Iterators

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming Concepts using Java

Week 4

Linear list

- A generic linear list of objects

Linear list

- A generic linear list of objects
- Internal implementation may vary

Linear list

- A generic linear list of objects
- Internal implementation may vary
- An array implementation

```
public class Linearlist {
  // Array implementation
  private int limit = 100;
  private Object[] data = new Object[limit];
  private int size;  // Current size

  public Linearlist(){ size = 0; }

  public void append(Object o){
    data[size] = o;
    size++;
    ...
  }
  ...
}
```

Linear list

- A generic linear list of objects
- Internal implementation may vary
- An array implementation
- A linked list implementation

```
public class Linearlist {
  private Node head;
  private int size;

  public Linearlist(){ size = 0; }

  public void append(Object o){
    Node m;

    for (m = head; m.next != null; m = m.next){}
    Node n = new Node(o);
    m.next = n;

    size++;
  }
  ...
  private class Node (...}
}
```

Iteration

- Want a loop to run through all values in a linear list

Iteration

- Want a loop to run through all values in a linear list
- If the list is an array with public access, we write this

```
int i;
for (i = 0; i < data.length; i++){
  ... // do something with data[i]
}
```

Iteration

- Want a loop to run through all values in a linear list
- If the list is an array with public access, we write this
- For a linked list with public access, we could write this

```
int i;
for (i = 0; i < data.length; i++){
  ... // do something with data[i]
}
```

```
Node m;
for (m = head; m != null; m = m.next}
  ... // do something with m.data
}
```

Iteration

- Want a loop to run through all values in a linear list
- If the list is an array with public access, we write this
- For a linked list with public access, we could write this
- We don't have public access . . .

```
int i;
for (i = 0; i < data.length; i++){
  ... // do something with data[i]
}
```

```
Node m;
for (m = head; m != null; m = m.next}
  ... // do something with m.data
}
```

Iteration

- Want a loop to run through all values in a linear list
- If the list is an array with public access, we write this
- For a linked list with public access, we could write this
- We don't have public access . . .
- . . . and we don't know which implementation is in use!

```
int i;
for (i = 0; i < data.length; i++){
  ... // do something with data[i]
}
```

```
Node m;
for (m = head; m != null; m = m.next}
  ... // do something with m.data
}
```

Iterators

- Need the following abstraction

```
Start at the beginning of the list;
while (there is a next element){
  get the next element;
  do something with it
}
```

Iterators

- Need the following abstraction

```
Start at the beginning of the list;
while (there is a next element){
  get the next element;
  do something with it
}
```

- Encapsulate this functionality in an interface called `Iterator`

```
public interface Iterator{
  public abstract boolean has_next();
  public abstract Object get_next();
}
```

Iterators

- How do we implement `Iterator` in `Linearlist`?

Iterators

- How do we implement `Iterator` in `Linearlist`?
- Need a "pointer" to remember position of the iterator

Iterators

- How do we implement `Iterator` in `Linearlist`?
- Need a "pointer" to remember position of the iterator
- How do we handle nested loops?

```
for (i = 0; i < data.length; i++){
  for (j = 0; j < data.length; j++){
     ... // do something with data[i] and data[j]
  }
}
```

Iterators

- Solution: Create an `Iterator` object and export it!

Iterators

- Solution: Create an `Iterator` object and export it!

```
public class Linearlist{

  private class Iter implements Iterator{
    private Node position;
    public Iter(){...}   // Constructor
    public boolean has_next(){...}
    public Object get_next(){...}
  }

 // Export a fresh iterator
  public Iterator get_iterator(){
    Iter it = new Iter();
    return(it);
  }
 }
```

Iterators

- Solution: Create an `Iterator` object and export it!

```
public class Linearlist{

  private class Iter implements Iterator{
    private Node position;
    public Iter(){...}   // Constructor
    public boolean has_next(){...}
    public Object get_next(){...}
  }

 // Export a fresh iterator
  public Iterator get_iterator(){
    Iter it = new Iter();
    return(it);
  }
 }
```

- Definition of `Iter` depends on linear list

Iterators

- Now, we can traverse the list externally as follows:

```
Linearlist l = new Linearlist();
...
Object o;
Iterator i = l.get_iterator();

while (i.has_next()){
  o = i.get_next();
  ...   // do something with o
}
...
```

Iterators

- Now, we can traverse the list externally as follows:

```
Linearlist l = new Linearlist();
...
Object o;
Iterator i = l.get_iterator();

while (i.has_next()){
  o = i.get_next();
  ...   // do something with o
}
...
```

- For nested loops, acquire multiple iterators!

```
Linearlist l = new Linearlist();
...
Object oi,oj;
Iterator i,j;

i = l.get_iterator();
while (i.has_next()){
  oi = i.get_next();
  j = l.get_iterator();
  while (j.has_next()){
    oj = j.get_next();
    ... // do something with oi, oj
  }
}
...
```

Summary

- Iterators are another example of interaction with state
  - Each iterator needs to remember its position in the list
- Export an object with a prespecified interface to handle the interaction
- The new Java `for` over lists implicitly constructs and uses an iterator

```
for (type x : a)
  do something with x;
}
```