**Important Dates**

- Project 2 due
  - DESIGN, DESIGN, DESIGN!!!
  - You may have 0 or 1 partner.
  - NO EXCEPTIONS!
  - Due Friday 7/24/2009 – 10pm
- Midterm Review
  - Saturday 7/26/2009 – 1-4pm in 306 Soda
- Midterm 2
  - Tuesday 7/28/2009 – 5-6pm in 10 Evans

---

**We want to use it in a TreeMap!**

```java
public class Person implements Comparable<Person> {
    String myName;
    public Person(String name) {
        this.myName = name;
    }
    public String toString() {
        return myName;
    }
    public int compareTo(Person p) {
        return myName.compareTo(p.toString());
    }
}
```

**Using java.util.TreeMap**

```java
public static void main(String[] args) {
    TreeMap<Person, String> myMap = new TreeMap<Person, String>();
    Person jon = new Person("Jon");
    myMap.put(jon, "Blue");
    Person kaushik = new Person("Kaushik");
    myMap.put(kaushik, "Green");
    Person david = new Person("David");
    myMap.put(david, "Blue");
}
```
• Creates a HashTree
• Calls put with a key and a value

• Needs to figure out where the key, value should go
• Uses compareTo() to find the spot in the tree

What was the point?

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jonathan</td>
<td>Blue</td>
</tr>
<tr>
<td>Kaushik</td>
<td>Green</td>
</tr>
<tr>
<td>David</td>
<td>Blue</td>
</tr>
<tr>
<td>George</td>
<td>Green</td>
</tr>
<tr>
<td>Angela</td>
<td>Orange</td>
</tr>
<tr>
<td>Stephanie</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

get: returns the value for that key
put: takes a key and a value and adds them to the map.
containsKey: returns a boolean

What if our keys were 2 letter words and our values their definitions?

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>Rough, cindery lava</td>
</tr>
<tr>
<td>ab</td>
<td>Abdominal muscle</td>
</tr>
<tr>
<td>ad</td>
<td>Advertisement</td>
</tr>
<tr>
<td>ae</td>
<td>One</td>
</tr>
<tr>
<td>ag</td>
<td>Pertaining to agriculture</td>
</tr>
<tr>
<td>ah</td>
<td>To exclaim in amazement</td>
</tr>
<tr>
<td>ai</td>
<td>Three-toed sloth</td>
</tr>
<tr>
<td>al</td>
<td>East Indian tree</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

TreeMap to represent 2 letter words & def.

TreeMap to represent 2 letter words and definitions (approximate picture)

26*26 = 676 nodes (height of 10)

Use a 676 element array!

String word = “aa”
26 * (word.charAt(0) - ‘a’) + (word.charAt(1) - ‘a’);
What are problems with the array representation?

- If we’re storing only 5 entries it is a lot of wasted space!
- If we want to store more than 2 letter words it would get huge quickly!
- We can do better!!! Using a HashTable!
  - Or using java.util.HashMap

Make Person work in a HashMap

```java
public class Person {
    String myName;
    public Person(String name) {
        this.myName = name;
    }
    public String toString() {
        return myName;
    }
    public boolean equals(Object obj) {
        return this.toString().equals(obj.toString());
    }
    public int hashCode() {
        return myName.length();
    }
}
```

```
public static void main(String[] args) {
    HashMap <Person, String> myMap = new HashMap <Person, String>();
    Person george = new Person("George");
    myMap.put(george, "Blue");
    Person stephanie = new Person("Stephanie");
    myMap.put(stephanie, "Yellow");
    Person angela = new Person("Angela");
    myMap.put(angela, "Orange");
    myMap.put(george, "Blue");
    System.out.println(myMap.get(george));
    System.out.println(myMap.get(stephanie));
    System.out.println(myMap.get(angela));
    System.out.println(myMap.containsKey("George"));
    System.out.println(myMap.containsKey("Stephanie"));
    System.out.println(myMap.containsKey("Angela"));
```

```java
myMap.put(george, "Blue");
```

• Creates a HashMap
• Calls put with a key and a value

• Needs to figure out where the key, value should go
• Calls the hashCode() function on the key
• Takes the return value and reduces it with mod to get an index
• Calculates a hashCode value for the object
• Identical object MUST return the same value for hashCode()
Sometimes we have collisions

- In the example in lecture we created a LinkedList when we had a collision (Chaining)
- You can have a collision in two ways.
  - Both objects return exactly the same value from the function `hashCode()`
  - When we get the value from `hashCode()` we need to calculate what bucket to put it in. We use a “compression-function” or in our simple case mod. The compression function can return the same number for two different `hashCode`

**Hashing Summary**

- In class `Object`, is function `hashCode()`
  - Inherited by every class.
  - By default, returns address of this.
  - Can override it for your particular type.
- Rule:
  - If `x.equals(y)` then `x.hashCode() == y.hashCode()`.
  - The inverse need not to be true.
- The types `Hashtable`, `HashSet`, and `HashMap` use `hashCode` to give you fast look-up of objects.