CSCI 136:
Data Structures
and
Advanced Programming

Lecture 27

Maps

Instructor: Dan Barowy

Williams

Topics

Map interface

Tree backed map

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)

key	value
Dan	C
Jeannie	A
Bill J	B
Iris	A
Sam	A+

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

`structure5` `Map` interface

structure5

Interface Map<K,V>

All Known Subinterfaces:
OrderedMap<K,V>

All Known Implementing Classes:
AbstractMap, ChainedHashtable, Hashtable, MapList, Table

```
public interface Map<K,V>
```

Method Summary	
boolean	**containsKey**(K k)
boolean	**containsValue**(V v)
V	**get**(K k)
V	**put**(K k, V v)
int	**size**()
Structure<V>	**values**()

(I omitted a few methods— see `structure5` docs)

structure5's only `Map` implementation

structure5

Class MapList<K,V>

```
java.lang.Object
  └ structure5.MapList<K,V>
```

All Implemented Interfaces:
Map<K,V>

```
public class MapList<K,V>
extends java.lang.Object
implements Map<K,V>
```

Associations establish a link between a key and a value. An associative array or map is a structure that allows a disjoint set of keys to become associated with an arbitrary set of values. The convenience of an associative array is that the values used to index the elements need not be comparable and their range need not be known ahead of time. Furthermore, there is no upper bound on the size of the structure. It is able to maintain an arbitrary number of different pieces of information simultaneously. Maps are sometimes called dictionaries because of the uniqueness of the association of words and definitions in a household dictionary.

This implementation is based on a list, so performance for most operations is linear.

What's the problem with this implementation?

Let's create a tree-backed Map

But first: how will it perform?

Recap & Next Class

Today:

- Map interface
- Tree backed map

Next class:

- Hash tables
- Collisions