LEC 03

CSE 121

Characters, Strings, Variables, Debugging

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 25sp Lecture Tunes](#) ✿

Instructor: Miya Natsuhara

TAs:

Chloë	Hibbah	Sushma
Ailsa	Julia	Kelsey
Johnathan	Sahej	Shayna
Christian	Ruslana	Hannah
Merav	Hanna	Zach
Judy	Maitreyi	
Janvi	Ayesha	

Agenda

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

Announcements, Reminders

- P0 was released on Wednesday and is due Tuesday, April 15th
- Expect C0 grades ~ 1 week from submission (we'll announce on Ed)
  - Shortly after, your first resubmission cycle will open!
- Guest lecture on Wednesday: Hannah!

Agenda

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

On accessibility…

Loved your reflection responses! Some themes:

- not knowing how blind people use computers (or program)
- being inspired by the speaker's perseverance and determination
- accessibility really matters, because:
  - "I think that it is incredibly important for people to be able to pursue their passion, and I'm very glad that the speaker was able to become a professional developer."
  - "Accessibility in computer science is important because computer science has become such a large thing that it effects almost everything in the modern world. Therefore it is extremely important that everyone has the opportunity to interact with computer science because computer science directly effects everyone."
  - "This accessibility is important in computer science because 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, CREATE, AccessComputing

Agenda

- Announcements, Reminders
- C0 reflection recap
- **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 xInt = (int) x;
```

PCM: Variables

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

```java
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!

Notice – this doesn't really make any mathematical sense! That's because, in Java, = is *assignment*, not equality!

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

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

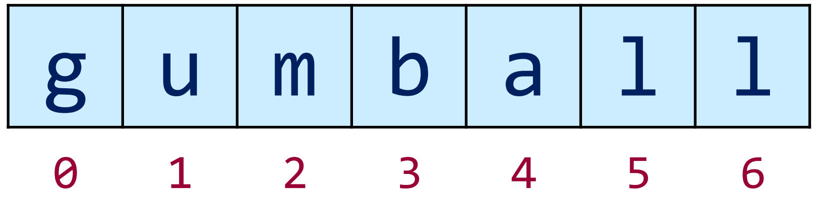
D. 20, 15, 30

Agenda

- Announcements, Reminders
- C0 reflection recap
- 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: 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";
```

Aside: Gumball