121

Agenda (1/5)

- **Announcements, Reminders (now)**
- C0 Reflection Recap
- Datatypes and Expressions Review
- Casting, More Variables and Operators!
- Strings and Characters Review
  - code example!

Announcements, Reminders

- C0 due tonight
  - Expect C0 grades ~ 1 week from submission (we'll announce on Ed)
  - shortly after, your first ***resubmission*** cycle will open!
- P0: Cornbear's Café releases tonight!
  - due Tuesday, July 7th
- No lecture this Friday! (university holiday)

Agenda (2/5)

- Announcements, Reminders
- **Intro Survey, C0 Reflection Recap (now)**
- Datatypes and Expressions Review
- Casting, More Variables and Operators!
- Strings and Characters Review
  - code example!

Intro survey responses

What we're excited about:

- "To learn something completely new to me - I have no programming experience"
- "Chatting with my peers and honestly just figuring out how to code."
- "To learn and build confidence in a new skill I've previously struggled with, hopefully open some creative doors for future projects."

Intro survey responses

What we're nervous about:

- "The fact that I have never taken a computer science course"
- "I'm a bit worried I'll get in over my head and get stuck on the projects and fall behind."
- "theres noone to ask for help"

Resources for help:

- Ed
- IPL
- Instructor office hours
- Email Hayden at hbfeeney@cs.washington.edu

Bugs!

```
~~~~~ grasshopper ~~~~~

 \\       /\
_________/_\__
|.       /   \ |
-------/-----\
/      /       \
```

```
 / / /
<|||||:)
 \ \ \
```

```
 \_\__/0
/ |  | \0
\ |  | /
 /-/--\
```

```
   |___|
 _/     \_
_/       \_
_\       /_
  \_____/
```

```
                    xxx
                   xxxxx
                 xxxm@@mx
                xxx @x@@@x
               xx  @@@x@@x
              xmm  @@@x  xx
              x xxxx @x   x
             xx@@@@@  xx  x
     i       xmm@@xx   x  @
i    i      x@@mmmm    xx@@@
mm   m      x@@@xxx   OO   xx@
 immm m     xxxx   OO OOOOOO @@@
       aaaaa xx  OOOO       OOO@@x
        aa0@axx  0  OOOO   O@    x
          a@@0a@mmmmmmmOOOO@@  xmmmxOOO
      immmma@OOOOOOOa mmmm OOOO@@@OO  OO
    ii  mmmmii@@OOOOOO   xmmOO @@
    i  ii   i   iia0000a   xmmmx
       i    i   i  i  0a
      i     i   i
```

```
  ____\/
 |____|
 ///\\\
```

```
a scorpion!!!

   ___
 //   T     ___
 II  .. ..//  ^
 LL==EEEEK
    (((( \\__V
```

```
          //
____[------]
```

```
_________________//
| o o o o o o o o |
|_________________|
/\/\/\/\/\/\/\/\/\
```

```
      /
      |
      |
      |
  __  |  __
__ \  | / __
  \*****/
__**0v0**__
/ __*****__ \
/           \
```

```
~~~~~~ Beetle ~~~~~~

     _         _
      \       /
       \     /
      (O >< O)
     /~~~~~~~~~\
 \__|====|=====|__/
   <<====0=====>>
   /|====|=====|\
 _/  \________/  \_
```

```
          ^    ^
          0    0
           \  /
      _____\/_____
     |------v--v------|
     |               |
/----|    computah   |------\
     |               |      |
     |               |      |
     |               |      |
     |               |      |
     |_______________|      |
     \-------|o-|-------/
             |  |
            /    \
           /------\
```

```
~~~ Combee (The Bug and Flying Type Pokemon) ~~~

   /---\    0       0    /---\
   \    \   |       |   /    /
    \    \/---\___/---\/    /
     \__/| 0 0 ||| 0 0 |\__/
          \_v_/\_/\_v_/
             | 0 0 |
              \_v_/
```

On Accessibility… (1/2)

Great engagement with the C0 reflection! Some themes:

Suprise!

- “I was really surprised how the speaker was able to understand the speaking tool that he uses to code at such high speeds. I wouldn't have been able to decipher anything that the speaking tool was saying, so it's really impressive that he is able to code efficiently using this tool.”

Programming as problem solving, not just typing

- “…programming is more about problem-solving than about just looking at the screen and understanding it immediately.”

On Accessibility… (2/2)

- Accessibility matters because…
  - "…having accessibility in computer science helps create inclusivity and an equal opportunity to learn and participate in technology"
  - "When we stop perceiving people within disabled communities as incompetent, is when we are able to create means for creating pathways so every person that wants to code is able to do so. "
  - "…fixes for blind users, like better keyboard navigation, end up helping everyone."

Is C0 Accessible?

Probably not…or at least **not *yet***.

When *looking* at ASCII art:

- "…the slashes and symbols aren't screen reader-friendly, as it's just gonna read what's on each line rather than make sense of the whole image."

So, What?

Broadly speaking: the **digital world is inaccessible**

- but, that's changing!
- and **we** have the power to change it!

In CSE 121, we don't have the full knowledge yet to make accessible ASCII art (or Java programs, applications, video games, websites, …)

However, we encourage you to:

- think about accessibility when you make things with computers
- keep on learning more! UW is a **global leader** in digital accessibility
- e.g. at UW: CSE 443E: Accessibility, EDSPE 304, CREATE, HuskyADAPT

Agenda (3/5)

- Announcements, Reminders
- Intro Survey, C0 Reflection Recap
- **Datatypes and Expressions Review (now)**
- Casting, More Variables and Operators
- Strings and Characters Review
  - code example!

Practice: Think

sli.do #cse121

What does this expression evaluate to? Be sure to indicate the type of the resulting value (e.g. 7.0 rather than 7 for a double, String in quotes).

```
2 % 5 + (26 % 6) - 5 / 2 + " corgi " + 1.2 + 1
```

A. `"4 corgi 1.21"`
B. `"2 corgi 2.2"`
C. `"2 corgi 1.21"`
D. `"4 corgi 2.2"`

Practice: Pair

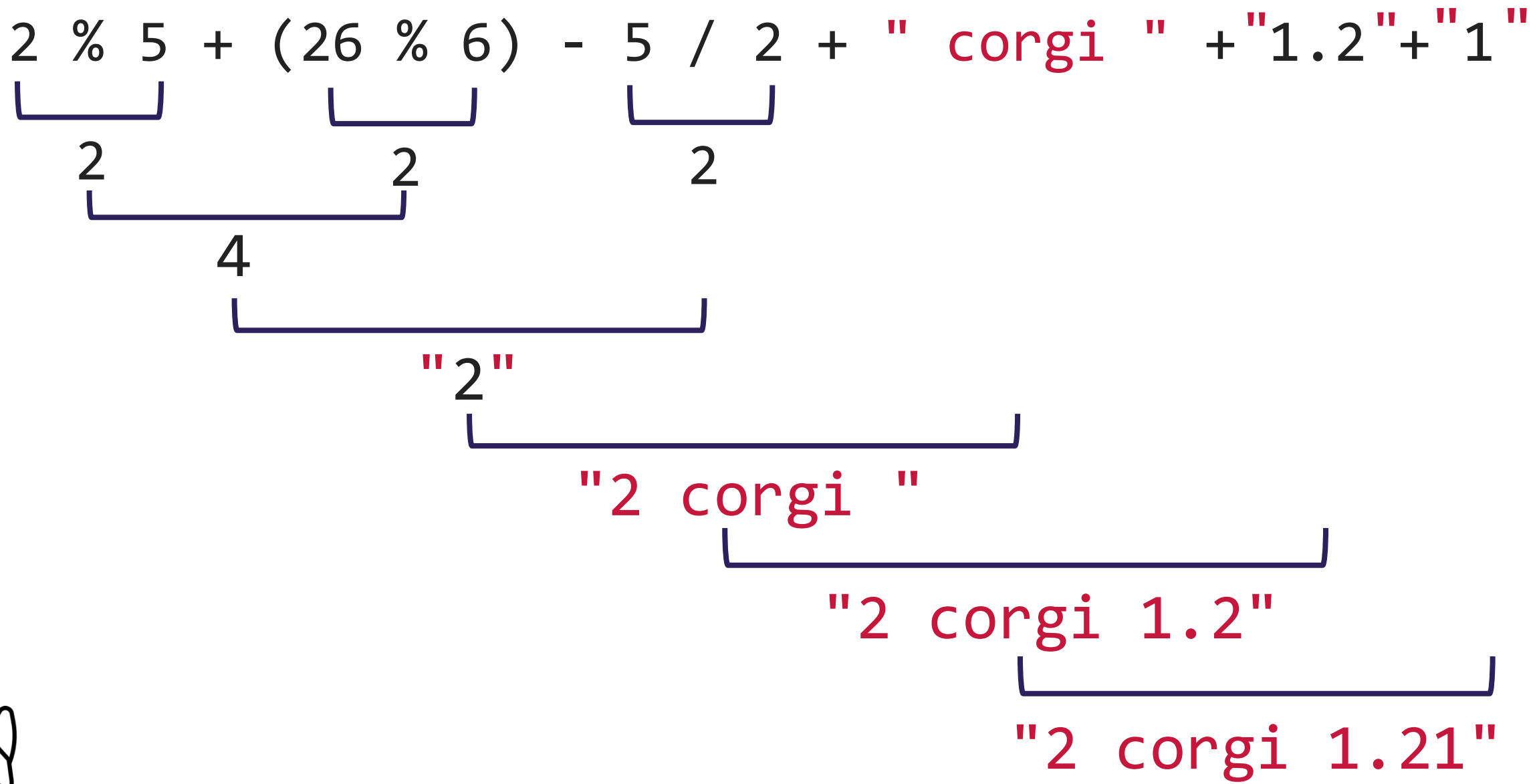
sli.do #cse121

What does this expression evaluate to? Be sure to indicate the type of the resulting value (e.g. 7.0 rather than 7 for a double, String in quotes).

```
2 % 5 + (26 % 6) - 5 / 2 + " corgi " + 1.2 + 1
```

A. `"4 corgi 1.21"`
B. `"2 corgi 2.2"`
C. `"2 corgi 1.21"`
D. `"4 corgi 2.2"`

Worked Out Think-Pair-Share Example



Agenda (4/5)

- Announcements, Reminders
- Intro Survey, C0 Reflection Recap
- Datatypes and Expressions Review
- **Casting, More Variables and Operators! (now)**
- Strings and Characters Review
  - code example!

PCM: Variables

- Recall: Variables allow us to give a name to a specific value
  - 3 parts: declaration, initialization, usage
  - Example:

```
String bestBoy = "gumball";
System.out.println(bestBoy);
```

- Declaration: `int x;`
- Initialization: `x = 30;`
- Or all in one line: `int x = 30;`

PCM: Casting

- Java will do some type conversions for us
  - E.g., `int` to `double`, `double` to `String`, `int` to `String`

- ***BUT*** some conversions Java won't do for us…
  - Nonsensical conversions (e.g., `"Gumball"` to `int`)
  - Conversions that are **"lossy"** (e.g., `double` to `int`)
    - We can ask Java to **typecast** for us

```
double x = 8.83;
int xInt = (int) x;
```

New: Manipulating Variables

They're made to be manipulated, modified, and re-used!

```
int myFavoriteNumber = 7;
int tripleFavNum = myFavoriteNumber * 3;
myFavoriteNumber = myFavoriteNumber + 3;
```

Note! This doesn't really make any mathematical sense. That's because in Java, = is *assignment*, not equality!

New Operator: +=

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

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

```
myFavoriteNumber += 3;
```

This works for both numeric addition and string concatenation!

More Shorthand Operators

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

```
myFavoriteNumber /= 3;
```

Should this work for integers? Doubles? Strings?

Even Shorter Shorthands

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

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

Should this work for integers? Doubles? Strings?

Practice: Think

sli.do #cse121

What values do a, b, and c hold after this code is executed?

```
int a = 10;
int b = 30;
int c = a + b;
c -= 10;
a = b + 5;
b /= 2;
```

A. 10, 30, 40

B. 35, 15, 30

C. 35, 15.5, 30

D. 20, 15, 30

Practice: Pair

sli.do #cse121

What values do a, b, and c hold after this code is executed?

```
int a = 10;
int b = 30;
int c = a + b;
c -= 10;
a = b + 5;
b /= 2;
```

A. 10, 30, 40

B. 35, 15, 30

C. 35, 15.5, 30

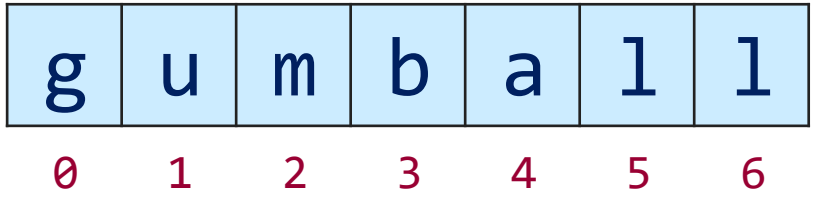
D. 20, 15, 30

Agenda (5/5)

- Announcements, Reminders
- Intro Survey, C0 Reflection Recap
- Typecasting
- Casting, More Variables and Operators!
- **Strings and Characters Review (now)**
  - code example!

PCM: Strings & chars

- Recall: String literals are a sequence of characters that are *strung* together, begin and end with ""
  - Use zero-based indexing
- A `char` represents a single character
  - Begin and end with single quotes (')
  - Strings are made up of chars!




```
char letter = 'g';
char anotherLetter = 'b';
```

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