Overview

- "Scheme"-like Lists
- The State Pattern

“Scheme”-like Lists

- See the handout for a complete implementation

```
public abstract class ASchemeList {
  public abstract int car();
  public abstract ASchemeList cdr();
  protected abstract String toString4Cdr();
}

public class Empty extends ASchemeList {
   ...
}

public class Cons extends ASchemeList {
   ...
}
```

A Problem

- Consider

```
public class Test {
  public static void main(String args[])
  {
    ASchemeList list = new Empty(); // Create an empty list.
    ASchemeList anotherRef = list;

    list = new Cons(7, list);       // Add "7" to the list.
    list = new Cons(13, list);      // Add "13" to the list.

    System.out.println(list);
    System.out.println(anotherRef); // What is printed?
  }
}
```

A Solution

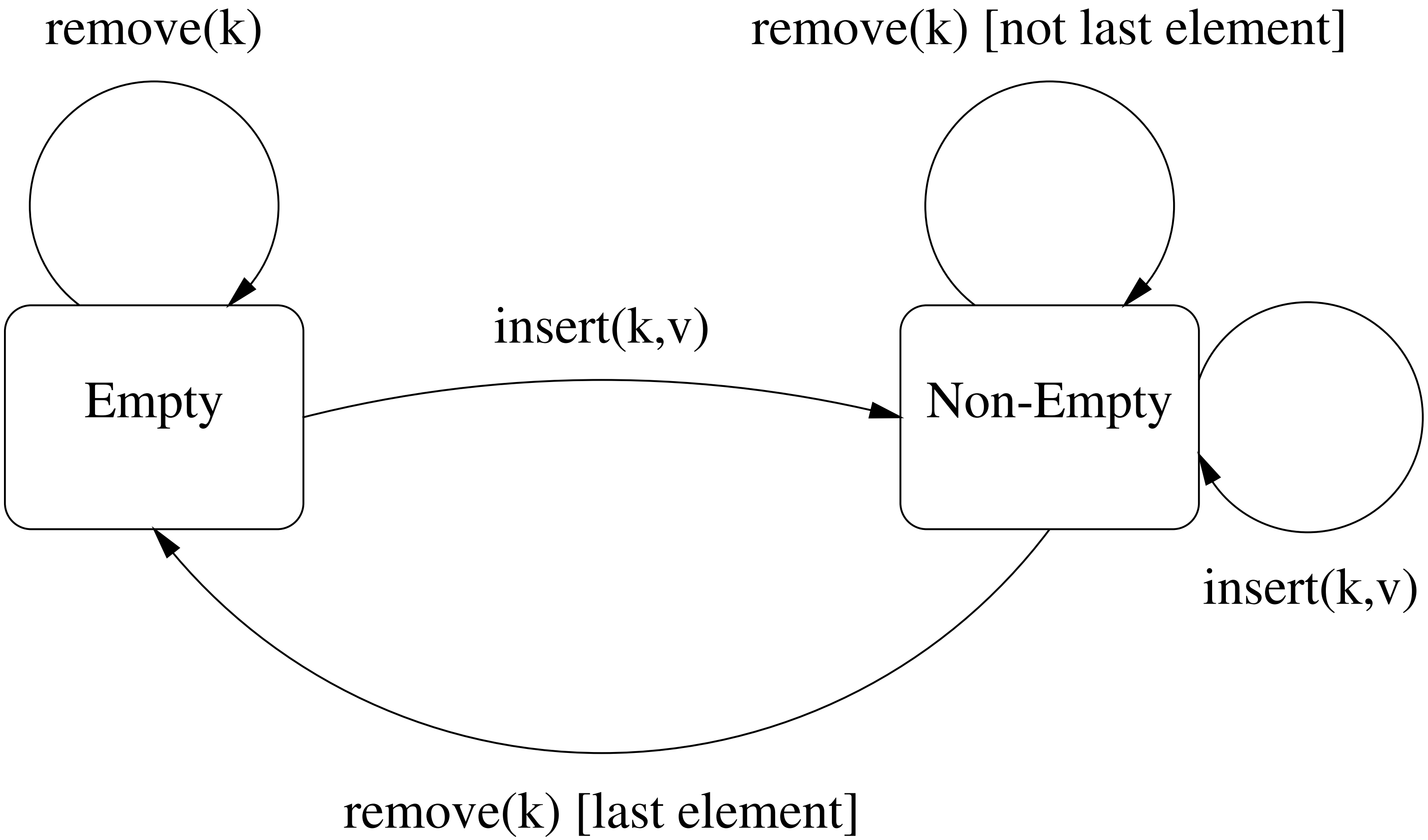
- View the list as a collection of objects ...
  - that have *states* and
  - that can change states dynamically.
- The key is to encapsulate the states of a system as classes.

A General-Purpose Container

- A container for objects

```
public abstract class Container {
  abstract Object find(Object key);
   // If there is an object associated with key
   // then this object is returned else null is returned.
  abstract Object remove(Object key);
   // Afterwards, find(key) returns null, and if there is
   // an object associated with key then this object is
   // returned else null is returned.
  abstract void insert(Object key, Object value);
   // (key, value) is stored in this container with no
   // duplication and such that find(key) returns value.
}
```

Container: State Diagram



The State Pattern

- Define an abstract class for the states of the system.
  - This abstract state class should provide all the abstract methods for all of the concrete subclasses.
- Define a concrete subclass of the abstract class for each state of the system.
  - Each concrete state must implement its own concrete methods.
- Represent the system by a class containing an instance of a concrete state.
  - This instance represents the current state of the system.

The State Pattern (cont.)

- Define methods for the system to return the current state and to change state.
- Delegate all requests made to the system to the current state instance.
  - Since this instance can change dynamically, the system will behave as if it can change its class dynamically.