Announcement

Midterm exam will be held at 4:00 pm on Wed., Feb. 18.

Programing Assignment # 2

Due Date

Sunday, March 15, 11:59 pm. Submit your files to eLearning.

Late Policy

-1 point per minute late.

Instruction

This is an individual assignment. All work should be your own.

Objective

Create and work with a new implementation of the abstract data type List.

Description

You are requested to create a new data structure named ArrayLists, implementing the ADT List. ArrayLists is a combination of ArrayList and LinkedList. It is a 2-level data structure. At the first level is an ArrayList. Each element of the ArrayList has two fields, a LinkedList called sublist, of a fixed maximum size (say 4), and an integer lastIndex, which is the index of the last item in the sublist with respect to the entire List. All sublists are on the second
level. Conceptually, the concatenation of all sublists constitutes the List. Your program should be generic, i.e. it can be used for various types for data items in the List.

Assume the smallest index is 0. Consider Fig. 1. The ArrayList has 3 non-empty elements. The first element contains a sublist of 4 items, with indices 0, 1, 2 and 3 respectively. The second element has a sublist of 1 item with index 4. The third element has a sublist of 4 items, with indices 5, 6, 7 and 8, respectively.

Figure 1: An ArrayofLists of 9 items. The numbers associated with the items of sublists are the indices of item in the corresponding List.

Suppose that we want to remove the item of index 6. We use binary search on the lastIndex fields of ArrayofLists to locate the sublist the item 6 is in (the third sublist), and delete it from this sublist to obtain the ArrayofLists structure shown in Fig.2. Note that the lastIndex values are updated.

Figure 2: The ArrayofLists after item with index 6 in Fig. 1 is removed.

Now, suppose we want to add a new item as the item of index 4. Then this element is added in the second sublist, and the lastIndex fields of part of ArrayofLists must be updated. This is shown in Fig. 3.

Suppose that we want to remove the item of index 4 in Fig. 2. Then, after removal of this item, the second sublist becomes empty. The non-empty sublist(s) to the right of this empty sublist must be shifted left, and their lastIndex fields must be updated. This is shown in Fig. 4.
Figure 3: The ArrayofLists after a new item is added as the item of index 4 to Fig. 2.

Figure 4: The ArrayofLists the item index 4 is removed from Fig. 2.
It is not difficult to see that the performance of **ArrayofLists** is a trade-off between **LinkedList** and **ArrayList**. When the size of sublist is large, **ArrayofLists** may have a small number of sublists. It is more or less like an **LinkedList**. If the size of sublists is small, **ArrayofLists** is more or less like an **ArrayList**. By carefully selecting the size of sublists, this new data structure takes advantages of both **LinkedList** and **ArrayList**, and remedies the disadvantages of both.

You are requested to implement the following methods:

- **clear()**
- **int size()**
- **boolean isEmpty()**
- **AnyType get(int idx)**: Return the value of the item of idx.
- **set(int idx, AnyType newVal)**: Replace the value of the item of idx by value newVal
- **add(AnyType x)**: add x as the last item in the entire list. If a new sublist is needed, create this sublist and update the lastIndex field of **ArrayofLists**.
- **add(int idx, AnyType x)**: add x at position idx of the list. If the current sublist is beyond its limit, remove the last item of the current sublist and insert it as the first item in the sublist to the right of this sublist. Repeat this process until a sublist has the room for a new item, or a new small list must be created to accommodate an item. Update the lastIndex fields of **ArrayofLists**.
- **AnyType remove(int idx)**: remove the item of index idx. If the sublist contained this item is empty, then shift all sublists to the right of this sublist one position to the left in the **ArrayofLists**.
- **boolean hasNext()**
- **AnyType next()**
• printList(Iterator<AnyType> itr): print out the items in the entire list (all sublists).

There are two major components, ArrayList and LinkedList in your assignment. You can either utilize the two corresponding classes in the Java library, or use the MyArrayList and MyLinkedList given in the textbook. The central part of this assignment is to integrate the two components into one data structure.

Running your program:

Let the size of sublist be 4. You are requested to test your program twice.

First, use your ArrayofLists constructor to construct a List of 20 integers, with values 0,1,2,...,19. Then read a sequence of commands from an input file named INPUT1 (will be provided). There are 20 lines in this file, each line being a command. A command consists of 1, 2 or 3 fields, with the first field being a character. There are 6 types of commands: a, r, g, s, p, q:

(1) a i v
   Call add(i,v) to add value v as the new item with index i. If the total number of items in the List is fewer than i+1, then do not add and print an error message.

(2) r i
   Remove the item value with index i. If the total number of items in the List is fewer than i+1, then do not add and print an error message.

(3) g i
   Call get(i) to get and print the item value with index i. If the total number of items in the List is fewer than i+1, then do not add and print an error message.

(4) s i v
   Call set(i,v) to set item value with index i by new value v. If the total number of items in the List is fewer than i+1, then do not add and print an error message.

(5) p
   Call print to print all item values in the current List iteratively (not recursively) by invoking hasNext and next, such as “The values in the current List arr: x, y, z, ....

(6) q
   Call add(x) to add a new item of value x as the last item of the List.

Input File Example:

a 10 3
r 5
a 15 25
Then, use your `ArrayLists` constructor to construct a List of 20 `floats`, with values 0.5, 1.5, 2.5, · · · 19.5. Then read a sequence of commands from an input file named INPUT2 (will be provided). There are 20 lines in this file, each line being a command.

**Input File Example:**

```
a 10 3.5
s 8 1.5
r 5
a 15 25.5
a 10 32.5
g 11
r 1
r 1
r 2
r 3
p
```