Lab#07

Implementation of singly linked list

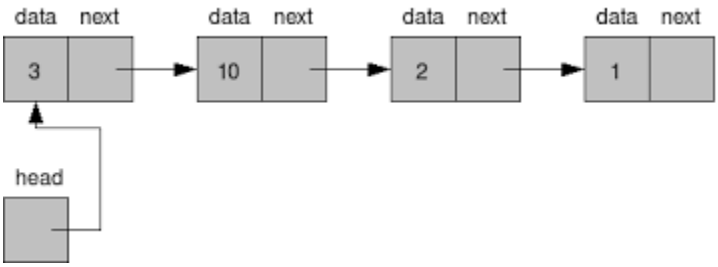
Objective

- To understand the basic concepts of the Implementation of Linked List and its operations.

Theory

Linked List

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. A linked list is a linear data structure where each element is a separate object. Each element is called a node of a list and is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty, then the head is a null reference. A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can grow and shrink on demand. Any application which must deal with an unknown number of objects will need to use a linked list.



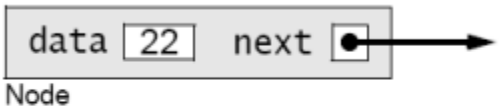


Figure 3.9 A Node object

Happy coding 😊

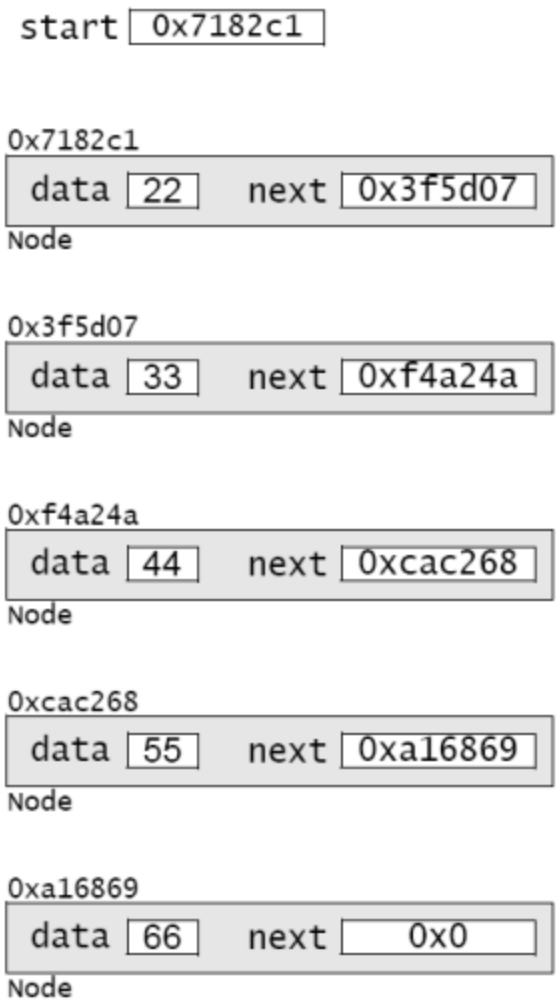
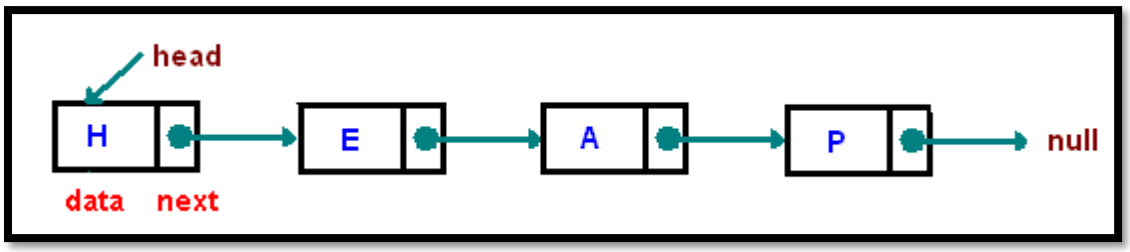


Figure 3.17 The five Node objects

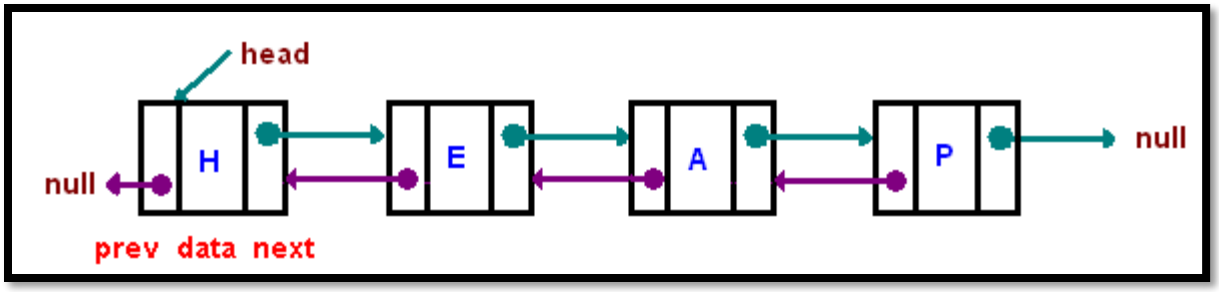
Types of Linked Lists

- **Singly linked list**: It is described above

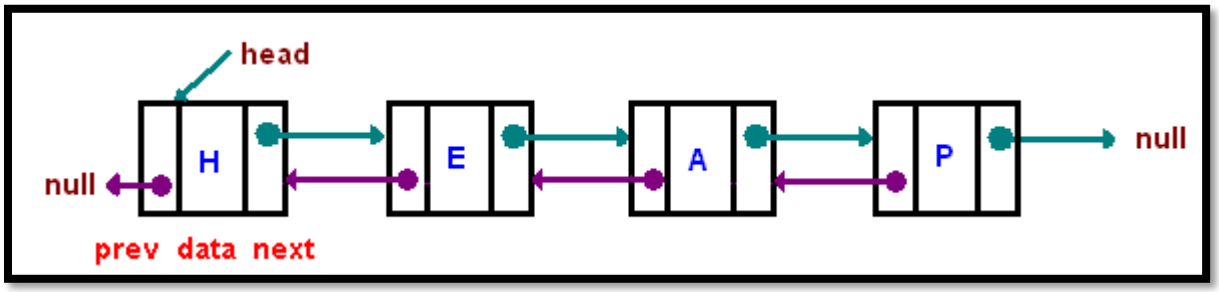


- **Doubly linked list**: It is a list that has two references, one to the next node and another to previous node.

Happy coding 😊



- **Circular linked list**: In this type of the last node of the list points back to the first node (or the head/root) of the list.



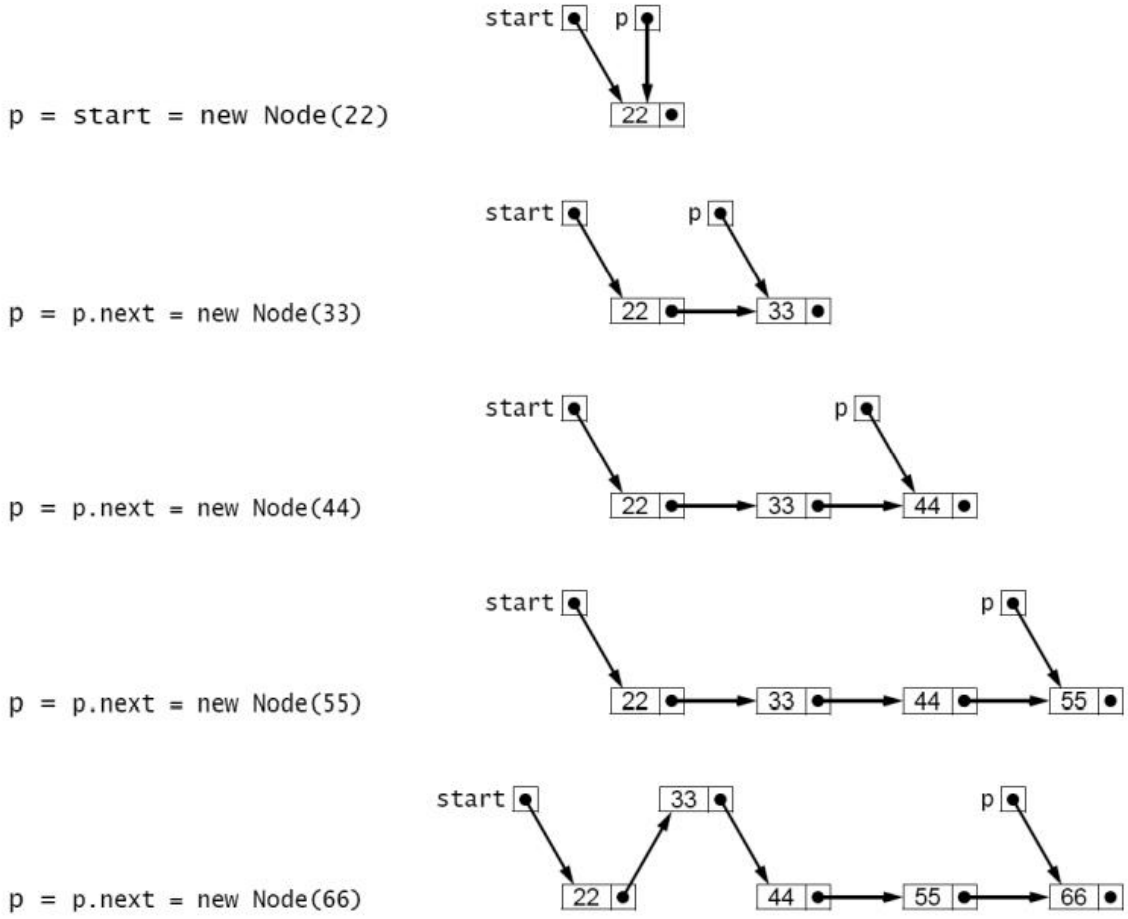


Figure 3.16 Trace of Example 3.6

Program Example # 01

```
class LinkedList{
        Node head;
        LinkedList(){
        this.head=null;
        }
}
```

```
class Node{
        int data;
        Node next;
```

Happy coding 😊

```
        Node(int d){
        data=d;
        next=null;
        }
}
```

```
class Main{
        public static void main(String[] args) {
                LinkedList l1 = new LinkedList();
                Node n1 = new Node (20);
                Node n2 = new Node (100);
                l1.head = n1;
                n1.next = n2;
                Node temp = l1.head;
                while(temp != null){
                        System.out.println(temp.data);
                        temp = temp.next;
                }
        }
}
```

Program Example # 02

Node class

```
public class Node
{

    int value;
    Node nextNode;

    public Node()
    {
        value = 0;
        nextNode = null;
    }

    public Node(int value)
    {
        this.value = value;
    }
```

LinkedList class

```
public class MyLinkedList
{

    Node root;  int size;

    public MyLinkedList()
    {
        root = null;
        size = 0;
    }

}
```

Linked list operations

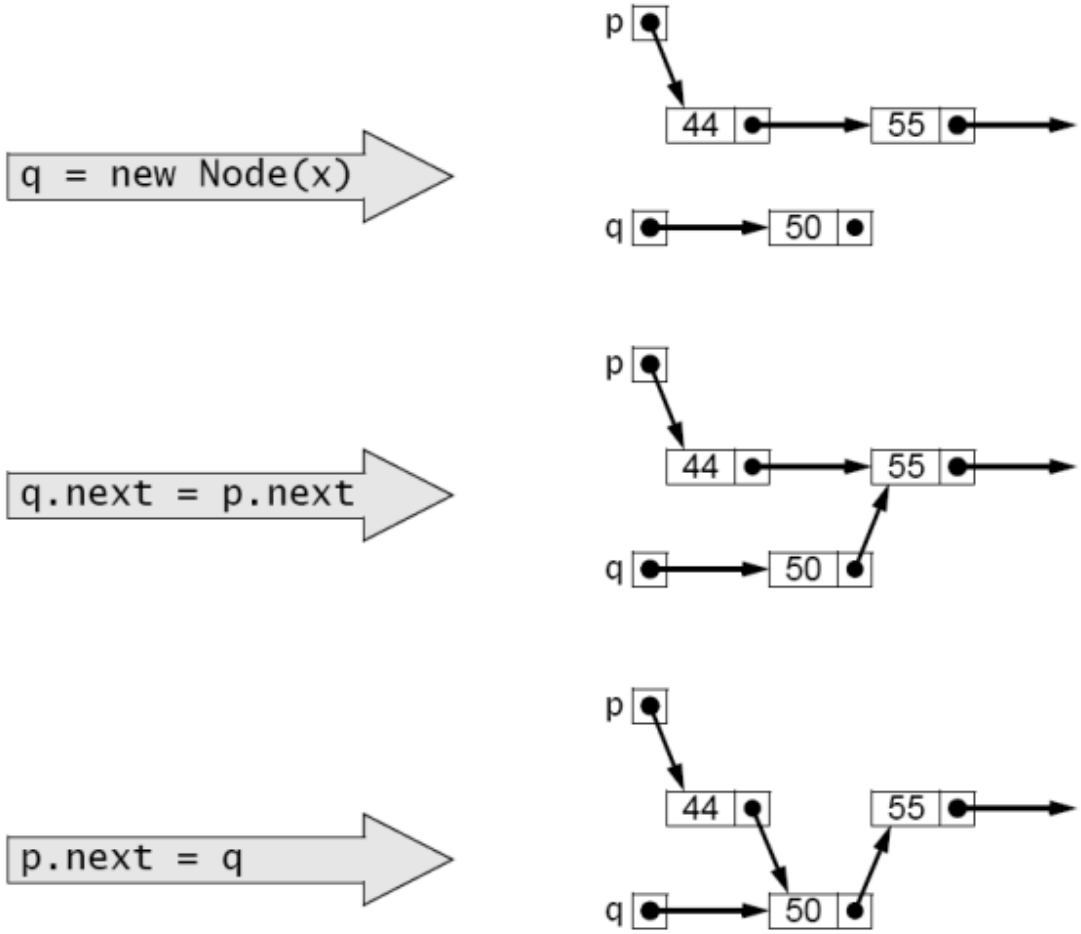


Figure 3.20 Inserting the new node in three steps

1) insertAtFirst

```
public void insertAtStart(int value)
{
    Node n;
    n = new Node();
    n.value = value;
    n.nextNode= root;
    root = n;
    size++;

}
```

2) insertAtLast

3) insertAtAnyPos

Happy coding 😊

```
public void insertAt(int value, int pos)
{
    if (pos == 1)
    {
        insertAtStart(value);
    }
    else if(pos == size + 1)
    {
        insertAtLast(value);

    }
    else if (pos > 1 && pos <=size)
    {
        Node n,t;
        n = new Node(value, null);
        t = root;
        for (int i = 1; i < pos - 1 ; i++)
        {
            t = t.nextNode;
            n.nextNode = t.nextNode;
            t.nextNode = n;

            size++;
        }

    }
    else
    {
        System.out.println("Not Possible ");

    }

}
```

4) Traversing

Start with the head/root and access each node until you reach null. Do not change the head/root reference.

Happy coding 😊

```
public void displayList()
{
    Node n;

    if (isEmpty())
    {
        System.out.println("empty list..");
    }

    else
    {
        n = root;
        for (int i = 1; i <=size; i++)
        {
           System.out.println(" " + n.value);
           n = n.nextNode;
        }
    }

}
```

Main

```
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);

    MyLinkedList m = new MyLinkedList();
    boolean flag = true;

  while(flag == true)
  {
      System.out.println("\t\t\t ----- LINKED LIST OPERATIONS -----");
      System.out.println("1. Add at Start");
      System.out.println("2. Add at any position");
      System.out.println("3. Display List");
      System.out.println("4. Exit");

      System.out.println("\nEnter choice: ");
      int choice = sc.nextInt();
      int position ,value;
```

```
        switch(choice)
        {
            case 1:
              System.out.println("Enter value to insert: ");
              value = sc.nextInt();
              m.insertAtStart(value);
              break;

            case 2:
              System.out.println("Enter position: ");
              position = sc.nextInt();
              System.out.println("Enter value to insert: ");
              value = sc.nextInt();
              m.insertAt(value, position);
              break;

            case 3:
             m.displayList();
             break;

          case 4:
           flag =false;
           break;

          default:
           System.out.println("Invalid choice");
           break;
    }

}
```

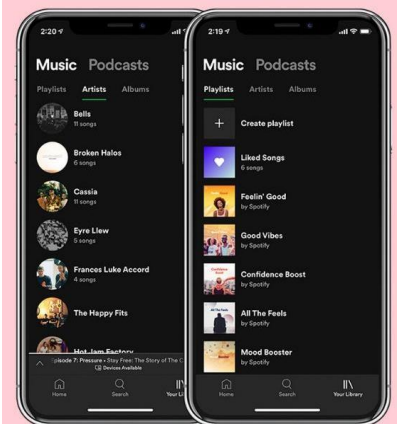
Exercise

INSTRUCTIONS:

NOTE: Violation of any of the following instructions may lead to the cancellation of your submission.

1. Create a word file name it by your student rollNo(NOTE format is given on teams use that format)
2. Paste all the code along with the out-output screenshot and for each question with the names such as Q1.java, Q2.java
3. Submit the word/pdf file on teams.

QUESTION#1 (Concrete class)



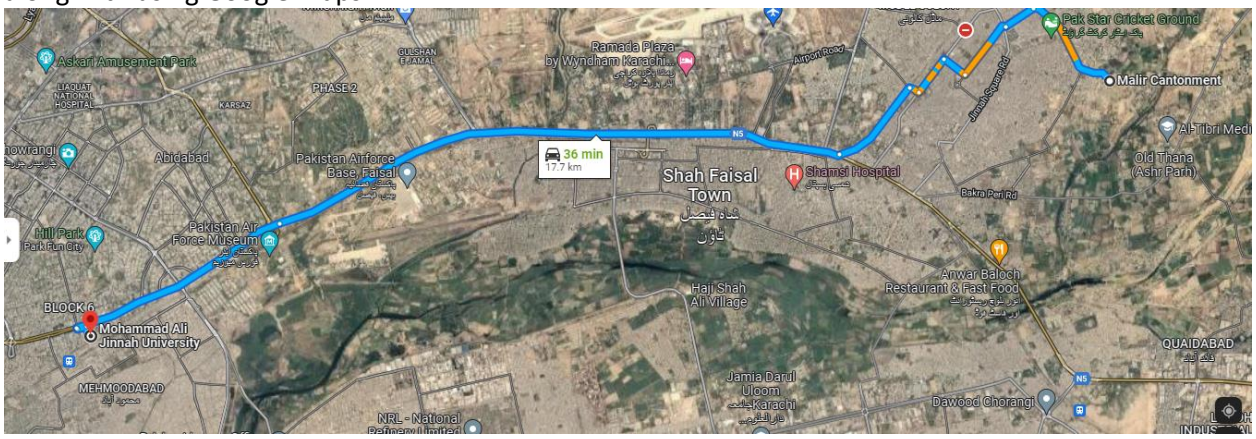
The Music app name is FireAir(class) music app which can perform following functions

- print all the liked list of songs/music names one by one.
- Add new songs/music (insertion at start, end, middle)
- Deletes a song/music using the number (Deletion at start, end, middle)
- Searches song/music.

QUESTION#2 (Concrete class)

Making The Trip to Mehran UET

Daily you would very often make the long trip out to Mehran UET. This route required a very specific sequence of buses, trains, and subways to get to the destination which you would follow along with using Google Maps.



Happy coding 😊

When you type in my origin, and destination, Google uses a complex algorithm to map out all of my possible routes and returns the top options in terms of minimizing the time and effort to get to your direction. At the end of the day, however, all it's doing is returning a Linked List to your phone. Now create this process.

QUESTION#03(generic class)

Create a generic class and implements the following operations:

Task # 1: Implement the isEmpty() method in the Linked List

Task # 2: Implement the getSize() method in the Linked List

Task # 3: Implement the insertAtLast() method in the Linked List

Task # 4: Implement the insertAtLast() method in the Linked List

Task # 5: Implement the insertAtPosition() method in the Linked List

Task # 6: Implement the deleteFirst() method in the Linked List

Task # 7: Implement the deleteLast() method in the Linked List

Task # 8: Implement the deleteAtPosition() method in the Linked List

Task # 9: Implement the search() method in the Linked List to check whether the element exists in the list or not.

Task # 10: delete element by value; implement deleteValue(int value)

Task # 11: display method

Happy coding 😊