**Practical 8**

**Name: Shantanu Sethi**

**Roll no. 163**

**Aim: To implement linear linked list**

**Objectives:**

1. Learn how to implement different operations on linear linked list
   * Insert node: from front and from last
   * Delete node
   * Search list
   * Display
   * Count

**Theory:**

**Structure of node**

|  |  |
| --- | --- |
| **INFO** | **LINK** |

**Algorithm :**

**Insert from front:**

1. [initialize field of new node]

INFO(Node1)= item

2. [ check Linked list is empty]

if(FIRST==NULL)

LINK(Node1)= NULL

FIRST=Node1

else

LINK(Node1)=FIRST

FIRST= Node1

3. [Return address of new node]

return [ Node1]

**Insert node form end:**

1.[initialize field of new node]

INFO(Node1)= item

LINK(Node1)= NULL

2. [ check Linked list is empty]

if(FIRST==NULL)

FIRST=Node1

3.[Initialize search for the last node]

move= FIRST

4.[ search for end of list]

repeat while(LINK(move)!= NULL)

move=LINK(move)

5. [Set the link field of last node to Node1]

LINK(move)= Node1

6 .[Return First node pointer]

Return(FIRST)

**Delete node from front:**

1.[ empty list ]

If FIRST = NULL

then write (“ underflow”)

return

2.[ Delete first node and reinitialize first ]

MOVE=FIRST

write (deleted element is INFO(MOVE))

MOVE= LINK( MOVE)

FIRST=MOVE

Return FIRST

**Delete from last node:**

1.[ empty list ]

If (FIRST = NULL)

then write (“ underflow”)

return

2.[ Initialize first to save ]

Move = FIRST

3.[Find last node]

Repeat through step 5 while LINK (SAVE) ≠ NULL

3.1 [ Update predecessor Marker ]

PRED = Move

3.2 Move = LINK (Move )

4 .[ Deleted element is]

INFO(Move)

5. [ Delete Last Node ]

LINK [PRED ]= NULL

**Searching node in Link list:**

1.[ empty list ]

If FIRST = NULL

then write (“ underflow”)

return

2.[ Initialize Move to first node of Link list and count to 1 ]

Move= FIRST

C=1

3.[ count number of node by visiting every node]

Repeat while LINK (SAVE) ≠ NULL

begin

C = C +1

If (INFO(Move)=item)

begin

F= 1

return

End

Move= LINK (Move )

end

4.[ return position if element is found ]

if( f=1)

Write (“ element found”)

return ( C)

else

Write (“ element not found”)

return

**Display node in LL:**

1.[ empty list ]

If FIRST = NULL

then write (“ underflow”)

return

2.[ Initialize Move to first node of Link list ]

Move= FIRST

3.[ Print info of every node]

Repeat while (Move) ≠ NULL

begin

write( INFO ( Move))

Move= LINK (Move )

end

**Count Node in LL:**

1.[ empty list ]

If FIRST = NULL

then write (“ underflow”)

return

2.[ Initialize Move to first node of Link list and count to 1 ]

Move= FIRST

c=1

3.[ count number of node by visiting every node]

Repeat while LINK (Move) ≠ NULL

begin

c = c +1

Move=LINK (Move )

end

4.[ return counter ]

Return ( C)

**Program:**

1. **Insert node:** from front and from last

**Code:**

package linkedlist;

public class insert {

public static void main(String[] args) {

System.out.println("Abhinav Singh --184");

class Node {

int data;

Node next;

Node(int item) {

this.data = item;

this.next = null;

}

}

Node first = null;

Node newNode = new Node(10);

if (first == null) {

first = newNode;

} else {

newNode.next = first;

first = newNode;

}

System.out.println("Inserted at Front: 10");

System.out.print("Linked List: ");

Node temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

}

System.out.println("NULL");

newNode = new Node(20);

if (first == null) {

first = newNode;

} else {

Node move = first;

while (move.next != null) {

move = move.next;

}

move.next = newNode;

}

System.out.println("Inserted at End: 20");

System.out.print("Linked List: ");

temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

}

System.out.println("NULL");

newNode = new Node(5);

newNode.next = first;

first = newNode;

System.out.println("Inserted at Front: 5");

System.out.print("Linked List: ");

temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

}

System.out.println("NULL");

newNode = new Node(30);

Node move = first;

while (move.next != null) {

move = move.next;

}

move.next = newNode;

System.out.println("Inserted at End: 30");

System.out.print("Linked List: ");

temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

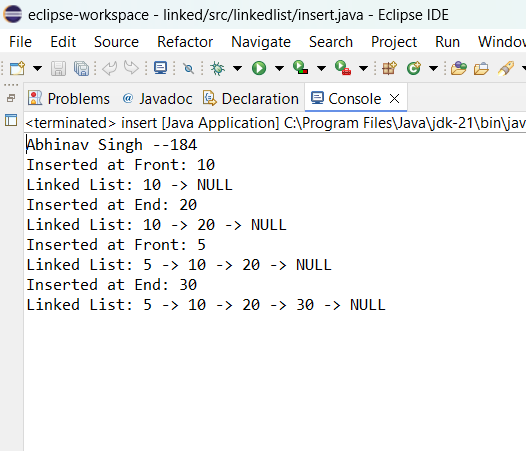
}

System.out.println("NULL");

}

}

**Output:**

****

1. **Delete node**

**Code:**

package linkedlist;

public class Delete {

public static void main(String[] args) {

System.out.println("Abhinav Singh --184");

class Node {

int data;

Node next;

Node(int item) {

this.data = item;

this.next = null;

}

}

Node first = null;

Node newNode = new Node(10);

first = newNode;

newNode = new Node(20);

first.next = newNode;

newNode = new Node(30);

first.next.next = newNode;

System.out.print("Initial Linked List: ");

Node temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

}

System.out.println("NULL");

if (first == null) {

System.out.println("Underflow - List is empty!");

} else {

Node move = first;

System.out.println("Deleted from Front: " + move.data);

first = first.next;

}

System.out.print("Linked List after Deletion from Front: ");

temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

}

System.out.println("NULL");

if (first == null) {

System.out.println("Underflow - List is empty!");

} else if (first.next == null) {

System.out.println("Deleted from End: " + first.data);

first = null;

} else {

Node move = first;

Node pred = null;

while (move.next != null) {

pred = move;

move = move.next;

}

System.out.println("Deleted from End: " + move.data);

pred.next = null;

}

System.out.print("Linked List after Deletion from End: ");

temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

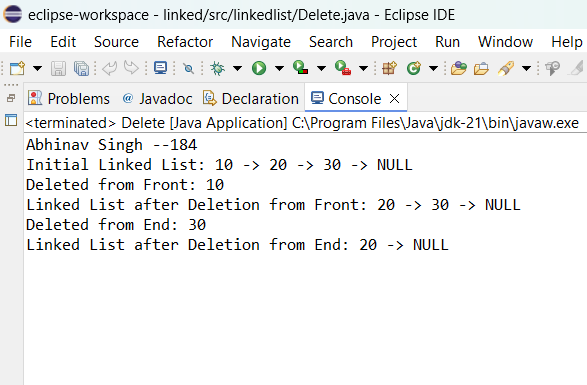
}

System.out.println("NULL");

}

}

**Output:**

****

1. **Search list**

**Code:**

package linkedlist;

public class Search {

public static void main(String[] args) {

System.out.println("Abhinav Singh --184");

class Node {

int data;

Node next;

Node(int item) {

this.data = item;

this.next = null;

}

}

Node first = null;

Node newNode = new Node(10);

first = newNode;

newNode = new Node(20);

first.next = newNode;

newNode = new Node(30);

first.next.next = newNode;

System.out.print("Linked List: ");

Node temp = first;

while (temp != null) {

System.out.print(temp.data + " -> ");

temp = temp.next;

}

System.out.println("NULL");

int item = 20;

if (first == null) {

System.out.println("Underflow - List is empty!");

} else {

Node move = first;

int count = 1;

boolean found = false;

while (move != null) {

if (move.data == item) {

found = true;

break;

}

move = move.next;

count++;

}

if (found) {

System.out.println("Element " + item + " found at position: " + count);

} else {

System.out.println("Element " + item + " not found in the list.");

}

}

item = 40;

if (first == null) {

System.out.println("Underflow - List is empty!");

} else {

Node move = first;

int count = 1;

boolean found = false;

while (move != null) {

if (move.data == item) {

found = true;

break;

}

move = move.next;

count++;

}

if (found) {

System.out.println("Element " + item + " found at position: " + count);

} else {

System.out.println("Element " + item + " not found in the list.");

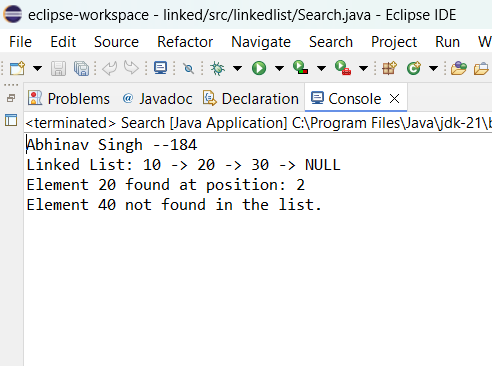
}

}

}

}

**Output:**

****

1. **Display**

**Code:**

package linkedlist;

import java.util.Scanner;

public class Display {

public static void main(String[] args) {

System.out.println("Abhinav Singh --184");

class Node {

int data;

Node next;

Node(int item) {

this.data = item;

this.next = null;

}

}

Node first = null;

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of nodes to insert: ");

int n = scanner.nextInt();

Node last = null;

for (int i = 1; i <= n; i++) {

System.out.print("Enter data for node " + i + ": ");

int data = scanner.nextInt();

Node newNode = new Node(data);

if (first == null) {

first = newNode; t

last = newNode;

} else {

last.next = newNode;

last = newNode;

}

}

System.out.print("Linked List: ");

if (first == null) {

System.out.println("Underflow - List is empty!");

} else {

Node move = first;

while (move != null) {

System.out.print(move.data + " -> ");

move = move.next;

}

System.out.println("NULL");

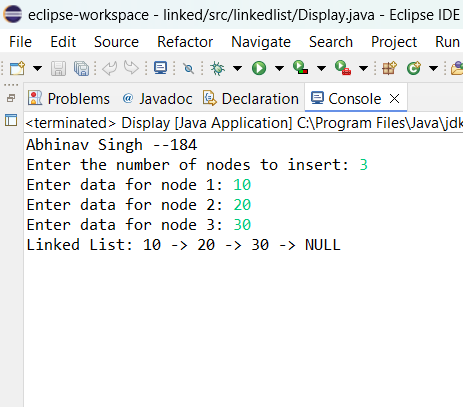
}

scanner.close();

}

}

**Output:**

****

1. **Count**

**Code:**

package linkedlist;

import java.util.Scanner;

public class Count {

public static void main(String[] args) {

System.out.println("Abhinav Singh --184");

class Node {

int data;

Node next;

Node(int item) {

this.data = item;

this.next = null;

}

}

Node first = null;

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of nodes to insert: ");

int n = scanner.nextInt();

Node last = null;

for (int i = 1; i <= n; i++) {

System.out.print("Enter data for node " + i + ": ");

int data = scanner.nextInt();

Node newNode = new Node(data);

if (first == null) {

first = newNode;

last = newNode;

} else {

last.next = newNode;

last = newNode;

}

}

if (first == null) {

System.out.println("Underflow - List is empty!");

} else {

Node move = first;

int count = 0;

while (move != null) {

count++;

move = move.next;

}

System.out.println("Total number of nodes: " + count);

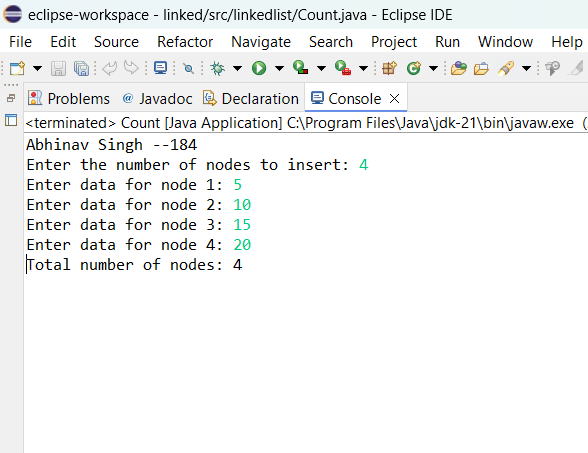
}

scanner.close();

}

}

**Output:**

****

**Conclusion:** Successfully implemented different operation on singly linked list