Com S 228
Fall 2010
Programming Assignment 3
Due at 11:59 pm Friday, October 29
200 points

This assignment is to be done on your own. If you need help, see me or one of the TAs. Please make sure you understand the "Academic dishonesty" section of the syllabus.

Please start the assignment as soon as possible and get your questions answered *early*. Read through this specification completely before you start.

Remember that Midterm 2 will be on Thursday, October 28 and will cover linked lists and iterators. Although the deadline for this assignment is not until after the exam, to allow you extra time to finish up details, you are very strongly encouraged to do this assignment well before the exam, particularly the first item described in "Some ideas for getting started."

Introduction

The purpose of this assignment is to give you some practice working with linked lists and the ListIterator interface.

WebCT

The WebCT discussion for Assignment 3 is a good place to post general questions. Please do not post or attach any source code for the assignment. Also check the "official clarification" thread for corrections, hints, and answers to common questions. *Any clarification posts prior to 24 hours before the deadline are considered to be part of this specification.*

Summary of tasks

In this assignment you will implement one class, edu.iastate.cs228.hw3.PriorityList (including several inner classes) that implements a subset of the java List interface plus the IPriorityList interface defined for this assignment.

Overview

The data structure to be implemented is basically a doubly-linked, null-terminated list with a dummy node at the head. Each item in the list is associated with a *priority*, which is an integer from 0 up through some fixed maximum value *max*. An array of length *max+1* will store, for each priority, a reference to the first element of that priority and the number of elements of that priority. All elements of a given priority are grouped together as a sublist, and all elements of a given priority come before elements of higher priority. The structure in its entirety satisfies the

Figure 1 - an empty list with three priority levels

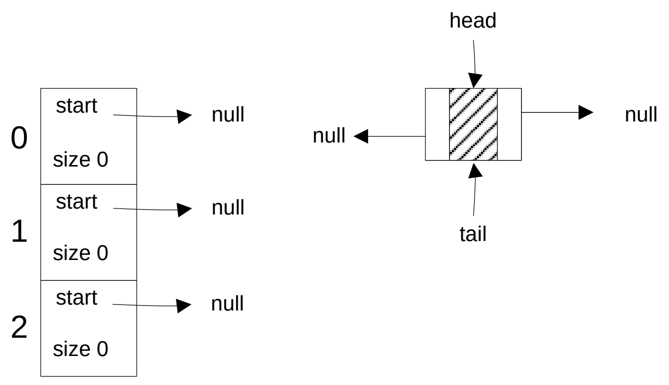
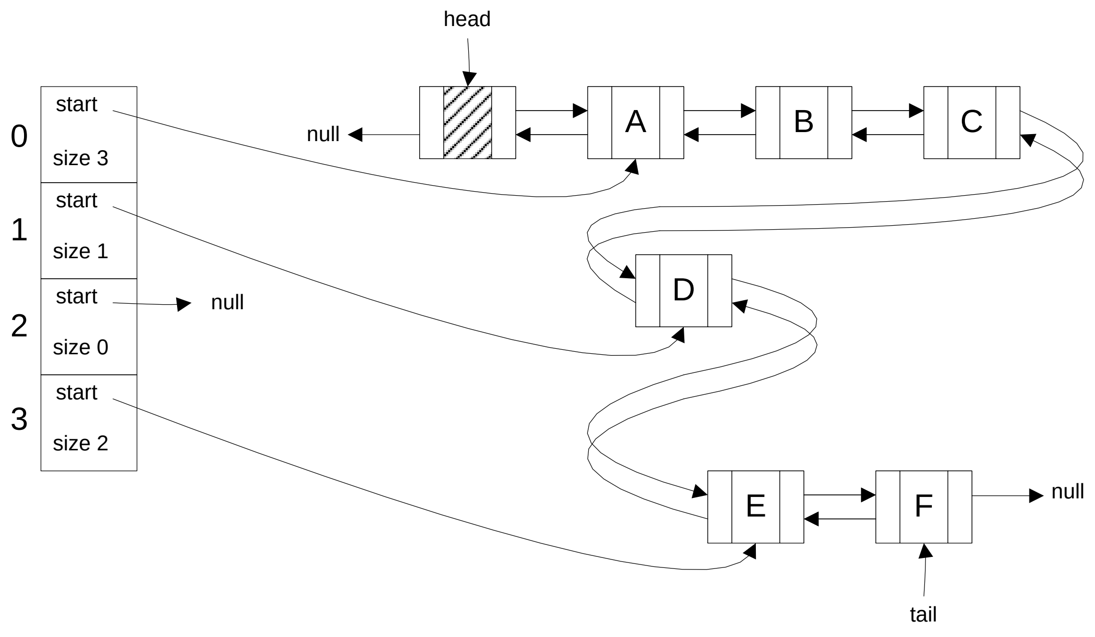


Figure 2 - a six-element list with four priority levels



List interface (actually a subset of the List interface specified below). That is, elements can be inserted and removed as in an ordinary list without specifying a priority. For example, in Figure

2, the structure appears to the client as a List containing six elements A, B, C, D, E, and F. If an element is added at index 2, it will automatically be grouped with the priority 0 elements.

The IPriorityList interface

However, in addition to the List operations, there is a small set of priority-related operations, defined in the interface IPriorityList, summarized below.

```
// adds the item at the END of the sublist for the given priority
void addWithPriority(int priority, E item);

// removes and returns the element at the BEGINNING of the sublist for
// the given priority
E removeWithPriority(int priority);

// returns the size of the sublist for the given priority
int sizeWithPriority(int priority);

// returns a ListIterator for the elements of the given priority
ListIterator<E> iteratorWithPriority(int prio);

// returns the maximum priority
int getMaxPriority()
```

The ListIterator returned by iteratorWithPriority() behaves as though the elements of the given priority are the only elements in existence. For example, in Figure 2, the call iteratorWithPriority(3).nextIndex() returns 0, not 4.

Required methods of the List interface

Include the following methods. All other methods of the List interface should throw UnsupportedOperationException. Indexed operations (add, addAll, get, remove, set) may not rely on the iterator (see item 4 below) but you can write the others using the iterator if you wish.

```
add(E item)
add(int pos, E item)
addAll(Collection<? extends E> c)
addAll(int pos, Collection<? extends E> c)
clear()
contains(Object obj)
get(int pos)
indexOf(Object obj)
isEmpty()
iterator()
listIterator()
listIterator(int pos)
```

Continued on next page

```
remove(Object obj)
remove(int pos)
set(int pos E item)
size()
```

Notes

1. Your class must include a one-argument constructor that specifies the number *max + 1* of priority levels. (This value does not change).

2. All operations that add or remove elements must correctly update the sublist size for the appropriate priority as specified in (3) and (4) below.

3. The add(E item) operation adds to the end of the entire list, and the added element is considered to be of the same priority as the last element currently in the list. If the list is empty, the element is added at priority 0.

4. The add(int pos, E item) operation adds the item at the same priority level as the element currently at position pos. For example, in Figure 2, add(2, X) would put X before C at priority 0, while add(3, X) would put X before D at priority 1. Adding at position 0 inserts at the same priority as the first element currently in the list, and inserts at priority 0 if the list is empty. Adding at position size() inserts at the same priority as the last element currently in the list, or at priority 0 if the list is empty. The version for adding a collection should behave similarly.

5. All indexed operations should rely on the size information for sublists to locate the correct priority level corresponding to a given position. For example, add(4, X) should not iterate over elements A, B, C, and D; rather, it should iterate over the array and use the fact that the total of the sizes of the first three sublists is less than or equal to 4, so it can start searching for position 4 in the priority 3 sublist.

6. A ListIterator for the entire list works without regard to priority, with the exception of the iterator’s add() and remove() methods. An item to be added is always placed at the same priority level as the element *before* the cursor position. For example, if the logical cursor is between C and D, a call to add(X) will insert the new element at priority 0 after C, rather than at priority 1. If there is no element before the cursor, add(X) will insert the new element just before the first element (and at the same priority). If the list is empty, the element is added at priority zero (same as calling add(item) on an empty list). The remove() operation must decrement the appropriate sublist size.

7. The ListIterators for a sublist of a given priority should not reimplement the main list iterator. You should be able to either adapt the iterator (i.e., use a special constructor and treat the sublist iterator as a special case), or extend it, to work correctly for the sublists.

Some ideas for getting started

Start by ignoring the priority stuff and just implement the list operations for a null-terminated, doubly-linked list with a dummy node at the head. This is pretty straightforward. The examples presented in class using singly-linked nodes are relevant, but doubly-linked nodes are actually simpler to work with. The text includes detailed examples of basic operations on doubly-linked nodes as well as implementation examples for the List interface. A significant fraction of the testing we will do in grading your assignment will simply be to check whether your implementation conforms to the List and ListIterator interfaces, without regard to the priorities.

Then, create an appropriate inner class for the array elements (i.e., a class with two instance variables for the start position and size for each sublist). Implement the addWithPriority and removeWithPriority operations.

Modify the add and remove operations so that they correctly update the sublist size as described in items (2)--(4) above.

Modify the index-based list operations to use the sublist sizes as described in (5). (Note you might want to start by writing a helper method that, given an integer index, returns the priority level for the sublist containing that index.)

Modify the list iterator so that the add and remove operations correctly update the sublist size.

Extend or adapt your list iterator so that you can use it for the sublists.

Documentation and style

Up to 15% of the points will be for documentation and style. See the style guidelines posted on WebCT.

What to turn in

All of your code should be in the class edu.iastate.cs228.hw3.PriorityList. Submit a zip file, containing your source code for that class and the source code for the IPriorityList interface, in the correct package structure. The top-level directory should be edu.

See the submission HOWTO on the WebCT main page if you not certain how to do this. You **will** lose points for submission errors (and in the worst case, if you do not correct your submission errors before the late deadline, you may receive no credit at all for the assignment). It is your responsibility to check the file that you have uploaded to WebCT and make sure that it is complete and correct.

Late Penalties

Assignments may be submitted up to 24 hours after the deadline with a 25% penalty (not counting weekends/holidays). No credit will be given for assignments submitted after that time.