CS 106A, Lecture 20

HashMaps

suggested reading:

Java Ch. 13.2
Learning Goals

• Know how to store data in and retrieve data from a **HashMap**.
Plan for today

• Recap: ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Plan for today

• Recap: ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Our First ArrayList

// Create an (initially empty) list
ArrayList<String> list = new ArrayList<>();

// Add an element to the back
list.add("Hello");  // now size 1

list.add("there!");  // now size 2
## ArrayList Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>list.add(value);</code></td>
<td>appends value at end of list</td>
</tr>
<tr>
<td><code>list.add(index, value);</code></td>
<td>inserts given value just before the given index, shifting subsequent values to the right</td>
</tr>
<tr>
<td><code>list.clear();</code></td>
<td>removes all elements of the list</td>
</tr>
<tr>
<td><code>list.get(index)</code></td>
<td>returns the value at given index</td>
</tr>
<tr>
<td><code>list.indexOf(value)</code></td>
<td>returns first index where given value is found in list (-1 if not found)</td>
</tr>
<tr>
<td><code>list.isEmpty()</code></td>
<td>returns true if the list contains no elements</td>
</tr>
<tr>
<td><code>list.remove(index);</code></td>
<td>removes/returns value at given index, shifting subsequent values to the left</td>
</tr>
<tr>
<td><code>list.remove(value);</code></td>
<td>removes the first occurrence of the value, if any</td>
</tr>
<tr>
<td><code>list.set(index, value);</code></td>
<td>replaces value at given index with given value</td>
</tr>
<tr>
<td><code>list.size()</code></td>
<td>returns the number of elements in the list</td>
</tr>
<tr>
<td><code>list.toString()</code></td>
<td>returns a string representation of the list such as &quot;[3, 42, -7, 15]&quot;</td>
</tr>
</tbody>
</table>
Insert/remove

- If you insert/remove in the front or middle of a list, elements **shift** to fit.

  ```java
  list.add(2, 42);
  ```

  - shift elements right to make room for the new element

  ![Shift elements right](image)

  ```java
  list.remove(1);
  ```

  - shift elements left to cover the space left by the removed element

  ![Shift elements left](image)
# ArrayLists + Primitives = 💔

<table>
<thead>
<tr>
<th>Primitive</th>
<th>“Wrapper” Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
</tr>
</tbody>
</table>
// Use wrapper classes when making an ArrayList
ArrayList<Integer> list = new ArrayList<>();

// Java converts Integer <-> int automatically!
int num = 123;
list.add(num);

int first = list.get(0);  // 123

Conversion happens automatically!
Example: Opening Crawl

Karel Wars Episode VII:
Revenge of the SuperKarel

An evil SuperKarel is threatening to disturb the grid world once again. What was once a peaceful planet has now been thrown into chaos.

By taking the turnRight() command hostage, SuperKarel has forced all other Karels to turnLeft() three times.

The Karels' only hope to avoid code bloat is to re-implement turnRight()
Example: Planner

• Let’s write a program that emulates the Star Wars “opening crawl”
  – The program first reads in a text file
  – It then animates this text flowing upwards

An evil SuperKarel is threatening to disturb the grid world once again.
What was once a peaceful planet has now been thrown into chaos.

By taking the turnRight() command hostage, SuperKarel has forced all other Karels to turnLeft() three times.

The Karels’ only hope to avoid code bloat is to re-implement turnRight()
• Recap: ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Limitations of Lists

• Can only look up by *index* (int), not by String, etc.
• Cumbersome for preventing duplicate information
• Slow for lookup

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>12</td>
<td>49</td>
<td>-2</td>
<td>26</td>
<td>5</td>
<td>17</td>
<td>-6</td>
<td>84</td>
<td>72</td>
<td>3</td>
</tr>
</tbody>
</table>
How Is Webpage Lookup So Fast?

Google search bar with the query "what does the fox say" and results showing "About 18,300,000 results (0.57 seconds)". 
Introducing... HashMaps!

• A variable type that represents a collection of **key-value pairs**
• You access values by **key**
• Keys and values can be any type of **object**
• Resizable – can add and remove pairs
• Has helpful methods for searching for keys
HashMap Examples

• **Phone book:** name -> phone number
• **Search engine:** URL -> webpage
• **Dictionary:** word -> definition
• **Bank:** account # -> balance
• **Social Network:** name -> profile
• **Counter:** text -> # occurrences
• And many more...
import java.util.*;

HashMap<String, String> myHashMap = new HashMap<>();
Our First HashMap

```
HashMap<String, String> myHashMap = new HashMap<>();
```
Our First HashMap

HashMap\(<\text{String}, \text{String}>\) myHashMap = new HashMap<>();
Our First HashMap

HashMap<String, String> myHashMap = new HashMap<>();

Type of values your HashMap will store.
Our First HashMap

```java
HashMap<String, String> myHashMap = new HashMap<>();
```
Our First HashMap

```java
HashMap<String, String> myHashMap = new HashMap<>();
```
Our First HashMap

HashMap<String, String> myHashMap = new HashMap<>();
// Create an (initially empty) HashMap
HashMap<String, String> map = new HashMap<>();
map.put("dog", "bark"); // Add a key-value pair
map.put("cat", "meow"); // Add another pair
map.put("seal", "ow ow"); // Add another pair
map.put("seal", "ow ow ow"); // Overwrites!
Our First HashMap - Get

String s = map.get("dog"); // Get a value for a key
String s = map.get("cat"); // Get a value for a key
String s = map.get("fox"); // null
map.remove("dog"); // Remove pair from map
map.remove("seal"); // Remove pair from map
map.remove("fox"); // Does nothing if not in map
Review: HashMap Operations

- `m.put(key, value);` Adds a key/value pair to the map.
  ```java
  m.put("Eric", "650-123-4567");
  ```
  - Replaces any previous value for that key.

- `m.get(key)` Returns the value paired with the given key.
  ```java
  String phoneNum = m.get("Jenny"); // "867-5309"
  ```
  - Returns null if the key is not found.

- `m.remove(key);` Removes the given key and its paired value.
  ```java
  m.remove("Rishi");
  ```
  - Has no effect if the key is not in the map.

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Jenny&quot;</td>
<td>&quot;867-5309&quot;</td>
</tr>
<tr>
<td>&quot;Mehran&quot;</td>
<td>&quot;123-4567&quot;</td>
</tr>
<tr>
<td>&quot;Marty&quot;</td>
<td>&quot;685-2181&quot;</td>
</tr>
<tr>
<td>&quot;Chris&quot;</td>
<td>&quot;947-2176&quot;</td>
</tr>
</tbody>
</table>
Using HashMaps

• A HashMap allows you to get from one half of a pair to the other.
  – Remembers one piece of information about every key.

    // key      value
    m.put("Jenny", "867-5309");

  – Later, we can supply only the key and get back the related value:

    Allows us to ask: What is Jenny’s phone number?

    m.get("Jenny")
    "867-5309"
Q: What are the correct map contents after the following code?

HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");

A. {C=Lee, J=Cain, M=Stepp, M=Sahami}
B. {C=Lee, J=Cain, M=Stepp}
C. {J=Cain M=Sahami, M=Stepp}
D. {J=Cain, K=Schwarz, M=Sahami}
E. other
Q: What are the correct map contents after the following code?

```java
HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");
```

**Values:**
- "Schwarz"
- "Sahami"
- "Lee"

**Keys:**
- "K"
- "M"
- "C"
Q: What are the correct map contents after the following code?

```java
HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");
```

Keys: "K" "M" "C"
Values: "Schwarz" "Stepp" "Lee"
Q: What are the correct map contents after the following code?

```java
HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");
```

Keys: 
- "K"
- "M"
- "C"

Values: 
- "Schwarz"
- "Stepp"
- "Lee"
Q: What are the correct map contents after the following code?

```java
HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");
```

Keys:
- "M"
- "C"

Values:
- "Stepp"
- "Lee"
Q: What are the correct map contents after the following code?

```java
HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");
```
Q: What are the correct map contents after the following code?

```java
HashMap<String, String> map = new HashMap<>();
map.put("K", "Schwarz");
map.put("C", "Lee");
map.put("M", "Sahami");
map.put("M", "Stepp");
map.remove("Stepp");
map.remove("K");
map.put("J", "Cain");
map.remove("C, Lee");
```

Values: "Cain", "Stepp", "Lee"

Keys: "J", "M", "C"
Plan for today

• Recap: ArrayLists
• HashMaps

• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Exercise: Dictionary

• Write a program to read a dictionary of words and definitions from a file, then prompt the user for words to look up.

  – Example data from the dictionary input file:

    | abate                      |
    | to lessen; to subside     |
    | pernicious                |
    | harmful, injurious        |

• How can a **HashMap** help us solve this problem?
Plan for today

• Recap: ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Iterating Over HashMaps

... for (String key : map.keySet()) {
    String value = map.get(key);
    // do something with key/value pair...
}
// Keys occur in an unpredictable order!

Values:
- “bark”
- “ow ow ow”
- “meow”

Keys:
- “dog”
- “seal”
- “cat”
Counting Exercise

• Write a program to count the number of occurrences of each unique word in a large text file (e.g. *Moby Dick*).
  – Allow the user to type a word and report how many times that word appeared in the book.
  – Report all words that appeared in the book at least 500 times.

• How can a **map** help us solve this problem?
  – Think about scanning over a file containing this input data:

```
To be or not to be or to be a bee not two bees ...
^```

Maps and Tallying

• a map can be thought of as generalization of a tallying array
  – the "index" (key) doesn't have to be an int
  – count digits: 22092310907

// (R)epublican, (D)emocrat, (I)ndependent
– count votes: "RDDDDDDRRRRRDDDDDDDRDRRIRDRRIRDRRID"

key | "R" | "D" | "I"
---|-----|-----|-----
value | 16   | 14   | 3

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;R&quot;</td>
<td>→ 16</td>
</tr>
<tr>
<td>&quot;D&quot;</td>
<td>→ 14</td>
</tr>
<tr>
<td>&quot;I&quot;</td>
<td>→ 3</td>
</tr>
</tbody>
</table>
Plan for today

- Recap: ArrayLists
- HashMaps
- Practice: Dictionary
- HashMaps as Counters
- Practice: What’s Trending
- Recap
• Social media can be used to monitor popular conversation topics.

• Write a program to count the frequency of #hashtags in tweets:
  – Read saved tweets from a large text file.
  – Report hashtags that occur at least 15 times.

• How can a **map** help us solve this problem?

  Given these hashtags...  
  
  | #stanford   |
  | #summer     |
  | #california |
  | #stanford   |

  We want to store...

  | "#stanford" → 2 |
  | "#summer" → 1  |
  | "#california" → 1 |
Recap

• Recap: ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending

Next time: defining our own variable types
Overflow (extra) slides
Anagram exercise

• Write a program to find all anagrams of a word the user types.

  Type a word [Enter to quit]: scared
  Anagrams of scared:
  cadres cedars sacred scared

• How can a map help us solve this problem?
**Anagram observation**

- Every word has a *sorted form* where its letters are arranged into alphabetical order.
  
  "fare" → "aefr"
  "fear" → "aefr"
  "swell" → "ellsw"
  "wells" → "ellsw"

- Notice that anagrams have the same sorted form as each other.
  - How is this helpful for solving the problem?
  - Suppose we were given a **sortLetters** method. How to use it?
public String sortLetters(String s) { ... }  // assume this exists ...

// build map of {sorted form => all words with that sorted form}
HashMap<String, String> anagrams = new
    HashMap<String, String>();
try {
    Scanner input = new Scanner(new File("dictionary.txt"));
    while (true) {
        String word = input.next();
        String sorted = sortLetters(word);  // "acders"
        if (anagrams.containsKey(sorted)) {
            String rest = anagrams.get(sorted);
            anagrams.put(sorted, rest + " " + word);  // append
        } else {
            anagrams.put(sorted, word);  // new k/v pair
        }
    }  // {"acders" => "cadres caders sacred scared", ...}
} catch (FileNotFoundException fnfe) {
    println("Error reading file: " + fnfe);
}
// prompt user for words and look up anagrams in map
String word = readLine("Type a word [Enter to quit]: ");
while (word.length() > 0) {
    String sorted = sortLetters(word.toLowerCase());
    if (anagrams.containsKey(sorted)) {
        println("Anagrams of " + word + ":");
        println(anagrams.get(sorted));
    } else {
        println("No anagrams for " + word + ".");
    }
    word = readLine("Type a word [Enter to quit]: ");
}