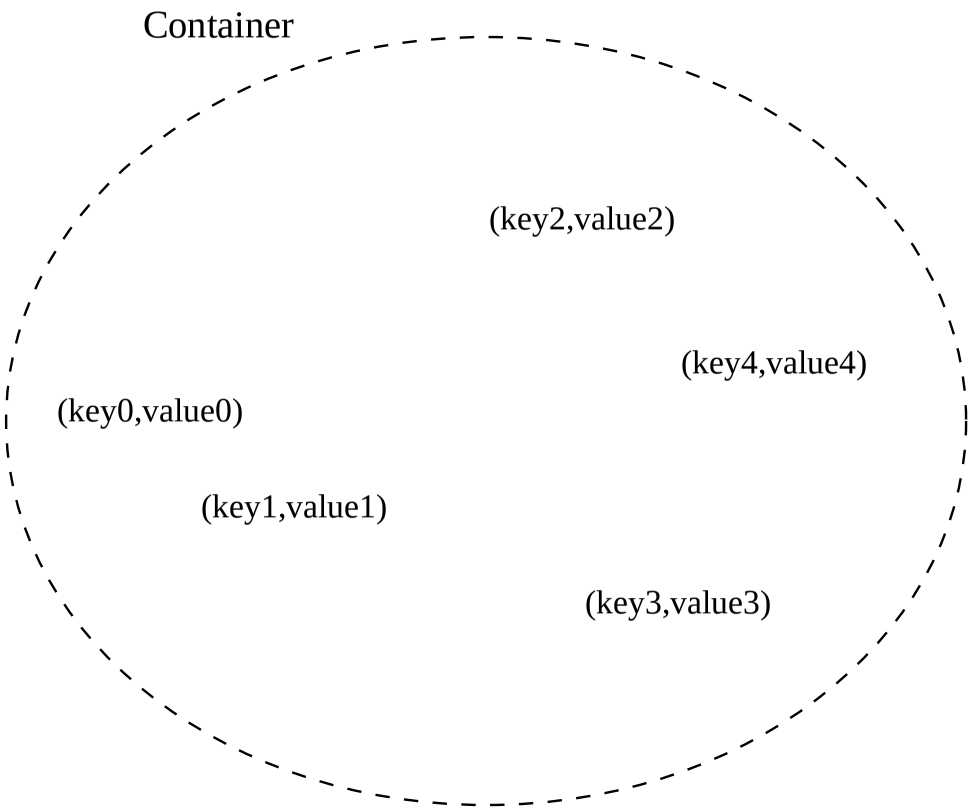
A General-Purpose Container



- A set of ordered pairs, *(key,value)*.
  - No two pairs have the same *key*.

A General-Purpose Container (cont.)

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

A General-Purpose Container: The Implementation

- Containers are easily implemented by composition with `LRStruct`.

```
public class LRSContainer implements IContainer {

  private LRStruct _lrs = new LRStruct();
  ...
  public Object find(Object key)
  {
    return _lrs.execute(LRSFindVisitor.singleton, key);
  }
  ...
  public void insert(Object key, Object value)
  {
    _lrs.execute(LRSInsertVisitor.singleton,
                 new KeyValuePair(key, value));
  }
```

A General-Purpose Container: The Implementation (cont.)

```
class KeyValuePair {
  private Object _key;
  private Object _value;
  ...
  KeyValuePair(Object key, Object value)
  {
    _key = key;
    _value = value;
  }
  ...
  Object getKey()
  {
    return _key;
  }
  ...
```

A General-Purpose Container: The Implementation (cont.)

- Each of the operations on a container are implemented using the Visitor interface to `LRStruct`.

```
class LRSFindVisitor implements IAlgo {

  final static LRSFindVisitor singleton = new LRSFindVisitor();

  private LRSFindVisitor()
  {
  }
  ...
```

A General-Purpose Container: The Implementation (cont.)

```
  public Object emptyCase(LRStruct host, Object input)
  {
    return null;
  }

  public Object nonEmptyCase(LRStruct host, Object input)
  {
    KeyValuePair pair = (KeyValuePair) host.getFirst();

    if (input.equals(pair.getKey()))
        return pair.getValue();
    else
        return host.getRest().execute(this, input);
  }
}
```

equals vs. ==

- What does this program print?

```
public class Main {

  public static void main(String[] args)
  {
    Integer five = new Integer(5);

    if (five == new Integer(5))
        System.out.println("==");
    else
        System.out.println("!=");
  }
}
```

Answer: !=

- Why? They are different objects.
- Many classes provide an `equals` method to address this problem.
  - Example

```
public class Integer {

    private int anInteger;

    public boolean equals(Integer i)
    {
        return anInteger == i.anInteger;
    }
    ...
```

A General-Purpose Container: The Implementation (cont.)

```
class LRSInsertVisitor implements IAlgo {
  ...
  public Object emptyCase(LRStruct host, Object input)
  {
    host.insertFront(input);
    return null;
  }
  ...
```

A General-Purpose Container: The Implementation (cont.)

```
  ...
  public Object nonEmptyCase(LRStruct host, Object input)
  {
    KeyValuePair pair = (KeyValuePair) host.getFirst();

    if (((KeyValuePair) input).getKey().equals(pair.getKey()))
      host.setFirst(input);
    else
      host.getRest().execute(this, input);

    return null;
  }
}
```

A General-Purpose Container: The Implementation (cont.)

```
class LRSRemoveVisitor implements IAlgo {
  ...
  public Object emptyCase(LRStruct host, Object input)
  {
    return null;
  }
  ...
```

A General-Purpose Container: The Implementation (cont.)

```
  ...
  public Object nonEmptyCase(LRStruct host, Object input)
  {
    KeyValuePair pair = (KeyValuePair) host.getFirst();

    if (input.equals(pair.getKey())) {
      host.removeFront();
      return pair.getValue();
    }
    else
      return host.getRest().execute(this, input);
  }
}
```

A General-Purpose Container: A Diagram