List Interface

- A list can be defined as an ordered collection
- In Java, several classes can be implemented using List interface
  - ArrayList, LinkedList, Vector
- Most programming languages provide constructs for creating and managing lists
  - Array lists
- Array’s are static lists
  - size is fixed at compile time
  - Can resize, but requires effort
Fundamentals of a Dynamic List

- Many applications require data structures that can be changed easily
  - For example, inserting an element to a static array is painful, in fact it is $O(n)$

- So think of a data structure that supports inserts and deletions in constant time or $O(1)$
Linked Lists

- Linked Lists can be built dynamically
- Basic building of a dynamic linked list is a node
- A linked list is a collection of nodes, each having a “reference” to the next node
Connecting Nodes

- A collection of nodes is a list

- A collection of these nodes connected to each other is called a Linked List
Each node holds a reference (or address) of the next node

Address: FFAAOO

7/2/2011
Head of the List

- Head of the list is the “entry” point to the list

- Head of a list is a reference to first node
Linked List Nodes

- A linked list node object can be generated from a class
- Minimally a node contains two things
  - Data object
  - A reference
Implementation
// Class node
public class Node {
    public Comparable data;
    public Node next;
    
    ......
}

public class MyLinkedList implements List {
    Node head;  
    ........  
    ........
}
Homework (not graded)

- Write the following methods of MyLinkedList Class
  - public void reverse(MyLinkedList L);
  - public MyLinkedList findDups(MyLinkedList L);
    - Return a list that contains the duplicate elements (one each) of the original list (do not change the original)
  - public void makeCircular(MyLinkedList L);
    - Given a singly LL , make it circular
  - public void print(MyLinkedList L);
    - Prints the LL sequentially

- Tomorrow – Types of LL’s and their applications