// Traverse a doubly linked list in forward and backward directions

public class DLL {

Node head;

class Node {

int data;

Node prev;

Node next;

Node(int d) {

data = d;

}

}

public void push(int new\_data) {

Node new\_Node = new Node(new\_data);

new\_Node.next = head;

new\_Node.prev = null;

if (head != null)

head.prev = new\_Node;

head = new\_Node;

}

public void InsertAfter(Node prev\_Node, int new\_data) {

if (prev\_Node == null) {

System.out.println("The given previous node cannot be NULL ");

return;

}

Node new\_node = new Node(new\_data);

new\_node.next = prev\_Node.next;

prev\_Node.next = new\_node;

new\_node.prev = prev\_Node;

if (new\_node.next != null)

new\_node.next.prev = new\_node;

}

void append(int new\_data) {

Node new\_node = new Node(new\_data);

Node last = head;

new\_node.next = null;

if (head == null) {

new\_node.prev = null;

head = new\_node;

return;

}

while (last.next != null)

last = last.next;

last.next = new\_node;

new\_node.prev = last;

}

public void printlist(Node node) {

Node last = null;

System.out.println("Traversal in forward Direction");

while (node != null) {

System.out.print(node.data + " ");

last = node;

node = node.next;

}

System.out.println();

System.out.println("Traversal in reverse direction");

while (last != null) {

System.out.print(last.data + " ");

last = last.prev;

}

}

public static void main(String[] args) {

DLL dll = new DLL();

dll.append(6);

dll.push(7);

dll.push(1);

dll.append(4);

dll.InsertAfter(dll.head.next, 8);

System.out.println("Created DLL is: ");

dll.printlist(dll.head);

}

}