HashMaps

Lookup tables in Java

Why HashMaps?

Searching through arrays and ArrayLists for values is labor-intensive. Consider a situation where we want to look up the phone number corresponding to a person's name. On average, we'll have to look through half of the array.

```
String nameToFind = …;   // assume this is initialized
String numberToFind;      // will contain the number we're looking for
String[] names = …;       // contains Strings like "Hunter S Thompson"
String[] numbers = …;     // a String like "(123) 456-7890" corresponding to names[i]
for (int i = 0; i < names.length; i++) {
    if (nameToFind.equals(names[i])) {
        numberToFind = numbers[i];
        break;
    }
}
```

Why HashMaps? (con't)

With a HashMap, you can do the same thing instantly* without all of the work looping through an array.

```
String nameToFind = ...;    // assume this is initialized
String numberToFind;       // will contain the number we're looking for

// assume this is initialized with a bunch of key-value pairs like
// "Hunter S Thompson" -> "(123) 456-7890"
HashMap<String, String> nameToNumber = ...;

// a single line of code!
numberToFind = nameToNumber.get(nameToFind);
```

* not quite instantly, but usually a lot faster. Start at https://www.youtube.com/watch?v=MfhjkfocRR0

Instantiation

To instantiate a new, empty HashMap, you'll need to pick the type of the keys and values. Here's an example where we use String for the key type and String for the value type:

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

Here's an example where we use ints for the keys and booleans for the values. Remember that you have to use object types for the keys and values.

```
HashMap<Integer, Boolean> isPrime = new HashMap<Integer, Boolean();
```

Adding values

To add key-value pairs to a HashMap, use the put() method.

```
HashMap<String, String> namesToNumbers = new HashMap<String, String>();
namesToNumbers.put("Bob Dylan", "(555) 777-8888");
namesToNumbers.put("Carol Reed", "(123) 456-7890");
```

Keys	**Values**
`"Bob Dylan"`	`"(555) 777-8888"`
`"Carol Reed"`	`"(123) 456-7890"`

Getting values

To get a value out of a HashMap, use the get() method. It returns the value if the key-value pair exists in the HashMap; otherwise it returns null.

```
namesToNumbers.get("Bob Dylan");  // returns "(555) 777-8888"
namesToNumbers.get("Meek Mill");  // returns null
```

Keys	**Values**
`"Bob Dylan"`	`"(555) 777-8888"`
`"Carol Reed"`	`"(123) 456-7890"`

Size

Get the number of key-value pairs currently in the hashmap with the size() method.

```
namesToNumbers.size();  // returns 2
```

Keys	**Values**
`"Bob Dylan"`	`"(555) 777-8888"`
`"Carol Reed"`	`"(123) 456-7890"`

Iterating through keys

We know how to get a value if we already know a key, but what if we don't know what keys are available? The keySet() method lets us iterate through all of the keys.

```
Set<String> keys = namesToNumbers.keySet();
for (String key : keys) {
    String value = namesToNumbers.get(key);
    System.out.println(key + ", " + value);
}
```

Keys	**Values**
`"Bob Dylan"`	`"(555) 777-8888"`
`"Carol Reed"`	`"(123) 456-7890"`

Removing values

To take a value out of a HashMap, use the remove() method. It returns the value corresponding to the key that was removed. If the key-value pair did not exist it returns null.

```
namesToNumbers.remove("Bob Dylan");  // returns "(555) 777-8888"
namesToNumbers.remove("Meek Mill");  // returns null
namesToNumbers.size();  // returns 1
```

Keys	**Values**
"Carol Reed"	"(123) 456-7890"

Checking whether a key exists

There's a method called containsKey() that returns true if a particular key exists. It's a shortcut for calling get() and checking that it's not equal to null.

```
namesToNumbers.containsKey("Bob Dylan");   // returns false
namesToNumbers.containsKey("Carol Reed");  // returns true
```

Keys	Values
"Carol Reed"	"(123) 456-7890"

Overwriting values

HashMaps can only contain one value per key. So if you put in a new value for an existing key, the old value gets overwritten.

```
namesToNumbers.put("Carol Reed", "(333) 333-3333");
```

Keys	Values
`"Carol Reed"`	`"(333) 333-3333"`

HashMap documentation

You can read about of these methods and more at the official Java documentation for HashMap:

https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html