Announcements

Spring pre-registration begins today and runs until Fri, May 6.

The best way to get into the CS course you want is to pre-register.

Common “next steps” after CSCI 136:
CSCI 237: Computer Organization
CSCI 256: Algorithms
CSCI 334: Principles of Programming Languages
also, some electives.

Map ADT

A map (also known as a dictionary, associative array, or key-value store) is an abstract data type that stores a collection of (key, value) pairs. Each key appears at most once in a collection. Maps support lookup, insert, and remove operations.

More formally, a map is a function with a finite domain.
Map ADT (intuition)

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>C</td>
</tr>
<tr>
<td>Jeannie</td>
<td>A</td>
</tr>
<tr>
<td>Bill J</td>
<td>B</td>
</tr>
<tr>
<td>Iris</td>
<td>A</td>
</tr>
<tr>
<td>Sam</td>
<td>A+</td>
</tr>
</tbody>
</table>

You’ve seen something like this before (hint: WordGen)

structure5 Map interface

```java
public interface Map<K, V> {
    boolean containsKey(K k);
    boolean containsValue(V v);
    V get(K k);
    V put(K k, V v);
    int size();
}
```

(I omitted a few methods— see structure5 docs)

structure5’s only Map implementation

```java
public class MapList<K, V> extends java.lang.Object implements Map<K, V> {
    // implementation details...
}
```

Let’s create a tree-backed Map

But first: how will it perform?

What’s the problem with this implementation?
Recap & Next Class

**Today:**
- Map interface
- Tree backed map

**Next class:**
- Hash tables
- Collisions