Implementing Maps

Eric Roberts
CS 106B
May 6, 2009

Methods in the `Map<x>` Class

`map.size()` Returns the number of key/value pairs in the map.
`map.isEmpty()` Returns **`true`** if the map is empty.
`map.put(key, value)` *or* **`map[key] = value;`** Makes an association between **`key`** and **`value`**, discarding any existing one.
`map.get(key)` *or* **`map[key]`** Returns the most recent value associated with **`key`**.
`map.containsKey(key)` Returns **`true`** if there is a value associated with **`key`**.
`map.remove(key)` Removes **`key`** from the map along with its associated value, if any.
`map.clear()` Removes all key/value pairs from the map.

An Illustrative Mapping Application

- Suppose that you want to write a program that displays the name of a state given its two-letter postal abbreviation.
- This program is an ideal application for the **`Map`** class because what you need is a map between two-letter codes and state names. Each two-letter code uniquely identifies a particular state and therefore serves as a key for a **`Map`**; the state names are the corresponding values.
- To implement this program in C++, you need to perform the following steps, which are illustrated on the following slide:

  1. Create a **`Map<string>`** containing all 50 key/value pairs.
  2. Read in the two-letter abbreviation to translate.
  3. Call **`get`** on the **`Map`** to find the state name.
  4. Print out the name of the state.