1. **public** **class** MinMax {
2. //Represents the node of list.
3. **public** **class** Node{
4. **int** data;
5. Node next;
6. **public** Node(**int** data) {
7. **this**.data = data;
8. }
9. }
11. //Declaring head and tail pointer as null.
12. **public** Node head = **null**;
13. **public** Node tail = **null**;
15. //This function will add the new node at the end of the list.
16. **public** **void** add(**int** data){
17. //Create new node
18. Node newNode = **new** Node(data);
19. //Checks if the list is empty.
20. **if**(head == **null**) {
21. //If list is empty, both head and tail would point to new node.
22. head = newNode;
23. tail = newNode;
24. newNode.next = head;
25. }
26. **else** {
27. //tail will point to new node.
28. tail.next = newNode;
29. //New node will become new tail.
30. tail = newNode;
31. //Since, it is circular linked list tail will points to head.
32. tail.next = head;
33. }
34. }
36. //Finds out the minimum value node in the list
37. **public** **void** minNode() {
38. Node current = head;
39. //Initializing min to initial node data
40. **int** min = head.data;
41. **if**(head == **null**) {
42. System.out.println("List is empty");
43. }
44. **else** {
45. **do**{
46. //If current node's data is smaller than min
47. //Then replace value of min with current node's data
48. **if**(min > current.data) {
49. min = current.data;
50. }
51. current= current.next;
52. }**while**(current != head);
54. System.out.println("Minimum value node in the list: "+ min);
55. }
56. }
58. //Finds out the maximum value node in the list
59. **public** **void** maxNode() {
60. Node current = head;
61. //Initializing max to initial node data
62. **int** max = head.data;
63. **if**(head == **null**) {
64. System.out.println("List is empty");
65. }
66. **else** {
67. **do**{
68. //If current node's data is greater than max
69. //Then replace value of max with current node's data
70. **if**(max < current.data) {
71. max = current.data;
72. }
73. current= current.next;
74. }**while**(current != head);
76. System.out.println("Maximum value node in the list: "+ max);
77. }
78. }
80. **public** **static** **void** main(String[] args) {
81. MinMax cl = **new** MinMax();
82. //Adds data to the list
83. cl.add(5);
84. cl.add(20);
85. cl.add(10);
86. cl.add(1);
87. //Prints the minimum value node in the list
88. cl.minNode();
89. //Prints the maximum value node in the list
90. cl.maxNode();
91. }
92. }

