

LEC 08

CSE 122

Nested Collections

Questions during Class?
Raise hand or send here

sli.do #cse122



BEFORE WE START

Talk to your neighbors:

What book are you reading right now?

Respond on sli.do!

Instructor: Brett Wortzman

TAs:	Archit	Daniel	Naomi	Sreshta
	Blake P.	David	Neha	Suyash
	Blake R.	Hanna	Nicole	Trisha
	Cady	Isis	Rio	Wesley
	Carson	Jun	Rohan	Yash
	Cole	Landon	Shreyank	
	Connor	Michelle	Somya	

Music: 🎸 [CSE 122 26au Lecture Tunes](#) 🎵

Agenda

- **Announcements** 
- Review: Nested Collections
- Practice: mostFrequentStart
- Practice: Social Network

Announcements

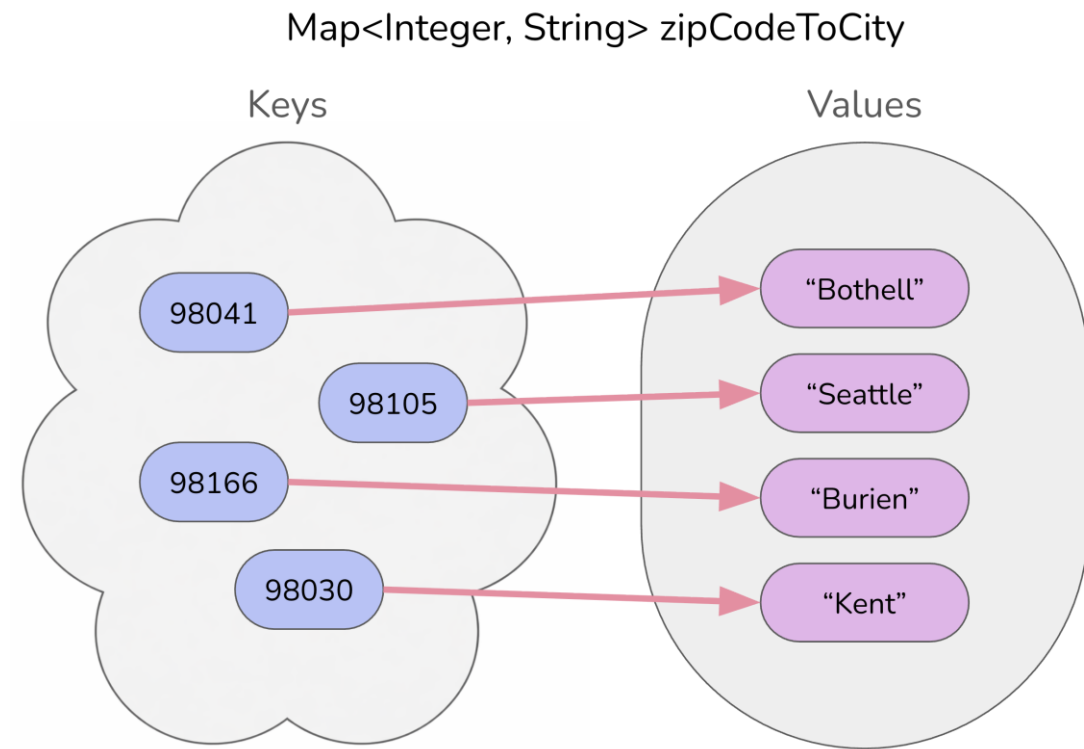
- P1 due tomorrow (Oct. 29)
 - Code quality and reflection not graded on project, but *will* impact computer quiz
 - You should take these seriously now
- CQ1 scheduling open; begins Friday, Oct. 30
 - *Sign up early— slots fill!*
 - *Same-day changes **not allowed**; email Brett if an emergency*
- Working on grades for CQ0 and PQ0
 - *Hoping* to publish CQ0 grades tomorrow, but can't commit yet
- Working on unified gradebook; will announce once ready

Agenda

- Announcements
- **Review: Nested Collections** ◀
- Practice: mostFrequentStart
- Practice: Social Network

Reminder: Map ADT

- Data structure to map keys to values
 - Keys can be any* type; Keys must be unique
 - Values can be any type
- Example: Mapping ticker to stock price in P0
- Operations
 - `put(key, value)`: Associate key to value
 - Overwrites existing keys
 - `get(key)`: Get value for key
 - `remove(key)`: Remove key/value pair

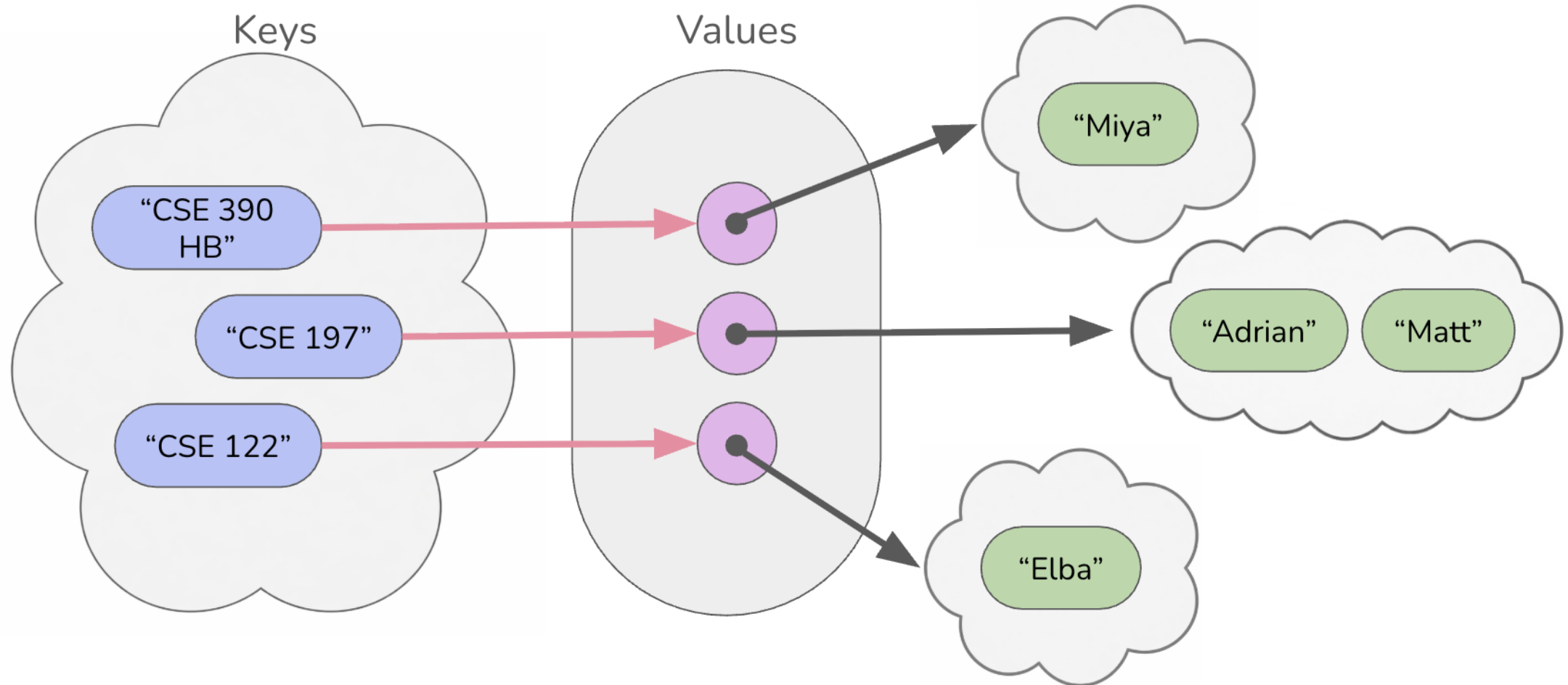


(PCM) Nested Collections

- The values inside a collection can be any type, including other collections
- Examples:
 - Queue: Students in the IPL by group
 - `Queue<Set<String>>`
 - Map: Section → **Set of students** in that section
 - `Map<String, Set<String>`
 - Map: Recipe → **List of ingredients** in that recipe
 - `Map<String, List<String>`
 - Or even `Map<String, Map<String, Double>>` to track amounts!

(PCM) Nested Collections

Map<String, Set<String>> courses

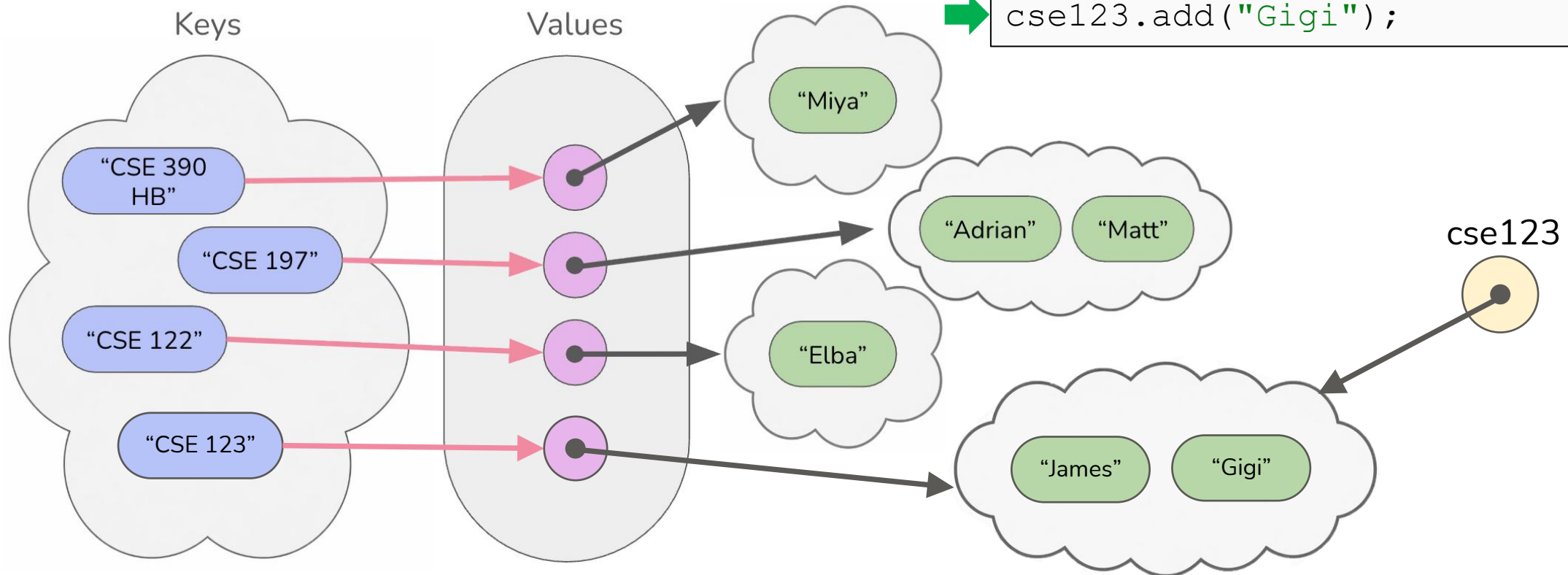


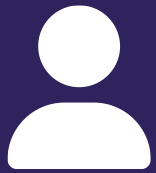
(PCM) Updating Nested Collections

The “value” inside the Map is a reference to the data structure!

- Think carefully about number of references to a particular object

```
courses.put("CSE 123",  
            new HashSet<String>());  
courses.get("CSE 123").add("James");  
Set<String> cse123 =  
    courses.get("CSE 123");  
cse123.add("Gigi");
```





Practice : Think

sli.do

#cse122

What would be stored in map after executing the following code?

```
map: { "KeyA"=[1, 2], "KeyB"=[3], "KeyC"=[4, 5, 6] }
```

```
Set<Integer> nums = map.get("KeyA");  
nums.add(7);  
map.put("KeyB", nums);  
map.get("KeyA").add(8);  
map.get("KeyB").add(9);
```

- A. { "KeyA"=[1, 2], "KeyB"=[1, 2, 7], "KeyC"=[4, 5, 6] }
- B. { "KeyA"=[1, 2, 8], "KeyB"=[1, 2, 7, 9], "KeyC"=[4, 5, 6] }
- C. { "KeyA"=[1, 2, 7, 8], "KeyB"=[1, 2, 7, 9], "KeyC"=[4, 5, 6] }
- D. { "KeyA"=[1, 2, 7, 8, 9], "KeyB"=[1, 2, 7, 8, 9], "KeyC"=[4, 5, 6] }



Practice : Pair

sli.do [#cse122](https://twitter.com/cse122)

What would be stored in map after executing the following code?

```
map: { "KeyA"=[1, 2], "KeyB"=[3], "KeyC"=[4, 5, 6] }
```

```
Set<Integer> nums = map.get("KeyA");  
nums.add(7);  
map.put("KeyB", nums);  
map.get("KeyA").add(8);  
map.get("KeyB").add(9);
```

- A. { "KeyA"=[1, 2], "KeyB"=[1, 2, 7], "KeyC"=[4, 5, 6] }
- B. { "KeyA"=[1, 2, 8], "KeyB"=[1, 2, 7, 9], "KeyC"=[4, 5, 6] }
- C. { "KeyA"=[1, 2, 7, 8], "KeyB"=[1, 2, 7, 9], "KeyC"=[4, 5, 6] }
- D. { "KeyA"=[1, 2, 7, 8, 9], "KeyB"=[1, 2, 7, 8, 9], "KeyC"=[4, 5, 6] }



Practice : Pair

[sli.do](#) [#cse122](#)

What would be stored in map after executing the following code?

```
map: { "KeyA"=[1, 2], "KeyB"=[3], "KeyC"=[4, 5, 6] }
```



```
Set<Integer> nums = map.get("KeyA");  
nums.add(7);  
map.put("KeyB", nums);  
map.get("KeyA").add(8);  
map.get("KeyB").add(9);
```

nums

KeyA: [1, 2, 7, 8, 9]

KeyB: [3]

KeyC: [4, 5, 6]

- A. { "KeyA"=[1, 2], "KeyB"=[1, 2, 7], "KeyC"=[4, 5, 6] }
- B. { "KeyA"=[1, 2, 8], "KeyB"=[1, 2, 7, 9], "KeyC"=[4, 5, 6] }
- C. { "KeyA"=[1, 2, 7, 8], "KeyB"=[1, 2, 7, 9], "KeyC"=[4, 5, 6] }
- D. { "KeyA"=[1, 2, 7, 8, 9], "KeyB"=[1, 2, 7, 8, 9], "KeyC"=[4, 5, 6] }

Agenda

- Announcements
- Review: Nested Collections
- **Practice: mostFrequentStart** 
- Practice: Social Network

mostFrequentStart

Write a method called `mostFrequentStart` that takes a Set of words and does the following steps:

- Organizes words into “word families” based on which letter they start with
- Selects the largest “word family” as defined as the family with the most words in it
- Updates the Set of words to only have words from the selected family
- Returns the starting letter of the largest word family

mostFrequentStart

For example, if the Set words stored the values:

```
["hello", "goodbye", "library", "literary", "little", "repel"]
```

The word families produced would be:

```
'h' -> 1 word ("hello")
```

```
'g' -> 1 word ("goodbye")
```

```
'l' -> 3 words ("library", "literary", "little")
```

```
'r' -> 1 word ("repel")
```

Since 'l' has the largest word family, we **modify the Set** to only contain Strings starting with 'l' and finally **return 'l'**.

Agenda

- Announcements
- Review: Nested Collections
- Practice: mostFrequentStart
- **Practice: Social Network** ◀

Social Network

- Write methods to complete our implementation of a simple social network program. You should complete the following methods:
 - addUser: adds a new user to the network with no friends (🤔)
 - addFriend: adds a friend to a user (friendships are *not* symmetric)
 - removeUser: removes a user from the network and all friends lists
 - getAllFriendsOf: returns a Set of all the given user's friends *and* all users who are friends with the given user