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  Gaurav
Andrew  Hilal
Anson  Hitesh
Anthony  Jake
Audrey  Jin
Chloe  Joe
Colton  Joe
Connor  Karen
Elizabeth  Kyler
Evelyn  Leon

Questions during Class?
Raise hand or send here

sli.do  #cse122
(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
**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 value
    - Overwrites duplicate keys
  - `get(key)`: Get value for key
  - `remove(key)`: Remove key/value pair

Same as Python’s `dict`
Programming with Maps

• Interface: Map
• Implementations: TreeMap, HashMap

```java
// 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);
```
# Programming with Maps

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

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