LEC 02

CSE 121

String Methods, char, More Variables

Questions during Class?

Raise hand or send here

sli.do #cse121

BEFORE WE START

Talk to your neighbors:

What is your favorite emoji? 😛

Music: ♣ CSE 121 25su Lecture Tunes ♣

Instructor:	Hannah Swoffer	
TAs:	Abby	Merav
	Hannah	Trey
	Julia	

Agenda

- **Announcements, Reminders** ⬅
- C0 reflection recap
- Datatypes Review
- 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)
- Programming Assignment 0 releases later today
  - making a receipt generator!
- Resub 0 also releases later today
  - either turn in C0 for the first time or resubmit C0 after receiving feedback for it
- Both due Tuesday, July 7th at 11:59 PM
- Now on regular "cadence" (Wed release, due following Tue)

Reminder: Resubmissions (or "resubs")

- Each week, you may resubmit one Programming Assignment or Creative Project with **no penalty**. The grade of your resubmission will completely replace your previous grade.

- This is a huge opportunity: you get to resubmit your work after we grade it and give you feedback! Please take advantage of this :)

- If you miss an assignment and/or only finish it late – use a resub!

- The time in between receiving assignment feedback and each resub cycle due date is short since summer quarter is short, but I wanted to make sure you still got 7 resub cycles like non-summer quarters.

Resub Logistics

Some logistics:

- There are 8 total resub cycles this quarter (and 8 assignments)
- Assignments eligible to resubmit for 3 cycles max <u>after</u> feedback is out

To resubmit:

1. Make and <u>submit</u> your changes
2. Set the submission you want graded as “Final”
3. **Submit a Google Form**, with a reflection, to confirm your resub
   - You **<u>must</u>** submit the form before the deadline for resub to count

Announcements, Reminders

- IPL is open! Schedule & instructions on website.
- If you joined late, welcome!
  - Check out the course website and lecture recordings

Agenda

- Announcements, Reminders
- **C0 reflection recap** ⬅
- Datatypes Review
- More Variables and Operators!
- Strings and Characters Review
- Code example!

On accessibility…

Loved your reflection responses! Some themes:

- not knowing how low vision people use computers (or program)
- being inspired by the speaker’s perseverance and determination
- accessibility really matters, because:
  - it is important for all people to be able to pursue their passion
  - everyone should have the opportunity to interact with computer science since it impacts many aspects of our world
  - every individual brings new perspectives to the world of code

Is C0 accessible?

Broad spectrum of answers, but **most of you said no.**

When *looking* at ASCII art:

- screenreaders *alone* aren't suited for reading ASCII art
- the caption is probably not enough context for a blind user
- the caption could be low-quality or wrong!

Doing the assignment would be even harder!

- have to learn coding *alongside* new interaction techniques
- Ed doesn't have the same accessibility features

So, what?

Broadly speaking: the digital world is inaccessible (but that's changing)!

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 493E: Accessibility

Agenda

- Announcements, Reminders
- C0 reflection recap
- **Datatypes Review** ←
- 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

$$\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{2 \;\%\; 5}_{2} + \underbrace{(26 \;\%\; 6)}_{2}}_{4} - \underbrace{5 \;/\; 2}_{2}}_{2} + \texttt{" corgi "}}_{\texttt{"2 corgi "}} + 1.2}_{\texttt{"2 corgi 1.2"}} + 1}_{\texttt{"2 corgi 1.21"}}$$

Agenda

- Announcements, Reminders
- C0 reflection recap
- Datatypes Review
- **More Variables and Operators!** ⬅
- Strings and Characters Review
- Code example!

PCM: Typecasting

- 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 xAsAnInt = (int) x;
```

PCM: Variables

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

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

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

New: Manipulating Variables

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

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

Notice – 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 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 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;
```

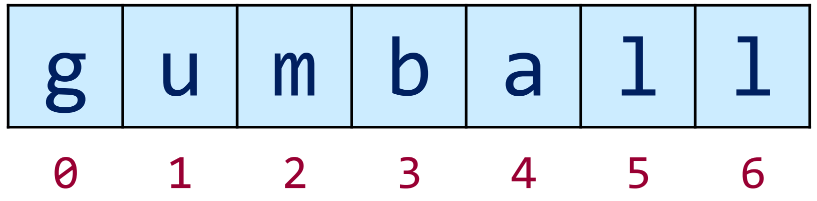
A. 10, 30, 40
B. 35, 15, 30
C. 35, 15.5, 30
D. 20, 15, 30

Agenda

- Announcements, Reminders
- C0 reflection recap
- Datatypes Review
- More Variables and Operators!
- **Strings and Characters Review**
- 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 `char`s!



```
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

Practice: Think

sli.do #cse121

Suppose s contains the String "bubble gum".

Which statement would result in s containing "Gumball" instead?

b	u	b	b	l	e		g	u	m
0	1	2	3	4	5	6	7	8	9

A. `s.substring(7) + "ball";`

B. `s = s.substring(7, 9) + "ball";`

C. `s = s.charAt(7).toUpperCase() + "ball";`

D. `s = s.substring(7, 8).toUpperCase() + s.substring(8) + "ball";`

Practice: Pair

sli.do #cse121

Suppose `s` contains the String `"bubble gum"`.

Which statement would result in `s` containing `"Gumball"` instead?

b	u	b	b	l	e		g	u	m
0	1	2	3	4	5	6	7	8	9

A. `s.substring(7) + "ball";`

B. `s = s.substring(7, 9) + "ball";`

C.
```
s = s.charAt(7).toUpperCase() +
"ball";
```

D.
```
s = s.substring(7, 8).toUpperCase()
    + s.substring(8) + "ball";
```

Aside: Gumball

Agenda

- Announcements, Reminders
- C0 reflection recap
- Datatypes Review
- More Variables and Operators!
- Strings and Characters Review
- **Code example!** ⬅