Data Structure 2018
Lab 03

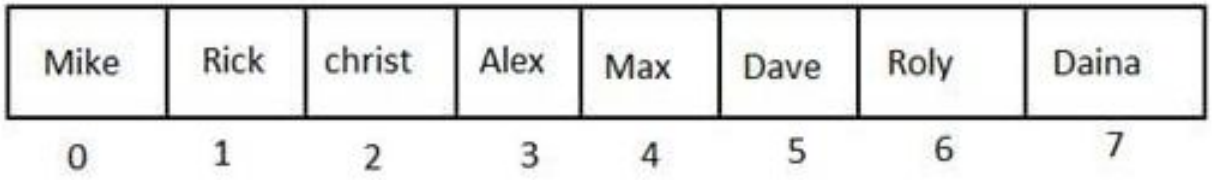
Mohammad Sadegh Najafi- Jonghyun Lee

March 22, 2019

Linear List

- A linear list is a list of finite numbers of elements stored in the memory.
- For representation, use a one-dimensional array element[]
- Each element of the linear list is referred by an index set.
- We can access all these elements with the help of the index set.

Linear List Example



Linear List operations

1. **isEmpty() -** Returns true if the list is empty, false otherwise.
2. **size() -** Gives the number of element in the list.
3. **get(theIndex) -** Gives the element with a given index.
4. **indexOf(theElement) -** Determines the index of a given element.
5. **remove(theIndex) -** Removes element with a given index and returns it.
6. **add(theIndex,theElement) -** Adds a given element so that new element has a specified index.

Linear List Interface

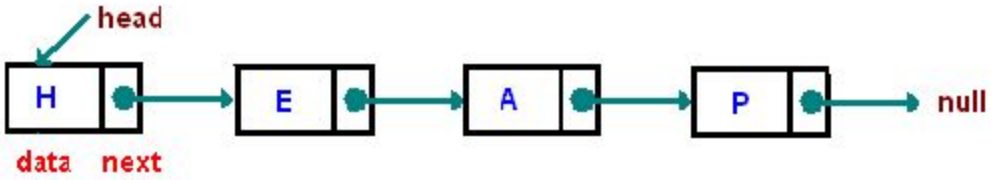
```
public interface LinearList {
    public boolean isEmpty();
    public int size();
    public Object get(int index);
    public int indexOf(Object elem);
    public Object remove(int index);
    public void add(int index, Object obj);
}
```

Iterators

The java.util.Iterator interface provides the following methods:

- **boolean hasNext()** - Returns true if the iteration has more elements.
- **E next()** - Returns the next element in the iteration.
- **void remove()** - Removes from the underlying collection the last element returned by the iterator (optional operation).

Linked Lists

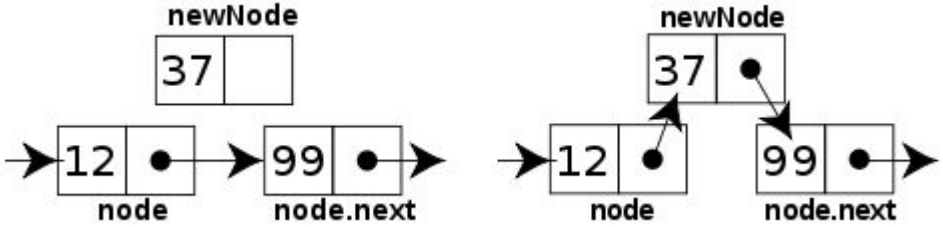


LinkedList Interface

- public void add(int v);
  - creates new Node object which holds value v and inserts into place where list will satisfy to remain sorted
- public boolean remove(int v);
  - removes Node object that holds value v. If list doesn't have object that holds value v, then returns false, otherwise true
- public void print();
  - prints each element in LinkedList
- public boolean isEmpty();
  - returns boolean of either is empty or not
- public int size();
  - returns size of list
- public int indexOf(int v);
  - returns index of Node that holds value v. If there is non, return -1

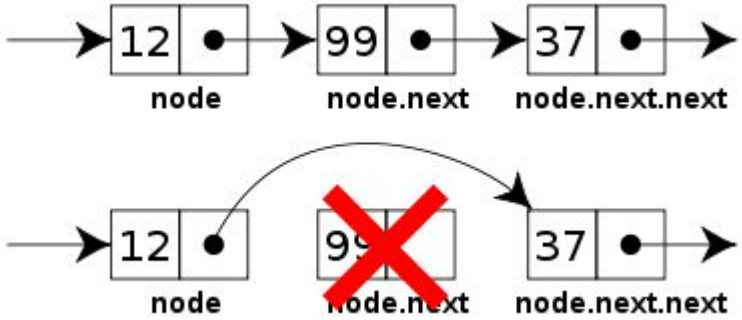
Linked List - add(v)

- if(head == null)
  - Create new node and set head
- if(v < head's v)
  - Create new node
  - Set new node's next to head
  - Set head to new node
- while(iterator's next != null)
  - if(v >= iterator's value && v <= iterator's next node's value) ?
- else?



Linked List - remove(v)

- if(head == null)
- if(v == head's v)
- while(iterator's next != null)
  - if(iterator's next's value == v)
- if non?



Write a sorted LinkedList

- Write a class MyLinkedList that implements all the LinkedList interface.

Expected output from Lab03.java

isEmpty? true
size is 0
False //(remove 12)
Index of 0: -1
2 4 4 5 7 10 //Add
size is 6
Index of 2: 0
Index of 7: 4
Index of 10: 5
True //(remove 4, and remove 2)
False //(remove 2)
1 4 5 10 //print list
is Empty? false
size is 4