While you wait

• Create a class Name
  • Instance variables for first and last name
  • Add .toString that prints the name as first then last. “Joe Smith”
  • Add .hashCode that prints a unique hashCode for your Name object

Read the ClientsList

The mysterious hash

How can I find out if my delicious hash contains a specific “ingredient” in constant time?
### Runtime (Big-oh)

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(Array)List</th>
<th>(Hash)Set</th>
<th>(Hash)Map</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>O(1)</td>
<td>*O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td><strong>Get</strong></td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

### Hashing

- **Hash table**
  - array of fixed size
    - with a key to each location
    - each key is mapped to an index in the table
Hashing

• **HashCode**
  • Every object has a `hashCode`
    • integer value
  
  • In our made-up example
    • Object – `joe`
    • `hashCode` – 31
  
  • Could two different objects have the same `hashCode`?

---

Hashing

• **Hash function**
  • simple to compute
    • example
      • `hashCode % (mod) 10`
  
  • Use hash function to calculate key to hash table
Hashing

- Hash function
  - simple to compute
  - example
    - hashCode % (mod) 10

- Use hash function to calculate key to hash table

Hashing

- Two equal objects should hash to the same place (have the same hash code and key)
Two equal objects should hash to the same place (have the same hash code and key)

```java
if a.equals(b)
    a.hashCode() == b.hashCode()
```

Hash function
- simple to compute
- ensure two distinct keys get different cells
Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells

Separate Chaining

- make your table into a list!
**.hashCode()**

```
if a.equals(b)
then a.hashCode() == b.hashCode()

HOWEVER

if a.hashCode() == b.hashCode()
then a.equals(b) || !a.equals(b)
```

**.equals()**

- Built in Java function for Object
- All objects inherit .equals()
  - You can Override .equals() with your own code!
.equals()

```java
def public boolean equals(Object obj){
    if (obj == this) {
        return true;
    }
    if (obj == null || obj.getClass() != this.getClass()) {
        return false;
    }
    YourObjectType temp = (YourObjectType) obj;
}
```

You do NOT need to memorize this. It will be given to you.

---

How to compare objects?

- Is name1 < name2 or name2 < name1 ???

```java
public class Name implements Comparable<Name>{

    public int compareTo(Name arg0) {
        //name1.compareTo(name2)
        //return negative int if name1 < name2
        //return positive int if name1 > name2
        //return zero if name1.equals(name2)
    }
```

More code

• Add `.equals` to your `Name` class such that a `Name` is equal if both the first and last names are equal.

• Add `.compareTo` that compares names alphabetically by last name then by first name.
  • Jane Doe, John Doe, Jim Smith

ClientsList

APT: ClientsList

Problem Statement

Your company has just undergone some software upgrades, and you will now be storing all of the names of your clients in a new database. Unfortunately, your existing data is inconsistent, and cannot be imported as it is. You have been tasked with writing a program to cleanse the data.

You are given a `String[]`, `names`, which is a list of the names of all of your existing clients. Some of the names are in "First Last" format, while the rest are in "Last, First" format. You are to return a `String[]` with all of the names in "First Last" format, sorted by last name, then by first name.
Sorting with Lists and Arrays

- **ArrayList**

```java
ArrayList<Name> list = new ArrayList<Name>();
Collections.sort(list);
```

- **Array**

```java
Name[] nameArray = new Name[names.length];
Arrays.sort(nameArray);
```

- .sort will call your compareTo!!!