Lecture 9

Lists I

- Conversations at Software Company
- Interfaces and Inheritance
- List and Friends
Conversations at a Software Company
We need a data structure for storing a bicycle ride. We want to add (time, location) pairs and compute speeds.

OK, my team will design an interface.

There could be GPS errors, so we should include delete.

Maybe we can inherit from the Vector class?
Here is an interface. Let us know if you need any additions.

OK, thanks.

Let's do a simple first implementation without optimizations.

We can use a singly linked list.

With this interface we can add the data like this.
Conversations at a Software Company

Here is our first implementation.

OK, thanks.

The GPS data is noisy and we need a faster delete method.
Can you improve the efficiency of delete?

Let's switch to a doubly linked list.

We need a faster delete.
Interfaces and Inheritance
and Abstract Classes
Interfaces

An interface provides a list of methods, but no specific implementation for these methods.

When a class implements an interface, it promises to implement these methods unless it is Abstract.

Java keywords: interface and implements.

Benefits of interfaces:

- Two classes can be used in the same way (i.e., if they implement the same interface).
- A class can be used in different ways (i.e., if it implements several interfaces).
- Similar modularity benefits as classes (i.e., know the use without the implementation, and the implementation can be changed, etc.)
Inheritance

When a class Y extends or inherits from another class X, it takes on all of its properties and methods.

- X is often called the base class or parent class.
- Y is often called the derived class or child class.

In addition, Y can add new properties and methods. It can also change the implementation of base methods.

Java keyword: extends

Warning: You may start seeing inheritance everywhere!

Benefits of inheritance:

- Use different derived classes in the same way (i.e., “feed all animals” regardless of which type)
- Save time and avoid errors by implementing common behaviors once inside of base classes.

```java
public class SinglyLinkedList<E> extends AbstractList<E> {
    // The number of elements in list.
    protected int count;   // list size

    // The head of the list. A reference to a singly linked list.
    protected Node<E> head; // ref. to first element

    // Construct an empty list.
    public SinglyLinkedList() {
        head = null;
        count = 0;
    }
}
```

A frog is an amphibian, which is a cold-blooded vertebrate, which is an animal.
Abstract Classes

When designing object and class hierarchies, it is sometimes helpful to supply an abstract class.

An abstract class cannot directly be “instantiated”. That is, objects of the class can’t be made with `new`.

A class is abstract if it has some unimplemented methods, or if it is specified to be abstract.

Java keyword: `abstract`

Benefits of abstract classes:

- Save time and avoid errors by implementing common behaviors once inside of abstract classes.

Vertebrates is a useful abstract classification. We cannot make a vertebrate.

The `AbstractList` class is abstract. Non-abstract classes like `SinglyLinkedList` inherit from it.
List and Friends
Activity: Understanding List and its Relationships in the structure Package

To properly understand and use the structure package, we need to be able to investigate the relationships between the various classes.

In this activity, you will focus on the relationships between the List class and the following: AbstractList, AbstractStructure, CircularList, DoublyLinkedList, DoublyLinkedNode, Node, SinglyLinkedList, Structure, Vector

Related questions:

● How would you diagram this information?
● Are you surprised by Vector’s relationship to List?
● What is the benefit of AbstractList?
● How can you find which other classes use List?
grep can be used on the command-line to help you search for patterns in files. For example, running `grep Node *.java` from your `~/cs136/js/src/structure` folder reveals that `Node` is used in a dozen files in the `structure` package.
Note: During our discussion, it was pointed out that the implementation of `isEmpty()` in `AbstractList` is redundant, since the class extends `AbstractStructure`, which implements `isEmpty()` in the same way. Nice observation!
Extra Time?
Next class we’ll return to singly linked lists.

- Try to understand how its `addLast` and `removeLast` methods are implemented.
Above are the same methods implemented in the DoubleLinkedList class.
Why are the implementations in a doubly linked list simpler?