LEC 08

CSE 122

Maps

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

What is your least favorite root vegetable?

Music: Hunter/Miya's Playlist

Instructor	Hunter Schafer / Miya Natsuhara
TAs	Ajay, Andrew, Anson, Anthony, Audrey, Chloe, Colton, Connor, Elizabeth, Evelyn, Gaurav, Hilal, Hitesh, Jake, Jin, Joe, Joe, Karen, Kyler, Leon, Melissa, Noa, Parker, Poojitha, Samuel, Sara, Simon, Sravani, Tan, Vivek

(Review) Choosing a Data Structure: Tradeoffs

- You got a bit of practice with this in your quiz sections on Tuesday!
  - Solving the same problem with an `ArrayList`, a `Stack`, and a Queue

- Things to consider:
  - Functionality
    - If you need duplicates or indexing, `Sets` are not for you!
  - Efficiency
    - Different data structures are “good at” different things!

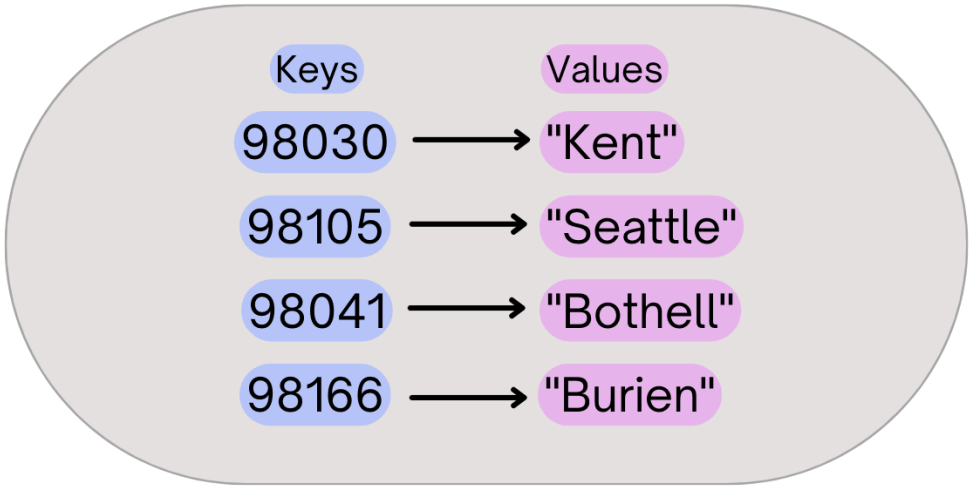
Lecture Outline

- Map Review
- Debrief PCM: Count Words
- Practice: Combine Rosters
- Practice: mostFrequentStart

(PCM) Map ADT

- Data structure to map keys to values
  - Keys can be any* type; Keys are unique
  - Values can be any type
- Example: Mapping nucleotides to counts!
- Operations
  - `put(key, value)`: Associate key to va
    - Overwrites duplicate keys
  - `get(key)`: Get value for key
  - `remove(key)`: Remove key/value pair

Map<Integer, String> zipCodeToCity



Same as Python's dict

(PCM) Programming with Maps

- Interface: `Map`
- Implementations: `TreeMap`, `HashMap`

```
// Making a Map
Map<String, String> favArtistToSong = new TreeMap<>();

// adding elements to the above Map
favArtistToSong.put("Steve Lacy", "Dark Red");
favArtistToSong.put("The Cranberries", "Linger");
favArtistToSong.put("Umi", "Bet");

// Getting a value for a key
String song = favArtistToSong.get("Umi");
System.out.println(song);
```

(PCM) Programming with Maps

Methods	Description
`put(`**key**`, `**value**`)`	adds a mapping from the given key to the given value; if the key already exists, replaces its value with the given one
`get(`**key**`)`	returns the value mapped to the given key (`null` if not found)
`containsKey(`**key**`)`	returns `true` if the map contains a mapping for the given key
`remove(`**key**`)`	removes any existing mapping for the given key
`clear()`	removes all key/value pairs from the map
`size()`	returns the number of key/value pairs in the map
`isEmpty()`	returns `true` if the map's size is 0
`toString()`	returns a string such as `"{a=90, d=60, c=70}"`
`keySet()`	returns a set of all keys in the map
`values()`	returns a collection of all values in the map

(PCM) Map Implementations

- Our first data structures with marked differences in how their implementations behave
- One `Map` ADT / Interface
- Two `Map` implementations
  - `TreeMap` – Pretty fast, sorted keys
  - `HashMap` – Extremely fast, unsorted keys

```
Map<String, Integer> map1 = new TreeMap<>();
Map<String, Integer> map2 = new HashMap<>();
...
```