CSE 121 – Lesson 3

Miya Natsuhara

Autumn 2023

Music: 121 23au Lecture Tunes 

sli.do #cse121

TAs:	Trey	Christina	Sahej	Vinay	Kriti
	Sebastian	Colton	Anju	Maria	Minh
	Annie	Janvi	Jonus	Shreya	Vivian
	Jasmine	Arkita	Lydia	Andy	Nicole
	Christian	Vidhi	Luke	Nicolas	Simon
	Lucas	Ritesh	Andras	Shayna	Jessie
	Logan	Hibbah	Archit	Hannah	Lydia
	Jacob	Julia	Ayesha	Aishah	Yijia

Announcements, Reminders

- C0 was due on Wednesday
- P0 was released on Wednesday and is due Tuesday, October 10
  - This will be our typical schedule for assignments!
- Quiz 0 scheduled for October 19 (about 2 weeks away)
  - More details about quizzes will be released in the coming week

(PCM) 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;
```

(PCM) Variables

They’re made to be manipulated, modified,

```
int myFavoriteNumber = 7;
int doubleFV = myFavoriteNumber * 2;
myFavoriteNumber = myFavoriteNumber + 3;
```

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

New Operators!

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

This type of pattern is so common, we have an even *shorter* way we can write it!

```
myFavoriteNumber += 3;
```

You can do the same for `-=`, `*=`, `/=`, and `%=`

And there are even shorter versions for *incrementing* and *decrementing*! `myFavoriteNumber++;` `myFavoriteNumber--;`

Poll in with your answer!

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

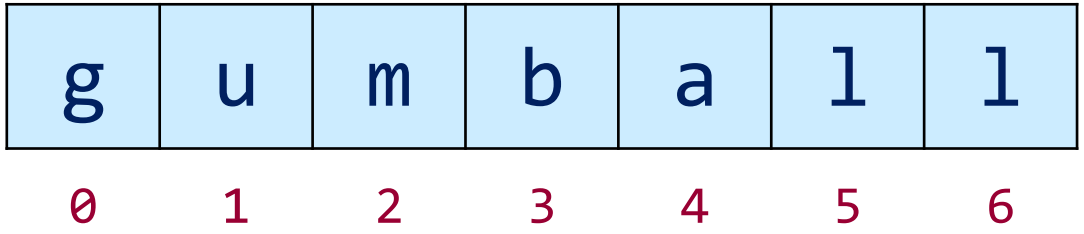
B. 35, 15, 30

C. 35, 15.5, 30

D. 20, 15, 30

(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!

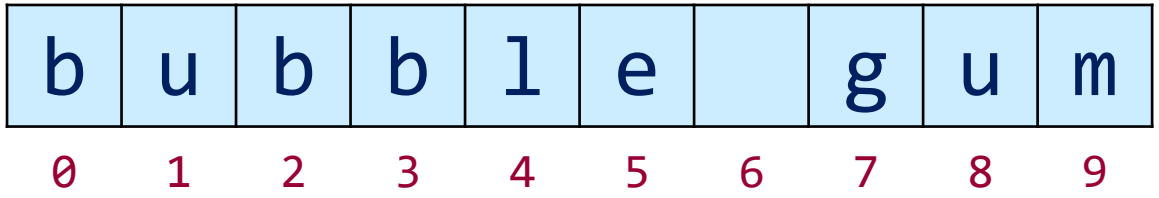
(PCM) String Methods

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

Poll in with your answer!

Suppose s contains the String "bubble gum". Which option below 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";
```

E.
```
s =
    s.substring(7, 8).toUpperCase()
  + s.substring(7, 10) + "ball";
```