LEC 03

CSE 121

Strings, char, and Variables

Questions during Class?

Raise hand or send here

sli.do #cse121

BEFORE WE START

Talk to your neighbors:

What is your favorite emoji? 😀

Respond on sli.do!

Music: ❄ CSE 121 26wi Lecture Tunes ❄

Instructor: Miya Natsuhara

TAs:

Amogh	Hayden	Anum	Sam	Shayna
William	Aki	Abdul	Ethan	Jesse
Johnathan	Spencer	Janvi	Jessica	Minh
Anant	Savannah	Navya	Paul	Cayden
Reese	Tamsyn	Ruslana	Carson	

Agenda

- **Announcements, Reminders**
- C0 Reflection Recap
- Casting, More Variables and Operators!
- Strings and Characters Review
  - code example!

Announcements, Reminders

- **P0: Cornbear's Café released!**
  - due Tuesday, January 20th
- Expect C0 grades ~ 1 week from submission (we'll announce on Ed)
  - shortly after, your first ***resubmission*** cycle will open!

Agenda

- Announcements, Reminders
- **C0 Reflection Recap**
- Casting, More Variables and Operators!
- Strings and Characters Review
  - code example!

On Accessibility…

Great engagement with the C0 reflection! Some themes:

- It was inspiring and eye-opening to learn how blind programmers code
  - “I was impressed throughout the video at his pace with the screenreader. When he talked about certain window changes making a world of difference for his productivity, it gave me pause.”
- Little ‘quality of life’ changes can make a big difference!
  - "how a small change in making the debugger’s local window readable by the screen reader allowed her to change how she worked independently. It shows that accessibility isn’t about limiting capability, but also the fact that you can remove unnecessary barriers."
- Accessibility matters because…
  - “technology shapes who is able to learn, work, and [participate] in creating digital tools. Technology should be served to everyone.”

Is C0 Accessible?

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

When *looking* at ASCII art:

- "Hearing the characters read line by line would not communicate the intended image or creativity. This project helped me realize that just because something is “text” does not automatically make it accessible."
- "No, because in my experience, this assignment involved lots of trial and error with writing a line, seeing how it looked after the code was run, and seeing that the art was off"
- "Even though we included a caption at the end to make it more accessible, it does not provide enough information."

Some ideas for improvement:

- Incorporating auditory elements
- Making a tactile version of the ASCII art

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 493E: Accessibility, CREATE, AccessComputing

Agenda

- Announcements, Reminders
- C0 Reflection Recap
- **Casting, More Variables and Operators!**
- 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

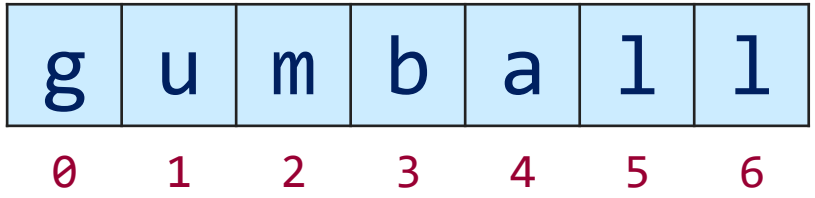
D. 20, 15, 30

Agenda

- Announcements, Reminders
- C0 Reflection Recap
- Typecasting
- Casting, 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 chars!




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

PCM: String Methods

Usage: `<string_variable>.<method>(…)`

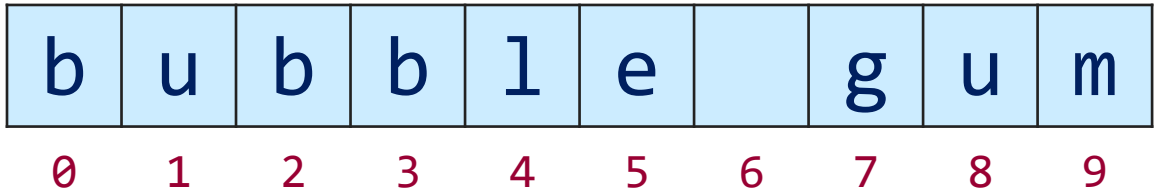
Method	Description
`length()`	Returns the length of the string.
`charAt(i)`	Returns the character at index *i* of the string
`indexOf(s)`	Returns the index of the first occurrence of *s* in the string; returns -1 if *s* doesn't appear in the string
`substring(i, j)` or `substring(i)`	Returns the characters 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

Practice: Think

sli.do #cse121

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

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



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";
```