LEC 10

CSE 122

Nested Collections

Questions during Class?

Raise hand or send here

sli.do #cse122

Slido vote & chat with neighbors:

What are you doing for Halloween?

Music: 122 25au Lecture Tunes

Instructor: Elba Garza

TAs: Sreshta, William, Arjun, Dani, Rohan, Andrew, Saachi, Ava, Merav, Nicole, Vrinda, Shreya, Wesley, Isis, Colin, Medha, Shivani, Naomi, Hanna, David, Sushma, Rio, Nicolae, Ivory, Yang, Cady, Diya, Katharine

Agenda

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

Announcements

- Programming Assignment 2 (P2) was released on Friday!
  - Seriously, start early! This assignment is much more involved…
  - Due Nov 4th by 11:59 PM
- Quiz 1 on Nov 4th in your registered Quiz Section
  - Topics: (Reference Semantics), Stacks and Queues, Sets, Maps
  - Practice Quiz 1 available Friday; solutions on Sunday
- Quiz 0 grades to be released this weekend!
- Tomorrow, Resubmission Cycle 3 (R3) form out, due Nov 4th by 11:59 PM
  - Available assignments: **P0**, C1, P1
  - Reminder: to use a resubmission cycle you need to
    - (1) submit your work (big blue "Submit" button on Ed)
    - **AND** (2) fill out the resubmission form (linked from Ed + course calendar)

Mid-Quarter Evaluation

Feedback coming soon.
Meeting with ET&L tomorrow!

An aside for quizzes…

Please be legible and clear on your written answers

a. Each option below describes a proposed functional decomposition of this code by listing which lines of code would appear in each **helper** method. Note that once created, helper methods can be called in multiple different places if needed.
Select **one** option below whose described methods would result in the **best** functional decomposition.

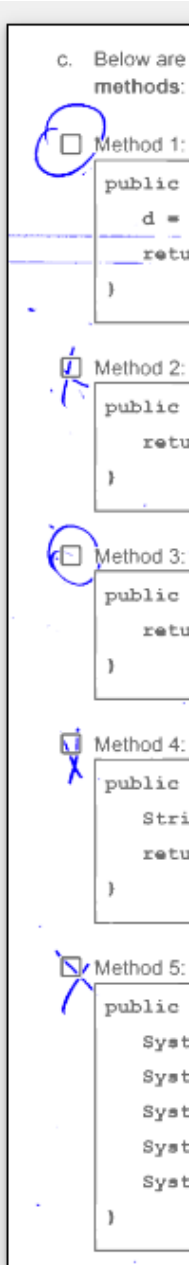
- [ ] Helper method 1: lines 6-8
  Helper method 2: lines 11-13
  Helper method 3: lines 15-17
- [ ] Helper method 1: lines 4-17
- [x] Helper method 1: lines 4-9
  Helper method 2: lines 10-13
  Helper method 3: lines 15-18
- [ ] Helper method 1: line 1
  Helper method 2: lines 3-9
  Helper method 3: lines 14-17
- [ ] Helper method 1: lines 5-8
  Helper method 2: lines 10-13

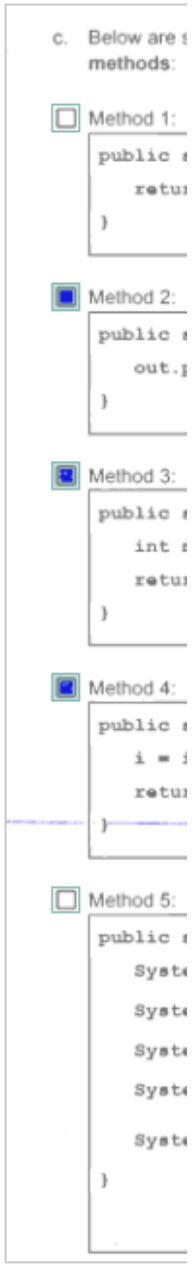
b. Select **each** option below that is a true statement.

- [x] Functional decomposition involves breaking down a complex process or task into smaller, discrete steps.
- [ ] "Good" functional decomposition should always reduce the number of lines of a program.
- [x] Functional decomposition makes it easier for outside programmers to revise and understand the code.
- [ ] All tasks, regardless of complexity, should be functionally decomposed into their own methods.

An aside for quizzes…

Please be legible and clear on your written answers 🥴





An aside for quizzes…

Please write your name legibly!

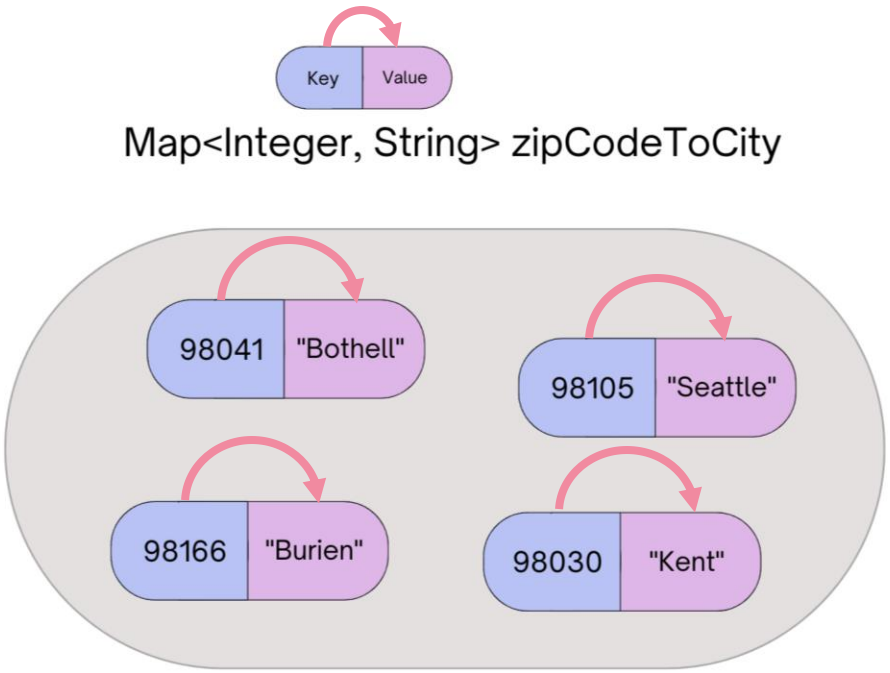
Name of Student: CSE 122

Agenda

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

Map ADT

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



Same as Python's `dict`

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 <u>largest</u> "word family" as defined as the family with the most words in it
- Returns the starting letter of the largest word family (and *should update the Set of words to only have words from the selected family*).

mostFrequentStart

For example, if the `Set` words stored the values:

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

The word families produced would be:

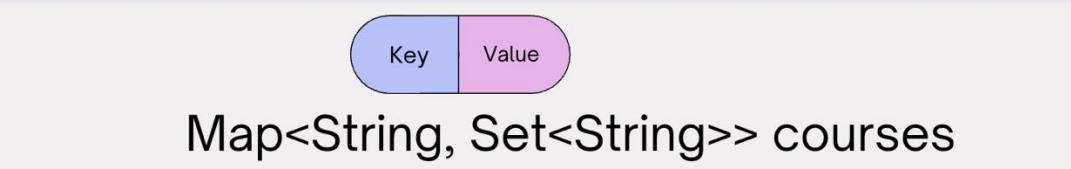
```
'h' -> 1 word ("hello")
'g' -> 1 word ("goodbye")
'l' -> 3 words ("library", "literary", "little")
'r' -> 1 word ("repel")
```

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

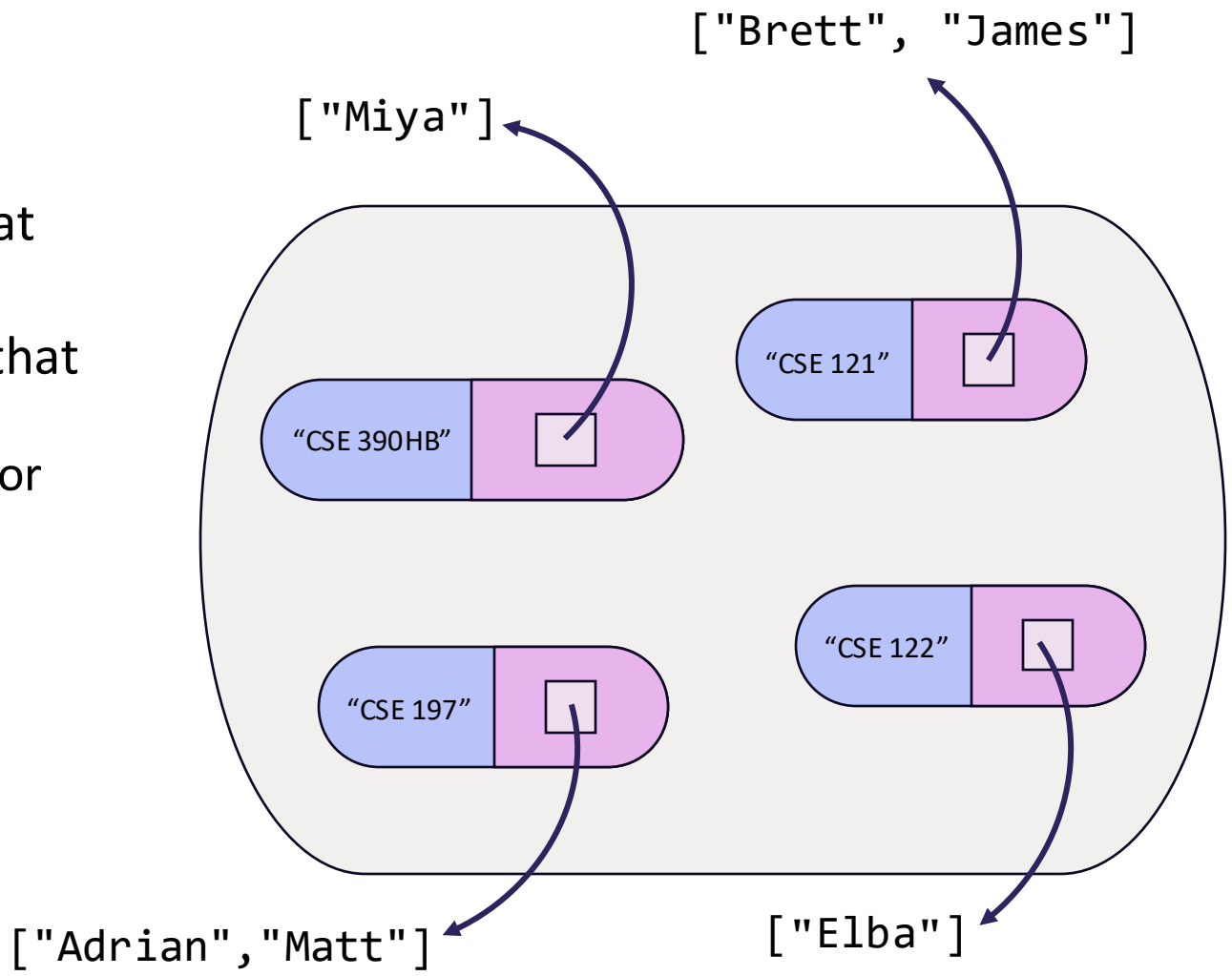
Agenda

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

Nested Collections



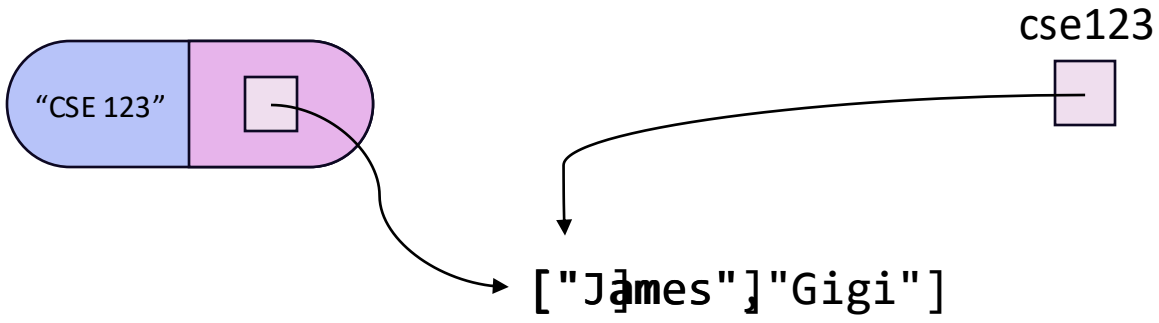
- The values inside a Map can be any type, including <u>data structures</u>
- Common examples:
  - Mapping: Section ➔ **Set of students** in that section
  - Mapping: Recipe ➔ **Set of ingredients** in that recipe
    (Or even Map<String, Map<String, Double>> for units!)



Updating Nested Collections

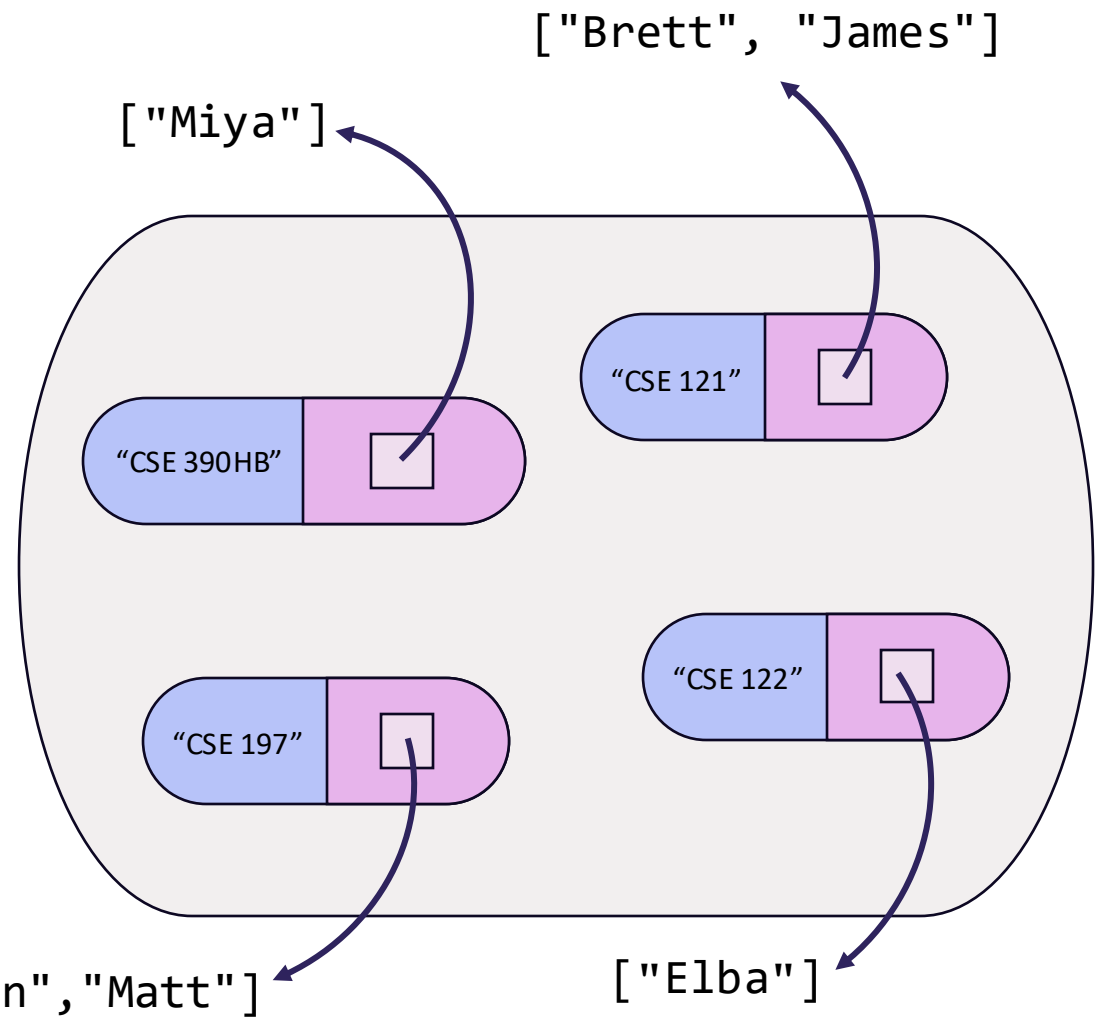
The "value" inside the Map is a <u>reference</u> to the data structure!

- 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

Suppose map had the following items. What would its items be after running this 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

Suppose map had the following items. What would its items be after running this 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: mostFrequentStart
- Recap: Nested Collections
- **Practice: Social Network**