

# CS 106A, Lecture 20

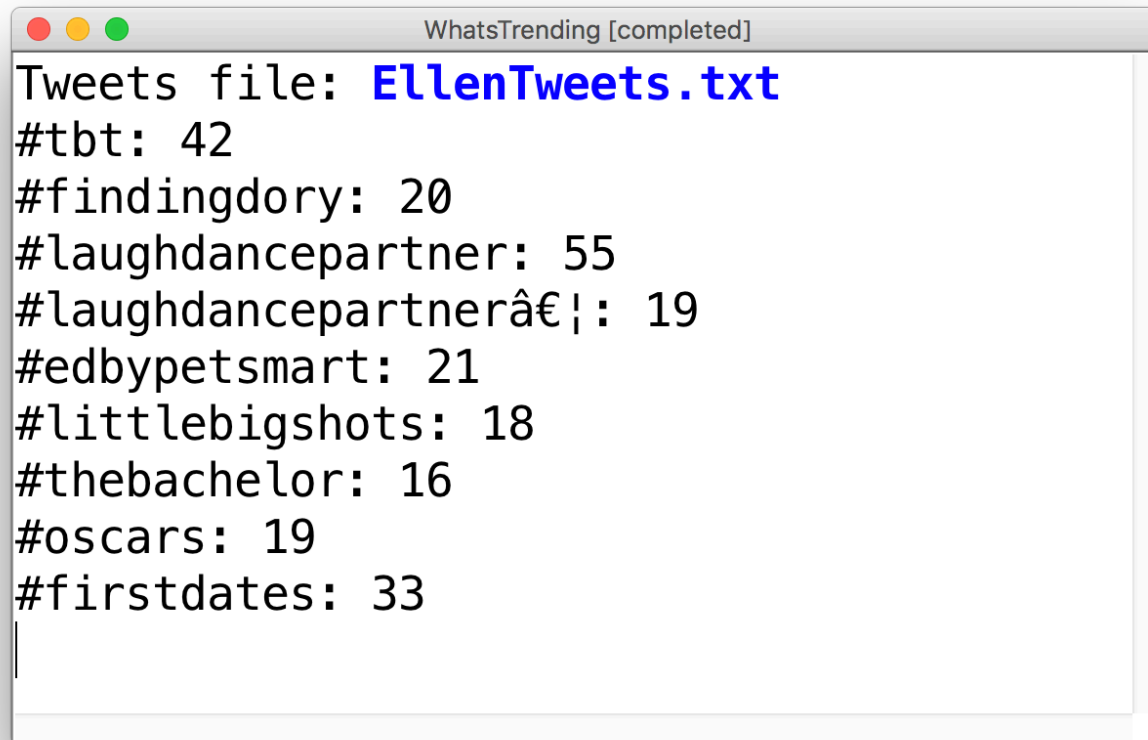
## HashMaps

suggested reading:

*Java Ch. 13.2*

# Learning Goals

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



```
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

- 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
```

**"Hello"**

```
list.add("there!");    // now size 2
```

**"Hello"**

**"there!"**

# ArrayList Methods

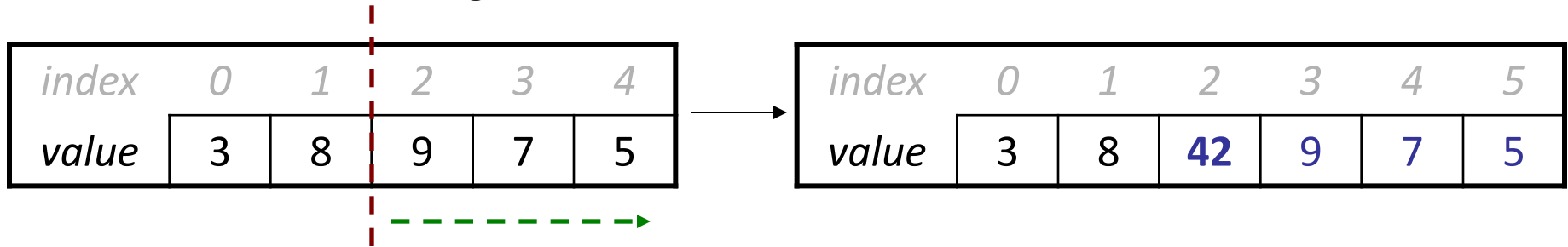
<code>list.add(value);</code>	appends value at end of list
<code>list.add(index, value);</code>	inserts given value just before the given index, shifting subsequent values to the right
<code>list.clear();</code>	removes all elements of the list
<code>list.get(index)</code>	returns the value at given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

# Insert/remove

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

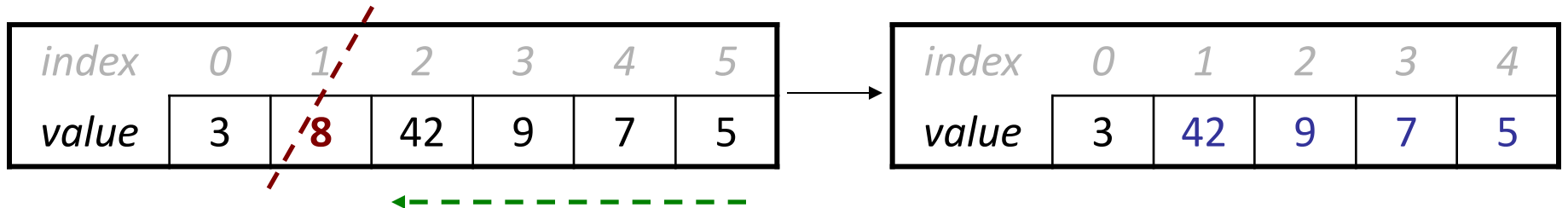
```
list.add(2, 42);
```

- shift elements right to make room for the new element



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

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



# ArrayLists + Primitives =

Primitive	“Wrapper” Class
int	Integer
double	Double
boolean	Boolean
char	Character



# ArrayLists + Wrappers = ❤️

// 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
```

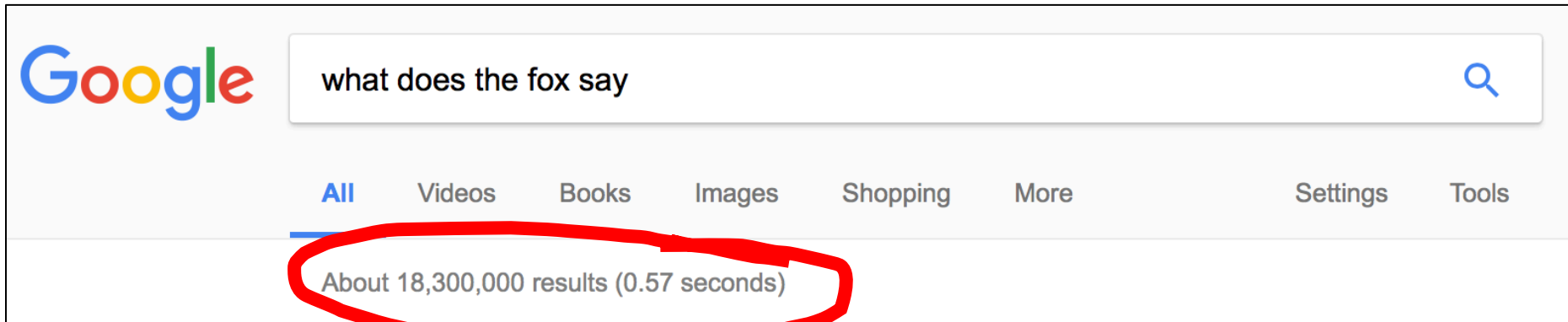
Conversion happens automatically!

# Limitations of Lists

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

<i>index</i>	0	1	2	3	4	5	6	7	8	9
<i>value</i>	12	49	-2	26	5	17	-6	84	72	3

# How Is Webpage Lookup So Fast?



# 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...

# Our First HashMap

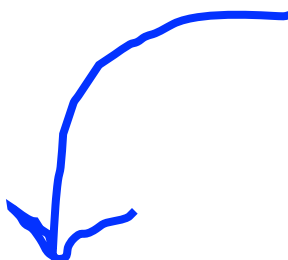
```
import java.util.*;
```

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

# Our First HashMap

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

# Our First HashMap



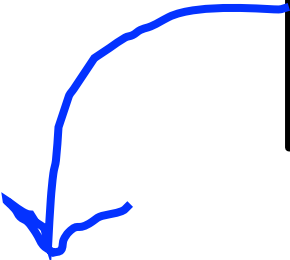
Type of keys your  
HashMap will store.

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



# Our First HashMap

Type of values your  
HashMap will store.



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

# Our First HashMap

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

# Our First HashMap

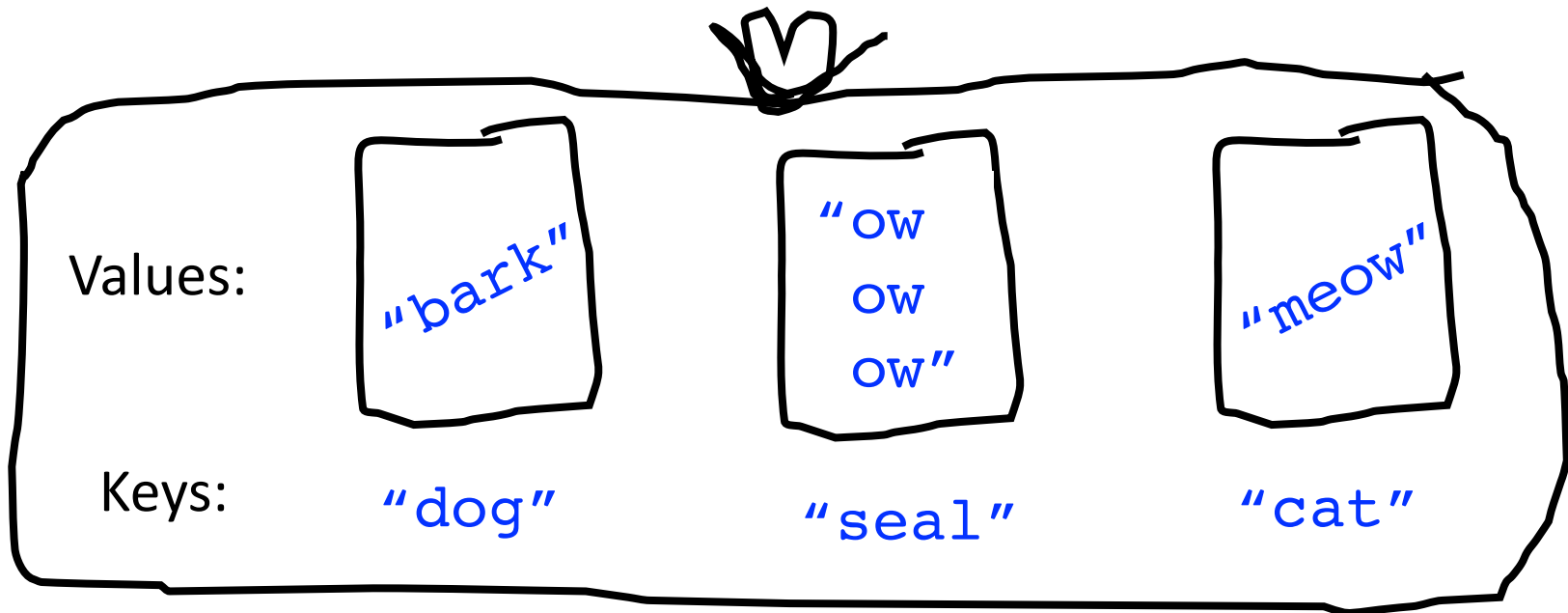
```
HashMap<String, String> myHashMap = new HashMap<>();
```

# Our First HashMap

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

# Our First HashMap - Put

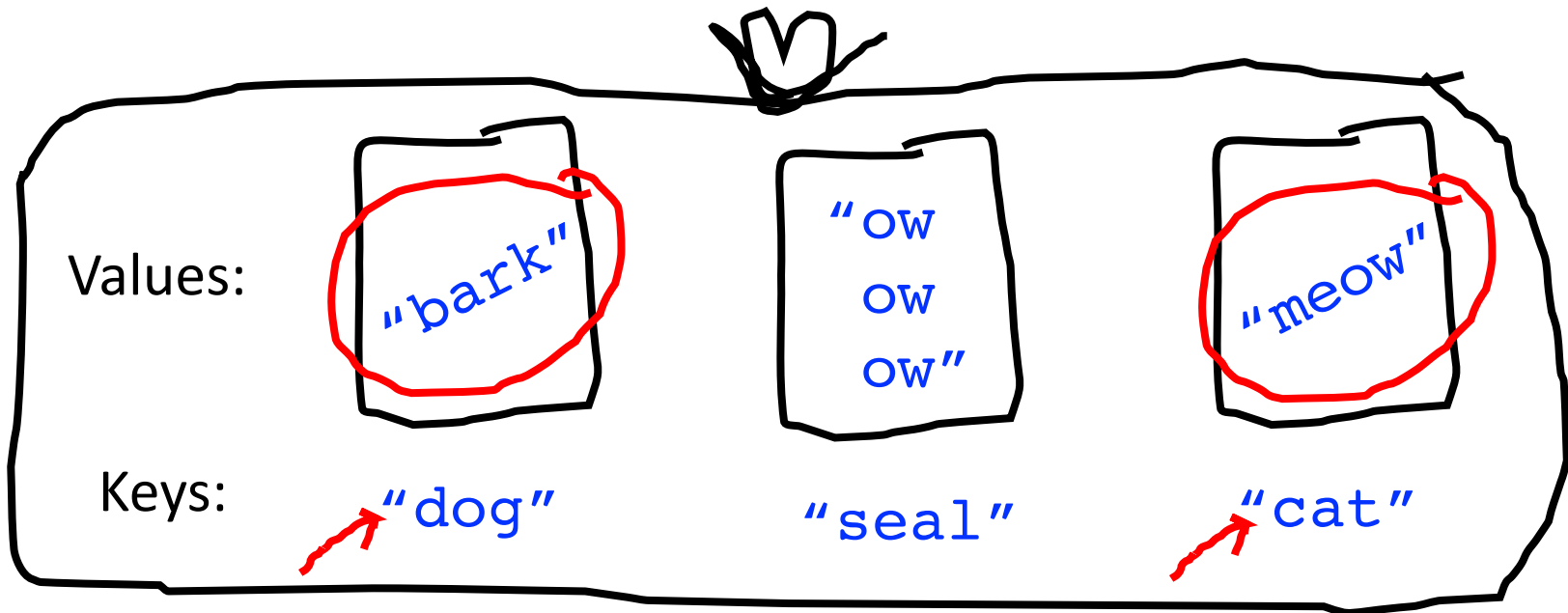
```
// Create an (initially empty) HashMap  
HashMap<String, String> map = new HashMap<>();  
map.put("dog", "bark"); // Add a key-value pair  
map.put("cat", "meow"); // Add another pair  
map.put("seal", "ow ow"); // Add another pair  
map.put("seal", "ow ow ow"); // Overwrites!
```



# Our First HashMap - Get

...

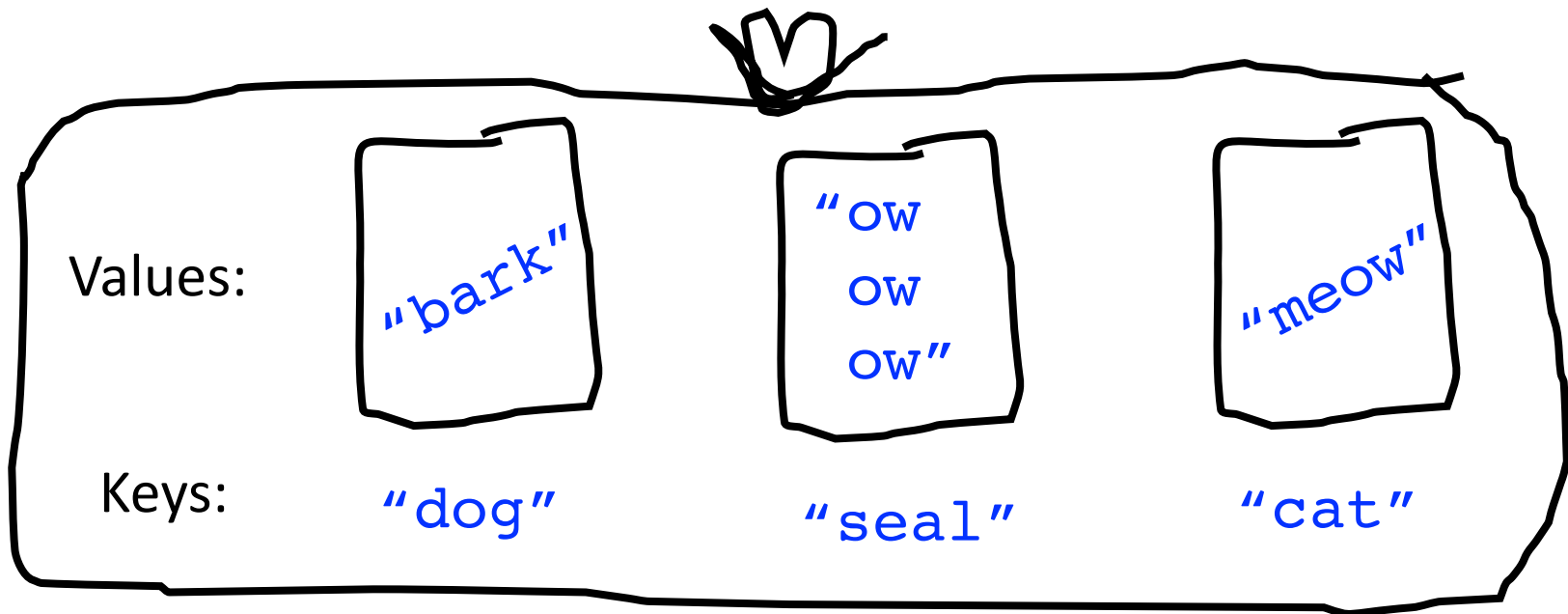
```
String s = map.get("dog"); // Get a value for a key  
String s = map.get("cat"); // Get a value for a key  
String s = map.get("fox"); // null
```



# 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.  
`m.put("Eric", "650-123-4567");`
  - Replaces any previous value for that key.
- **`m.get(key)`** Returns the value paired with the given key.  
`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.  
`m.remove("Rishi");`
  - Has no effect if the key is not in the map.

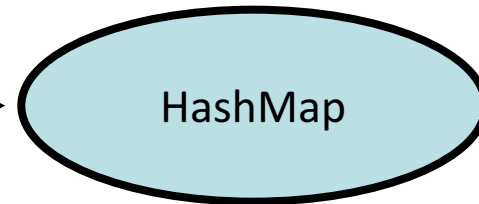
<u>key</u>	<u>value</u>
"Jenny"	→ "867-5309"
"Mehran"	→ "123-4567"
"Marty"	→ "685-2181"
"Chris"	→ "947-2176"



# Using HashMaps

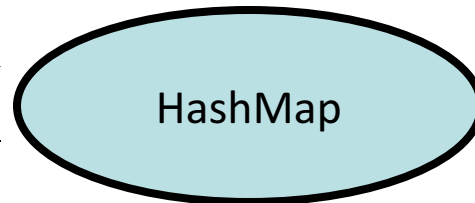
- A HashMap allows you to get from one half of a pair to the other.
  - Remembers one piece of information about every key.

```
//      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"
```



# Practice: Map Mystery

**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

# Practice: Map Mystery

**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");
```

Values:

"Schwarz"

"Sahami"

"Lee"

Keys:

"K"

"M"

"C"

# Practice: Map Mystery

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");
```

Values:

"Schwarz"

"Stepp"

"Lee"

Keys:

"K"

"M"

"C"

# Practice: Map Mystery

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");
```

Values:

"Schwarz"

"Stepp"

"Lee"

Keys:

"K"

"M"

"C"

# Practice: Map Mystery

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");
```

Values:

"Stepp"

"Lee"

Keys:

"M"

"C"

# Practice: Map Mystery

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");
```

Values:

"Cain"

"Stepp"

"Lee"

Keys:

"J"

"M"

"C"

# Practice: Map Mystery

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");
```

Values:

"Cain"

"Stepp"

"Lee"

Keys:

"J"

"M"

"C"



# 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!
```

