1. public class DoublyLinkedList {
3. //Represent a node of the doubly linked list
5. class Node{
6. int data;
7. Node previous;
8. Node next;
10. public Node(int data) {
11. this.data = data;
12. }
13. }
15. //Represent the head and tail of the doubly linked list
16. Node head, tail = null;
18. //addNode() will add a node to the list
19. public void addNode(int data) {
20. //Create a new node
21. Node newNode = new Node(data);
23. //If list is empty
24. if(head == null) {
25. //Both head and tail will point to newNode
26. head = tail = newNode;
27. //head's previous will point to null
28. head.previous = null;
29. //tail's next will point to null, as it is the last node of the list
30. tail.next = null;
31. }
32. else {
33. //newNode will be added after tail such that tail's next will point to newNode
34. tail.next = newNode;
35. //newNode's previous will point to tail
36. newNode.previous = tail;
37. //newNode will become new tail
38. tail = newNode;
39. //As it is last node, tail's next will point to null
40. tail.next = null;
41. }
42. }
44. //display() will print out the nodes of the list
45. public void display() {
46. //Node current will point to head
47. Node current = head;
48. if(head == null) {
49. System.out.println("List is empty");
50. return;
51. }
52. System.out.println("Nodes of doubly linked list: ");
53. while(current != null) {
54. //Prints each node by incrementing the pointer.
56. System.out.print(current.data + " ");
57. current = current.next;
58. }
59. }
61. public static void main(String[] args) {
63. DoublyLinkedList dList = new DoublyLinkedList();
64. //Add nodes to the list
65. dList.addNode(1);
66. dList.addNode(2);
67. dList.addNode(3);
68. dList.addNode(4);
69. dList.addNode(5);
71. //Displays the nodes present in the list
72. dList.display();
73. }
74. }

|  |
| --- |
| class DoublyLinkedList {      //A node class for doubly linked list      class Node{          int item;          Node previous;          Node next;            public Node(int item) {              this.item = item;          }      }      //Initially, heade and tail is set to null      Node head, tail = null;        //add a node to the list      public void addNode(int item) {          //Create a new node          Node newNode = new Node(item);            //if list is empty, head and tail points to newNode          if(head == null) {              head = tail = newNode;              //head's previous will be null              head.previous = null;              //tail's next will be null              tail.next = null;          }          else {              //add newNode to the end of list. tail->next set to newNode              tail.next = newNode;              //newNode->previous set to tail              newNode.previous = tail;              //newNode becomes new tail              tail = newNode;              //tail's next point to null              tail.next = null;          }      }    //print all the nodes of doubly linked list      public void printNodes() {          //Node current will point to head          Node current = head;          if(head == null) {              System.out.println("Doubly linked list is empty");              return;          }          System.out.println("Nodes of doubly linked list: ");          while(current != null) {              //Print each node and then go to next.              System.out.print(current.item + " ");              current = current.next;          }      }  }  class Main{      public static void main(String[] args) {          //create a DoublyLinkedList object          DoublyLinkedList dl\_List = new DoublyLinkedList();          //Add nodes to the list          dl\_List.addNode(10);          dl\_List.addNode(20);          dl\_List.addNode(30);          dl\_List.addNode(40);          dl\_List.addNode(50);            //print the nodes of DoublyLinkedList          dl\_List.printNodes();      }  } |

**Output:**

Nodes of doubly linked list:  
10 20 30 40 50