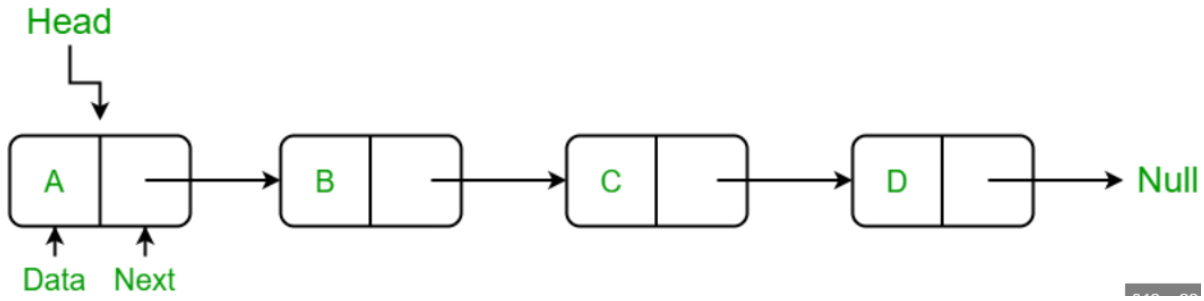
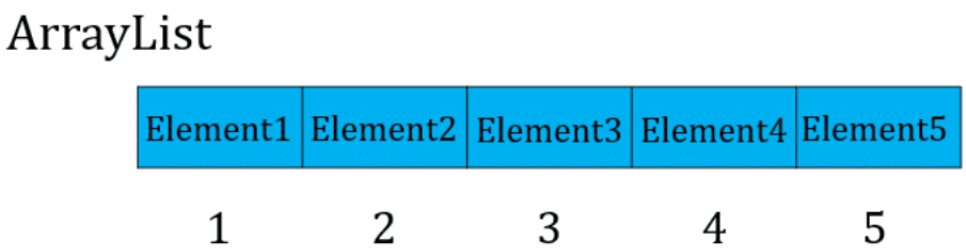
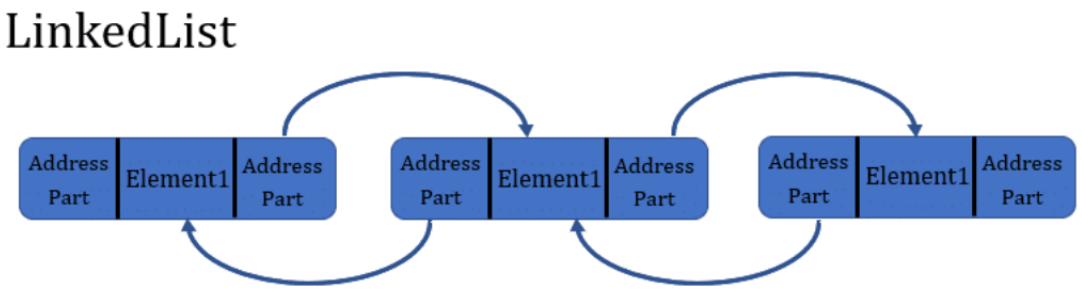
Introduction to Linked List
Java

Linked List

Linked List is a part of the Collection framework present in java.util package. This class is an implementation of the LinkedList data structure which is a linear data structure where the elements are not stored in contiguous locations and every element is a separate object with a data part and address part. The elements are linked using pointers and addresses and each element is known as a node.







For manipulation (insertion) – linkedlist
O(n)>O(1)
For search – arraylist
O(1)>O(n)

LinkedList class Implementation (Collection Framework)

Operations:

1. **Declare an LinkedList of different Types**

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
        LinkedList <String>list1 = new LinkedList<String>();
        LinkedList<Integer>list2 = new LinkedList<Integer>();
        LinkedList<Float>list3 = new LinkedList<Float>();
        LinkedList<Boolean>list4 = new LinkedList<Boolean>();

    }
}
```

Add element

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
        LinkedList <String>ll = new LinkedList<String>();

        //add in linklist
        ll.add("name");
        ll.add("is");
        System.out.println(ll);

        //add in first place
        ll.addFirst("my");
        System.out.println(ll);

        //add in last place
        ll.addLast("piyush");
        System.out.println(ll);

    }
}
```

Get Element

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
```

```
        LinkedList <String>ll = new LinkedList<String>();
        ll.add("my");
        ll.add("name");
        ll.add("is");
        ll.add("piyush");
        System.out.println(ll);
        //get element
        System.out.println(ll.get(2));

        //get first element
        System.out.println(ll.getFirst());

        //get last element
        System.out.println(ll.getLast());
    }
}
```

Set Element

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
        LinkedList <String>ll = new LinkedList<String>();
        ll.add("my");
        ll.add("name");
        ll.add("is");
        ll.add("piyush");
        System.out.println(ll);
        //set element
        ll.set(0, "your");
        System.out.println(ll);
        }
}
```

Delete Element

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
        LinkedList <String>ll = new LinkedList<String>();
        ll.add("my");
        ll.add("name");
        ll.add("is");
        ll.add("piyush");
```

```
        System.out.println(ll);

        //delete head element
        ll.remove();
        System.out.println(ll);

        //delete specific element
        ll.remove(1);
        System.out.println(ll);

        //delete last element
        ll.removeLast();
        System.out.println(ll);
        }
}
```

Size of the linklist

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
        LinkedList <String>ll = new LinkedList<String>();

        //add in linklist
        ll.add("name");
        ll.add("is");
        ll.add("piyush");
        System.out.println(ll);

        //add in first place
        ll.addFirst("my");
        System.out.println(ll);

        //size of the linklist
        System.out.println(ll.size());

    }
}
```

Iteration of linkedlist

```
package myPackage;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
```

```
        LinkedList <String>ll = new LinkedList<String>();

        //add in linklist
        ll.add("name");
        ll.add("is");
        ll.add("piyush");
        System.out.println(ll);

        //add in first place
        ll.addFirst("my");
        System.out.println(ll);

        //size of the linklist
        System.out.println(ll.size());

        //iteration of linklist
        for(int i=0;i<ll.size();i++){
            System.out.print(ll.get(i) + " -> ");
        }
        System.out.println("null");
    }
}
```

Reverse Linkedlist

Input: *head: 1 -> 2 -> 3 -> 4 -> NULL*
Output: *head: 4 -> 3 -> 2 -> 1 -> NULL*

```
package myPackage;
import java.util.Collections;
import java.util.LinkedList;
public class LL {
    public static void main(String[] args) {
        LinkedList <Integer>ll = new LinkedList<Integer>();
        ll.add(1);
        ll.add(2);
        ll.add(3);
        ll.add(4);
        System.out.println(ll);
        Collections.reverse(ll);
        System.out.println(ll);
        }
}
```

Using Iterative Method

```java
// Iterative Java program to reverse a linked list
package myPackage;

class Node {
    int data;
    Node next;

    Node(int new_data) {
        data = new_data;
        next = null;
    }
}

// Given the head of a list, reverse the list and return the
// head of reversed list
public class LL {
     static Node reverseList(Node head) {

        // Initialize three pointers: curr, prev and next
        Node curr = head, prev = null, next;

        // Traverse all the nodes of Linked List
        while (curr != null) {

            // Store next
            next = curr.next;

            // Reverse current node's next pointer
            curr.next = prev;

            // Move pointers one position ahead
            prev = curr;
            curr = next;
        }

        // Return the head of reversed linked list
        return prev;
    }

    // This function prints the contents
    // of the linked list starting from the head
    static void printList(Node node) {
        while (node != null) {
            System.out.print(" " + node.data);
            node = node.next;
        }
    }
```

```
    public static void main(String[] args) {

        // Create a hard-coded linked list:
        // 1 -> 2 -> 3 -> 4 -> 5
        Node head = new Node(1);
        head.next = new Node(2);
        head.next.next = new Node(3);
        head.next.next.next = new Node(4);
        head.next.next.next.next = new Node(5);

        System.out.print("Given Linked list:");
        printList(head);

        head = reverseList(head);

        System.out.print("\nReversed Linked List:");
        printList(head);
    }
}
```

Homework Problems

1. Make a Linked List & add the following elements to it : (1, 5, 7, 3 , 8, 2, 3). Search for the number 7 & display its index.
2. Take elements(numbers in the range of 1-50) of a Linked List as input from the user. Delete all nodes which have values greater than 25.

Stack:

Stack class is based on the basic principle of LIFO(last-in-first-out).

The Stack class extends Vector and provides additional functionality specifically for stack operations, such as push, pop, peek, empty, and search.

The Stack class be referred to as a subclass of Vector, inheriting its methods and properties.

```
package myPackage;

// Java Program Implementing Stack Class
import java.util.Stack;

public class Stacks
{
    public static void main(String[] args)
    {
        // Create a new stack
```

```
        Stack<Integer> s = new Stack<>();

        // Push elements onto the stack
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

        // Pop elements from the stack
        while(!s.isEmpty()) {
            System.out.println(s.pop());
        }
    }
}
```

Adding Elements: push()

```
package myPackage;

// Java Program Implementing Stack Class
import java.util.Stack;

public class Stacks
{
    public static void main(String[] args)
    {
        // Create a new stack
        Stack<Integer> s = new Stack<>();

        // Push elements onto the stack
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

        System.out.println(s);
        }
    }
```

Accessing the Element: peek()

```
package myPackage;

// Java Program Implementing Stack Class
import java.util.Stack;
```

```
public class Stacks
{
    public static void main(String[] args)
    {
        // Create a new stack
        Stack<Integer> s = new Stack<>();

        // Push elements onto the stack
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

        System.out.println(s);
       System.out.println(s.peek());

        }
    }
```

Removing Elements: pop()

```
package myPackage;

// Java Program Implementing Stack Class
import java.util.Stack;

public class Stacks
{
    public static void main(String[] args)
    {
        // Create a new stack
        Stack<Integer> s = new Stack<>();

        // Push elements onto the stack
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

        System.out.println(s);
        s.pop();
       System.out.println(s);

        }
    }
```

Search Element: search ()

```
package myPackage;

// Java Program Implementing Stack Class
import java.util.Stack;

public class Stacks
{
    public static void main(String[] args)
    {
        // Create a new stack
        Stack<Integer> s = new Stack<>();

        // Push elements onto the stack
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

        System.out.println(s);
       System.out.println(s.search(3));

        }
    }
```

Empty()

```
package myPackage;

// Java Program Implementing Stack Class
import java.util.Stack;

public class Stacks
{
    public static void main(String[] args)
    {
        // Create a new stack
        Stack<Integer> s = new Stack<>();

        // Push elements onto the stack
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

       System.out.println(s);
       System.out.println(s.empty());
```

```
        s.pop();
        s.pop();
        s.pop();
        s.pop();
        System.out.println(s.empty());


         }
    }
```