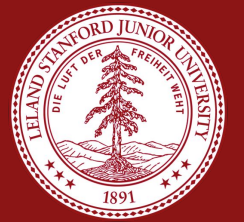
# HashMaps

## Lecture 19

CS106A, Summer 2019
Sarai Gould && Laura Cruz-Albrecht

With inspiration from slides created by Keith Schwarz, Mehran Sahami, Eric Roberts, Stuart Reges, Chris Piech and others.



# Announcements

- Blank code is available to download from the website.

# Announcements

- Midterm grades go out on Gradescope after lecture!
  - You will receive an email in your Stanford email inbox.
  - Regrade Requests are due by next Monday, Aug. 5th at 10am.
  - Sample solutions are online.

- ParaKarel was due at 10am.
- Assignment 5 goes out right after lecture!
  - If you work as a pair, submit as a pair on Paperless!

# Learning Goal for Today

Know how to store data in and retrieve data from a

`HashMap`

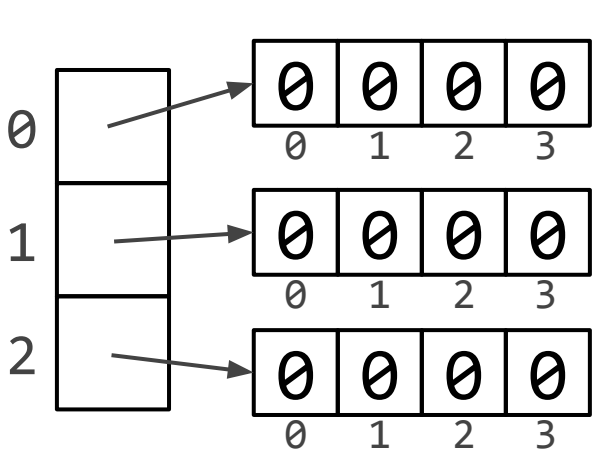
# Plan for Today

- Review: Arrays, 2D Arrays, ArrayLists
- HashMaps and Animal Sounds
- Example: Interesting Dictionary
- Example: CountingWords
- Example: Swapping Keys and Values

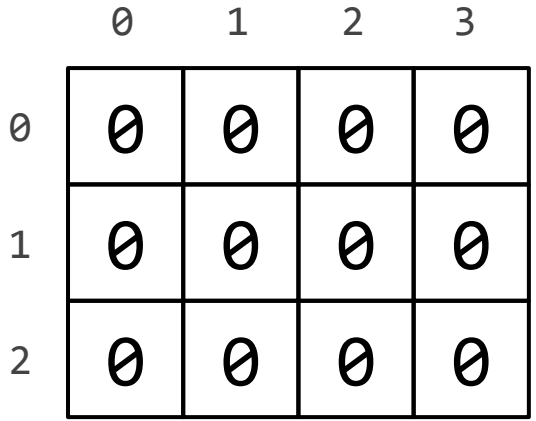
# Review 2D: Arrays

# Review 2D: Arrays

```
int[][] matrix = new int[3][4];
```




An array of arrays



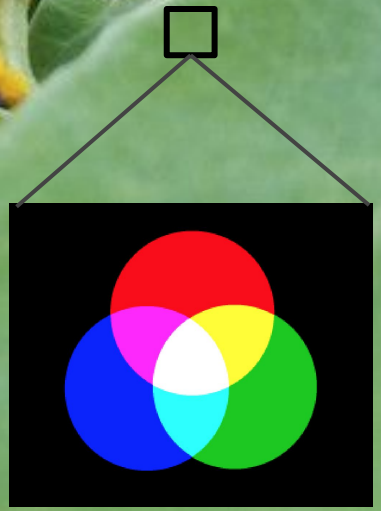

A unit (ie, a grid/matrix)

# Review: 2D Arrays ❤️ For Loops

- The canonical way to loop over a 2D array is with a double `for` loop

```
type[][] arr = ...
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        // do something with arr[row][col] ...
    }
}
```

Review: images are 2D arrays of *pixels*.



# Our First ArrayList

Type of thing your ArrayList will store

Same type here.

```
ArrayList<String> myArrayList = new ArrayList<String>();
```

# Our First ArrayList

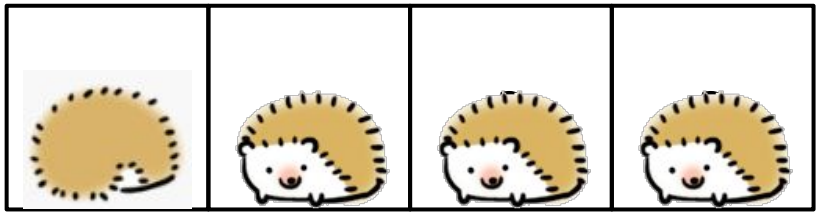
Type of thing your ArrayList will store

Can optionally leave empty because of type inference

```
ArrayList<String> myArrayList = new ArrayList<>();
```

# When you don't know how many  are coming to the party

- An ordered, *resizable* list of information
- Can add and remove elements (among other cool functionality)
- Homogenous
- Can store any **Object** type
- Access individual items by *index*




**hedgehogPartyList**

# Data Structures so Far

What if we want to associate multiple types of data together?

An arrays and ArrayLists can only store one data type at a time. How can we create a relationship between data?

# Data Structures so Far

List: `ArrayList<Type>`

Array: `type[]`

Matrix/2D Array: `type[][]`

**Note**: All of these are Objects and passed by reference.

# HashMaps!

HashMaps have a map of keys -> values.

If you look up a key in a HashMap, it will give you the associated value back!

# Our First HashMap

```
HashMap<String, String> firstMap = new HashMap<String, String>();
```

# Our First HashMap

Data Type of "keys"
in our HashMap

```
HashMap<String, String> firstMap = new HashMap<String, String>();
```

# Our First HashMap

Data Type of "values"
in our HashMap

```
HashMap<String, String> firstMap = new HashMap<String, String>();
```

# Our First HashMap

Repeated types of key->value pairs

```
HashMap<String, String> firstMap = new HashMap<String, String>();
```

# Our First HashMap

Like ArrayLists, only stores objects

```
HashMap<String, Integer> firstMap = new HashMap<String, Integer>();
```

# Our First HashMap

Or, we can leave this blank because of type inference!

```
HashMap<String, String> firstMap = new HashMap<>();
```

# Our First HashMap

```
import java.util.*;


HashMap<String, String> firstMap = new HashMap<String, String>();
```

# Our First HashMap

```
import java.util.*;


HashMap<String, String> firstMap = new HashMap<String, String>();
```

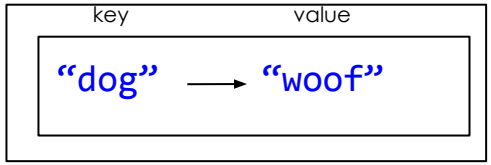
Let's create a HashMap that maps from animal names to animal sounds!

Basically, we're creating a dictionary where, if we look up an animal's name, it will tell us which sound it makes!

# Our First HashMap

```
import java.util.*;

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

sounds.put("dog", "woof");
```

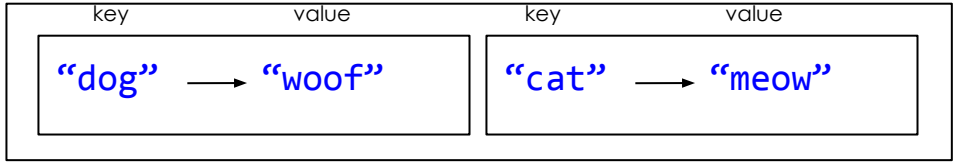
sounds



# Our First HashMap

```
import java.util.*;

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

sounds.put("dog", "woof");

sounds.put("cat", "meow");
```

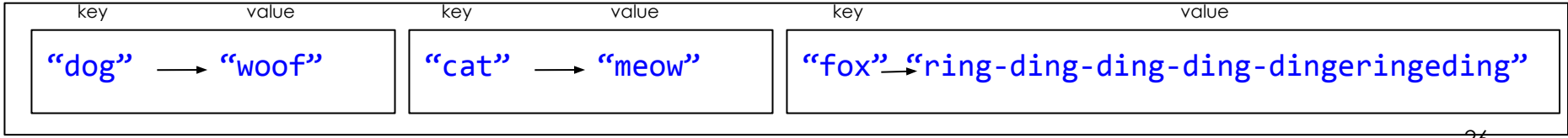
sounds



# Our First HashMap

```
import java.util.*;

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

sounds.put("dog", "woof");

sounds.put("cat", "meow");

// what does the fox say?*
sounds.put("fox", "ring-ding-ding-ding-dingeringeding");
```

sounds




* Music reference… Not just crazy.

# Our First HashMap

```
import java.util.*;
HashMap<String, String> sounds = new HashMap<String, String>();
sounds.put("dog", "woof");
sounds.put("cat", "meow");
// what does the fox say?*
sounds.put("fox", "ring-ding-ding-ding-dingeringeding");
```

sounds

| key | value |
| --- | --- |
| "dog" | "woof" |
| "cat" | "meow" |
| "fox" | "ring-ding-ding-ding-dingeringeding" |

* Music reference… Not just crazy.

# Our First HashMap

```
import java.util.*;
HashMap<String, String> sounds = new HashMap<String, String>();
sounds.put("dog", "woof");
sounds.put("cat", "meow");
// what does the fox say?*
sounds.put("fox", "ring-ding-ding-ding-dingeringeding");

String dogSound = sounds.get("dog");
```

sounds

| key | value | key | value | key | value |
|---|---|---|---|---|---|
| "dog" | "woof" | "cat" | "meow" | "fox" | "ring-ding-ding-ding-dingeringeding" |

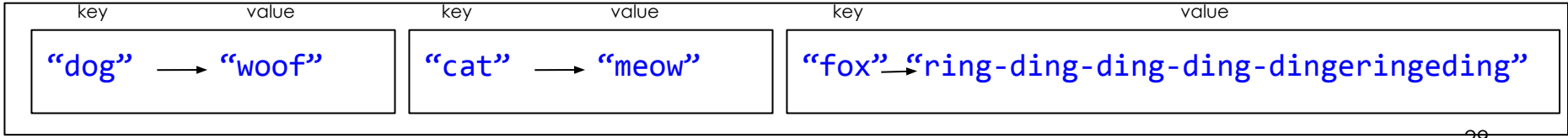
* Music reference… Not just crazy.

# Our First HashMap

```
import java.util.*;
HashMap<String, String> sounds = new HashMap<String, String>();
sounds.put("dog", "woof");
sounds.put("cat", "meow");
// what does the fox say?*
sounds.put("fox", "ring-ding-ding-ding-dingeringeding");

String dogSound = sounds.get("dog");
println(dogSound);
```

```
woof
```

sounds



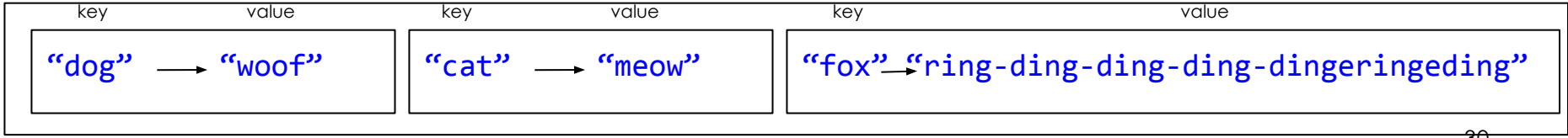

* Music reference… Not just crazy.

# Our First HashMap

```
import java.util.*;
HashMap<String, String> sounds = new HashMap<String, String>();
sounds.put("dog", "woof");
sounds.put("cat", "meow");
// what does the fox say?*
sounds.put("fox", "ring-ding-ding-ding-dingeringeding");

String dogSound = sounds.get("dog");
println(dogSound);
println(sounds.get("cat"));
```

```
woof
meow
```

sounds



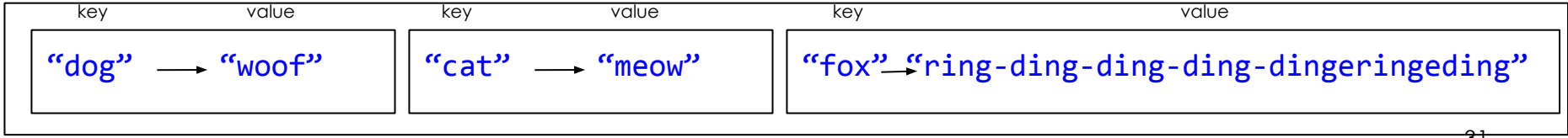

* Music reference… Not just crazy.

# Our First HashMap

```
import java.util.*;
HashMap<String, String> sounds = new HashMap<String, String>();
sounds.put("dog", "woof");
sounds.put("cat", "meow");
// what does the fox say?*
sounds.put("fox", "ring-ding-ding-ding-dingeringeding");

String dogSound = sounds.get("dog");
println(dogSound);
println(sounds.get("cat"));
println(sounds.get("hippopotamus")); // this isn't in our map!
```

```
woof
meow
null
```

sounds

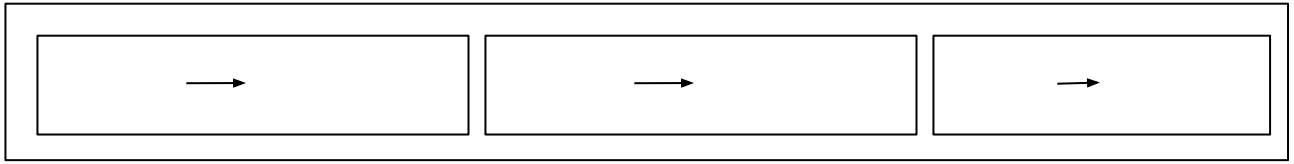



* Music reference… Not just crazy.

# What Does This Code Do?

```
import java.util.*;
HashMap<String, Integer> numLimbs = new HashMap<>();
numLimbs.put(“dog”, 4);
numLimbs.put(“snail”, 1);
// Ooops
numLimbs.put(“octopus”, 9);
// fixed
numLimbs.put(“octopus”, 8);
println(numLimbs.get(“dog”));
println(numLimbs.get(“snail”));
println(numLimbs.get(“octopus”));
```

numLimbs



# What Does This Code Do?

```
import java.util.*;
HashMap<String, Integer> numLimbs = new HashMap<>();
numLimbs.put("dog", 4);
numLimbs.put("snail", 1);
// Ooops
numLimbs.put("octopus", 9);
// fixed
numLimbs.put("octopus", 8);
println(numLimbs.get("dog"));
println(numLimbs.get("snail"));
println(numLimbs.get("octopus"));
```

**Writing .put("octopus", … ) a second time will overwrite the first octopus value!**

```
4
1
8
```

numLimbs

| key | value |
|---|---|
| "dog" | 4 |
| "snail" | 1 |
| "octopus" | 8 |

# More HashMap Commands

## Make a HashMap

```
HashMap<keyType, valueType> myMap = new HashMap<keyType, valueType>();
```

## Put and get values into a map

```
myMap.put(key, value);
myMap.get(key) // returns the corresponding value
```

## Some useful other methods

```
int size = myMap.size();
myMap.containsKey(key); // returns true or false if key is in map
myMap.keySet(); // gets list of keys
myMap.remove(key); // removes a key from a HashMap
```

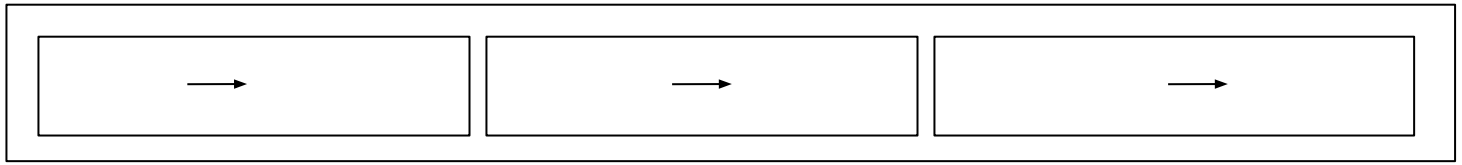
## Iterate using a foreach loop

```
for(keyType key : myMap.keySet()){  // not ordered, so don't expect it to be sorted!
   myMap.get(key); // do something with the key/value pair
}
```

# What Does This Code Do?

```
import java.util.*;
HashMap<String, Integer> numLimbs = new HashMap<>();
numLimbs.put(“dog”, 4);
numLimbs.remove(“dog”);
numLimbs.put(“snail”, 1);
numLimbs.put(“snail”, 105);
numLimbs.put(“octopus”, 8);
println(numLimbs.get(“dog”));
println(numLimbs.get(“snail”));
println(numLimbs.get(“octopus”));
```

numLimbs

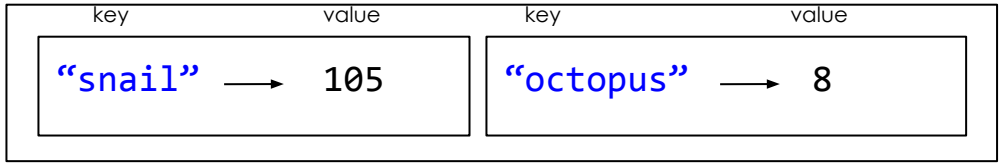


# What Does This Code Do?

```
import java.util.*;
HashMap<String, Integer> numLimbs = new HashMap<>();
numLimbs.put("dog", 4);
numLimbs.remove("dog");
numLimbs.put("snail", 1);
numLimbs.put("snail", 105);
numLimbs.put("octopus", 8);
println(numLimbs.get("dog"));
println(numLimbs.get("snail"));
println(numLimbs.get("octopus"));
```

```
null
105
8
```

numLimbs



# What Does This Code Do?

```
import java.util.*;
HashMap<String, Integer> numLimbs = new HashMap<>();
numLimbs.put("dog", 4);
numLimbs.remove("dog");
numLimbs.put("snail", 1);
numLimbs.put("snail", 105);
numLimbs.put("octopus", 8);
println(numLimbs.get("dog"));
println(numLimbs.get("snail"));
println(numLimbs.get("octopus"));
```

Writing .remove("dog") will completely remove "dog" from our HashMap.

```
null
105
8
```

numLimbs

| key | value | key | value |
|---|---|---|---|
| "snail" | 105 | "octopus" | 8 |

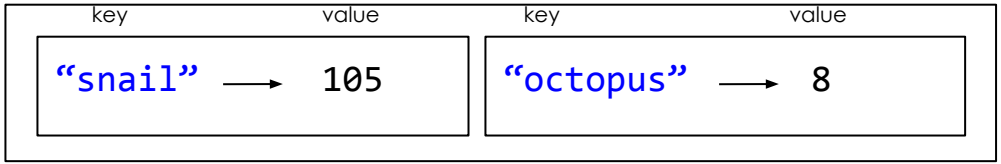
# What Does This Code Do?

```
import java.util.*;
HashMap<String, Integer> numLimbs = new HashMap<>();
numLimbs.put("dog", 4);
numLimbs.remove("dog");
numLimbs.put("snail", 1);
numLimbs.put("snail", 105);
numLimbs.put("octopus", 8);
println(numLimbs.get("dog"));
println(numLimbs.get("snail"));
println(numLimbs.get("octopus"));
```

**Writing .put("snail", … ) a second time will overwrite the first snail value!**

```
null
105
8
```

numLimbs



# You Mentioned Dictionaries…

Hey, I just found this new word today… "Kerfuffle". Do you know what it means?

# Let's Look Up Kerfuffle in Our Dictionary!

I have an "Interesting Dictionary" file that probably contains that word!

Let's write a program to read the file into a HashMap, so we can easily ask our HashMap what the definition of Kerfuffle is!

# Let's Look Up Kerfuffle in Our Dictionary!

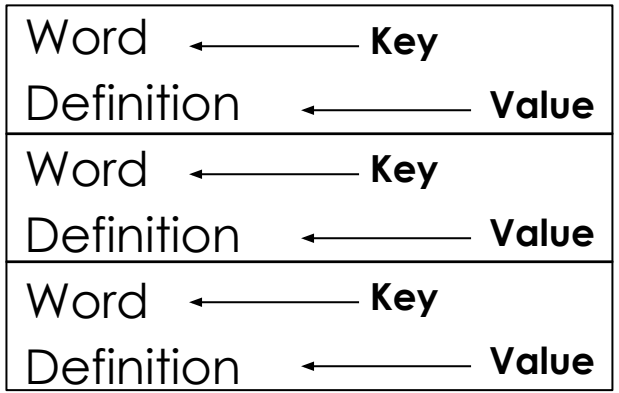
Our file is full of Strings and formatted as follows:

Word
*Definition*
Word
*Definition*
Word
*Definition*

Pseudocode:

# Let's Look Up Kerfuffle in Our Dictionary!

Our file is full of Strings and formatted as follows:




Pseudocode:

```
open file
while there are lines in the file:
        use first line as key in my map
        use second line as value for that key
close file

while user types in another word
        ask which word they would like the def of
        print the associated value/definition from map!
```

# Let’s Code It!

# Code: Printing from our HashMap

```
private void readDefinitions(HashMap<String, String> dict) {
    println("Available Words: ");
    for(String key : dict.keySet()) {
        println(key);
    }
    String word = readLine("Which word would you like a definition of?");
    while(!word.equals("")) {
        if(dict.containsKey(word)) {
            println("The definition of " + word + " is:");
            println(dict.get(word));
        } else {
            println("Sorry, that wasn't a valid word!");
        }

        println();
        word = readLine("Which word would you like a definition of?");
    }
}
```

# Important Note: HashMaps are...

Objects!

This means that HashMaps are **passed by reference**. If we pass a HashMap into a method, **we can change it without needing to return the HashMap**.

# ???

What else can you use HashMaps for?

# Counting Words

Another common use for HashMaps is for **counting occurrences of something.**

Let's write a program to count words in some files I've provided. **Based on the most common words, can you guess what the files are?**

# Let’s Code It!

# Code: Counting Words

```
private void printWordCount(HashMap<String, Integer> dict, int threshold) {
    for(String key : dict.keySet()) {
        if(dict.get(key) >= threshold) {
            println(key + ": " + dict.get(key));
        }
    }
}

private void fillDictionary(HashMap<String, Integer> wordCount){
    try {
        Scanner input = new Scanner(new File(TEXT));
        while(input.hasNext()) {
            String word = input.next();
            word = word.toLowerCase();
            if(wordCount.containsKey(word)) {
                int newCount = wordCount.get(word) + 1;
                wordCount.put(word, newCount);
            } else {
                wordCount.put(word, 1);
            }
        }
        input.close();

    } catch (IOException ex) {
        println("Couldn't open that file!");
    }
}
```

# HashMaps Aren't Sorted!

Remember, HashMaps **are not sorted**. That's why our words and counts weren't in any specific order!

# Extra Exercise: Swapping Keys & Values

Can you write a program which will take a HashMap and swap the keys and values so that they keys are now the values and the values are now the keys?

# Let’s Code It!

# Code: Swapping Keys & Values

```
public void run(){
    HashMap<String, String> dict = new HashMap<String, String>();

    fillDictionary(dict);

    dict = swapKeysAndValues(dict);

    for(String key: dict.keySet()) {
        println(key);
    }

}

private HashMap<String, String> swapKeysAndValues(HashMap<String, String> dict) {
    HashMap<String, String> reversedKeyValues = new HashMap<String, String>();

    for(String key: dict.keySet()) {
        String value = dict.get(key);
        reversedKeyValues.put(value, key);
    }

    dict = reversedKeyValues;
    return dict;
}
```

# 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

# Plan for Today

- Review: Arrays, 2D Arrays, ArrayLists
- HashMaps and Animal Sounds
- Example: Interesting Dictionary
- Example: CountingWords
- Example: Swapping Keys and Values

**Next Time:** More HashMaps and Sets!