

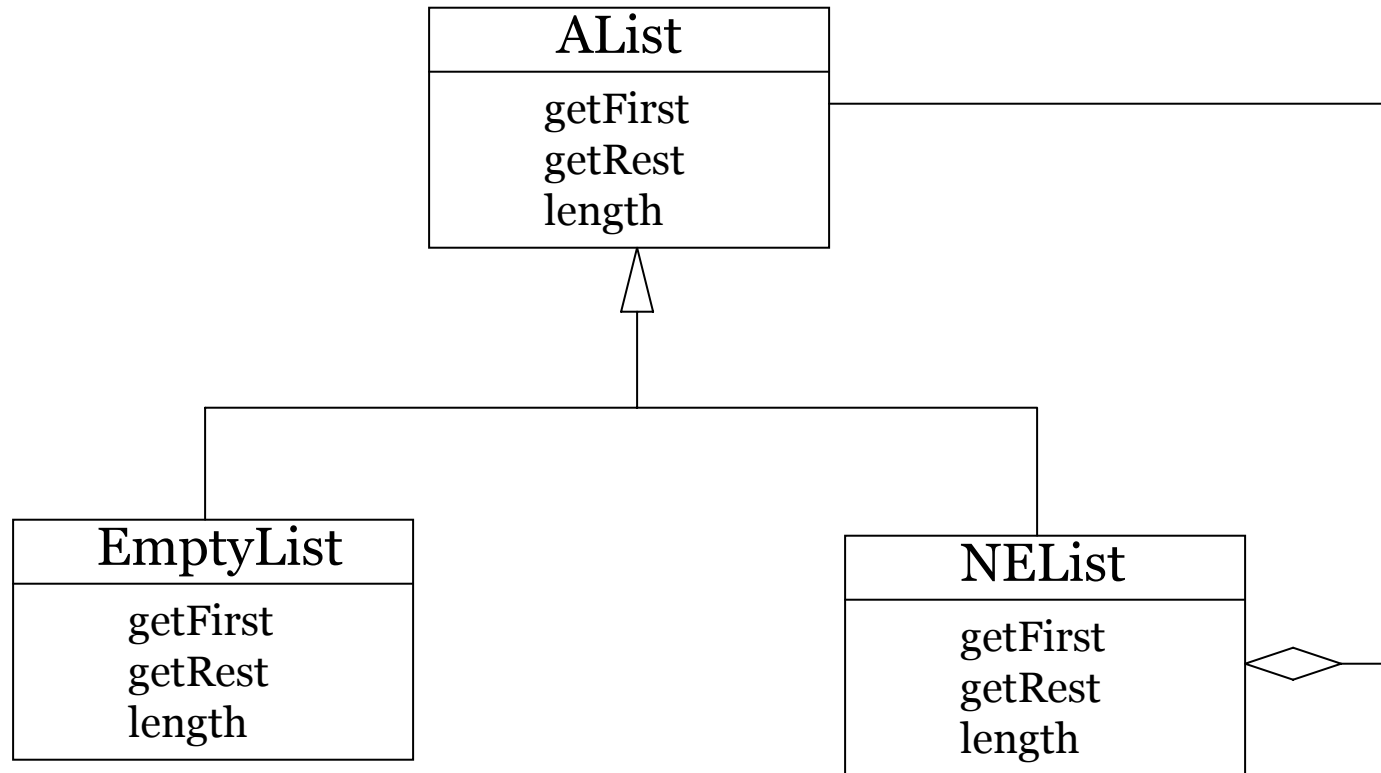
CS 91.204 Computing IV:
Advanced OO Programming and
Software Tools
Interface, Visitor Pattern

Prof. Li Xu
Dept. of Computer Science
UMass Lowell
Spring 2009

Midterm

- Midterm exam 3/11, 3/13
 - two sessions
 - Java and OO concepts
 - OO design, design patterns
 - programming problems
 - will include visitor pattern

OO List Design



Problem with OO List Design

- Each time we want to a new list op, we edit each class in order to add methods to it.
 - length
 - sum
 - toString
 - findMin
 - merge
 - filter
 - reverse
 - ...

Problem with OO List Design

- Is there a way to add new behavior to AList without touching any of the existing code, leaving everything that has been written so far unchanged?
- In C, we can leave data structure in place, and then manipulate the data thru direct access, think your linked list.
- How can we do this OO style and elegantly?

Toward a Solution

- The key is to abstract out the different behaviors of list operations in a separate Union Pattern (OOP Principle #1:Abstraction).
 - ABC will represent list op abstraction
 - The *invariant* behaviors are data structure access: the constructor and methods `getFirst()` and `getRest()`.
 - The *variant* behaviors are the infinitely many algorithms (operations) that we want AList to perform.

Toward a Solution

- Separate abstraction of algorithm from list data structure
 - host and visitor abstraction
- For AList to execute any of these algorithms, we just need to add one more method to AList.
 - Abstract the algorithms as generic visitor class which will perform operations.
 - The AList method will “invite” the abstract visitor class to “visit” the list object
 - Each List operation can be a concrete visitor class
 - New operations become new visitors – we never need to modify the old code (data structures)
 - visitor has knowledge how to deal with the variant cases in host union

The Visitor Pattern

- The visitor pattern is a framework for communication and collaboration between two union patterns: a "host" union and a "visitor" union.
 - An abstract visitor is usually defined as an interface in Java.
 - It has a separate method for each of the concrete variants of the host union.

Java Interface

- Java interface syntax

```
public interface IFoo {  
... //method declaration  
}
```

```
public class Foo implements IFoo {  
...//method implementation  
}
```

- Interface methods are public, abstract

The Host

```
public abstract class AList {  
    public Object getFirst();  
    public AList getRest();  
    public abstract Object invite(IListVisitor visitor,  
        Object input);  
}
```

Host

- public abstract Object invite(IListVisitor visitor, Object input)
 - defines a generic operation object
 - client passes in a visitor object and ask the host to invite it to visit
 - input provides additional argument
 - return value is Object to be generic

The Visitor

```
public interface IListVisitor {  
    Object visitEmptyList(AList host,  
        Object input);  
    Object visitNEList(AList host,  
        Object input);  
}
```

The Host: Concrete Variant

```
class EmptyList extends AList {  
    public Object getFirst() {...}  
    public AList getRest() {...}  
    public Object invite(IListVisitor visitor, Object input) {  
        return visitor.visitEmptyList(this, input);  
    }  
}
```

The Host: Concrete Variant

```
class NEList extends AList {  
    public Object getFirst() {...}  
    public AList getRest() {...}  
    public Object invite(IListVisitor visitor, Object input) {  
        return visitor.visitNEList(this, input);  
    }  
}
```

Concrete Visitor

- Length visitor
 - object instance implement the visitor interface
 - l.invite(new Length())