CS 106A, Lecture 20
ArrayLists and HashMaps

suggested reading:
Java Ch. 13.2
Learning Goals

• Know how to store data in and retrieve data from a HashMap.

```
Tweets file: EllenTweets.txt
#tbt: 42
#findingdory: 20
#laughdancepartner: 55
#laughdancepartnerâ¦19
#edbypetsmart: 21
#littlebigshots: 18
#thebachelor: 16
#oscars: 19
#firstdates: 33
```
Plan for today

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Plan for today

• ArrayLists Recap and Reversible Writing
  • ArrayList Methods and Planner
  • Arrays vs. 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

// Access elements by index (starting at 0!)
println(list.get(0)); // prints "Hello"
println(list.get(1)); // prints "there!"
Example: Reversible Writing

Let’s write a program that reverses a text file.

I am not a person who contributes
And I refuse to believe that
I will be useful

I will be useful
And I refuse to believe that
I am not a person who contributes
String filename = promptUserForFile("Filename: ", "res");
try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
String filename = promptUserForFile("Filename: ", "res");
try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
Example: Reversible Writing

String filename = promptUserForFile("Filename: ", "res");

try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
String filename = promptUserForFile("Filename: ", "res");
try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
String filename = promptUserForFile("Filename: ", "res");

try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
String filename = promptUserForFile("Filename: ", "res");
try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
Example: Reversible Writing

String filename = promptUserForFile("Filename: ", "res");

try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
String filename = promptUserForFile("Filename: ", "res");
try {
    Scanner s = new Scanner(new File(filename));
    ArrayList<String> lines = new ArrayList<>();

    // Read all lines and store in our ArrayList
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }

    // Output the lines from back to front
    for (int i = lines.size() - 1; i >= 0; i--) {
        println(lines.get(i));
    }
} catch (IOException ex) {
    println("Could not read file.");
}
Plan for today

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
## 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>
• 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

```
index  0  1  2  3  4  5
value  3  8  9  7  5
```

```
index  0  1  2  3  4  5
value  3  8  42 9  7  5
```

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

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

```
index  0  1  2  3  4  5
value  3 8  42 9  7  5
```

```
index  0  1  2  3  4  5
value  3 42 9  7  5
```
Example: Planner

• Let’s write a program to help plan out our day
  – The program first prompts for things you want to do today
  – Then, it asks the user to re-input them in order of completion
  – Finally, it outputs the order the user has chosen for their tasks
Planner: Approach

Todos:

“Do crossword”
**Planner: Approach**

<table>
<thead>
<tr>
<th><strong>Todos:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Do crossword”</td>
<td>“Sleep”</td>
</tr>
</tbody>
</table>
## Planner: Approach

<table>
<thead>
<tr>
<th>Todos:</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Do crossword”</td>
</tr>
<tr>
<td>“Sleep”</td>
</tr>
<tr>
<td>“Talk to Annie”</td>
</tr>
</tbody>
</table>
Planner: Approach

Todos:

- “Do crossword”
- “Sleep”
- “Talk to Annie”

Order:

- “Do crossword”
Planner: Approach

Todos:

```
| "Sleep" | "Talk to Annie" |
```

Order:

```
"Do crossword"
```
Planner: Approach

Todos:  
“Sleep”

Order:  
“Do crossword”  “Talk to Annie”
## Planner: Approach

### Todos:
DONE!

### Order:
- “Do crossword”
- “Talk to Annie”
- “Sleep”
To the code!
Plan for today

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
ArrayLists + Primitives = 💔

// Doesn't compile 😞
ArrayList<int> list = new ArrayList<>();

Unlike arrays, ArrayLists can only store Objects!
ArrayLists + Primitives =

<table>
<thead>
<tr>
<th>Primitive</th>
<th>&quot;Wrapper&quot; 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!
## Array vs. ArrayList

<table>
<thead>
<tr>
<th>ArrayList</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArrayList&lt;Integer&gt; list = new ArrayList&lt;&gt;();</code></td>
<td><code>int[] arr = new int[2]; // [0, 0]</code></td>
</tr>
<tr>
<td>list.add(1); // [1]</td>
<td>arr[0] = 1; // [1, 0]</td>
</tr>
<tr>
<td>list.add(2); // [1, 2]</td>
<td>arr[1] = 2; // [1, 2]</td>
</tr>
<tr>
<td>list.set(0, 3); // [3, 2]</td>
<td>arr[0] = 3; // [3, 2]</td>
</tr>
<tr>
<td>int x = list.get(0); // 3</td>
<td>int x = arr[0]; // 3</td>
</tr>
<tr>
<td>list.add(4); // [3, 2, 4]</td>
<td>[no equivalent]</td>
</tr>
<tr>
<td>list.contains(2); // true</td>
<td></td>
</tr>
</tbody>
</table>
Array vs. ArrayList

Why do both of these exist in the language?
- Arrays are Java's fundamental data storage
- ArrayList is a library built on top of an array

When would you choose an array over an ArrayList?
- When you need a fixed size that you know ahead of time
  - Simpler syntax for getting/setting
  - More efficient
- Multi-dimensional arrays (e.g., images)
- Histograms/tallying
Plan for today

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. 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 results for "what does the fox say" showing about 18,300,000 results in 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

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

Type of keys your HashMap will store.
Our First HashMap

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

Type of values your HashMap will store.
Our First HashMap

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

Values:
- "bark"
- "meow"

Keys:
- "dog"
- "seal"
- "cat"
... 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
Our First HashMap - Remove

... 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("Annie");
  ```
  
  • Has no effect if the key is not in the map.
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.
    
    ```java
    // key value
    m.put("Jenny", "867-5309");
    
    //     key         value
    m.get("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?
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");
```

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");
```

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;K&quot;</td>
<td>&quot;Schwarz&quot;</td>
</tr>
<tr>
<td>&quot;M&quot;</td>
<td>&quot;Sahami&quot;</td>
</tr>
<tr>
<td>&quot;C&quot;</td>
<td>&quot;Lee&quot;</td>
</tr>
</tbody>
</table>
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:  

Values:

```
"Schwarz"
"Stepp"
"Lee"
```

```
"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:
- "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");
```

Keys: "J" "M" "C"

Values: "Cain" "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");
```
Plan for today

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. 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

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. ArrayLists
• HashMaps
• Practice: Dictionary
• HashMaps as Counters
• Practice: What’s Trending
• Recap
Iterating Over HashMaps

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

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

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:

```plaintext
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: "RDDDDDDRRRRRRDDDDDDDRDRRIRDRRIRDRRRDRD"

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

<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>

index 0 1 2 3 4 5 6 7 8 9
value 3 1 3 0 0 0 0 1 0 2
Plan for today

• ArrayLists Recap and Reversible Writing
• ArrayList Methods and Planner
• Arrays vs. 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...

We want to store...

```
#stanford → 2
#summer → 1
#california → 1
```
Recap

• ArrayLists are a variable type representing a list of items
• Unlike arrays, ArrayLists have:
  – The ability to resize dynamically
  – Useful methods you can call on them
• Unlike ArrayLists, arrays have:
  – The ability to store any type of item, not just Objects
• HashMaps are a variable type representing a key-value pairs
• Unlike arrays and ArrayLists, HashMaps:
  – Are not ordered
  – Store information associated with a key of any Object type

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]: ");
}