```
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

class CircularLinkedList {
    Node head;

    CircularLinkedList() {
        this.head = null;
    }

    void insert(int data) {
        Node newNode = new Node(data);

        if (head == null) { // If the list is empty
            head = newNode;
            head.next = head; // Point back to itself to make it circular
        } else if (data <= head.data) { // If the new node is smaller than or equal to the head
            newNode.next = head;
            Node current = head;

            while (current.next != head) { // Find the last node in the list
                current = current.next;
            }
```

```
            current.next = newNode; // Make the list circular again
            head = newNode; // Update the head
        } else { // If the new node should be inserted between two nodes or at the end
            Node current = head;

            while (current.next != head && current.next.data < data) { // Find the correct position
                current = current.next;
            }

            newNode.next = current.next; // Insert the new node
            current.next = newNode;
        }
    }

    void display() {
        if (head == null) {
            System.out.println("List is empty.");
            return;
        }

        Node current = head;

        do {
            System.out.print(current.data + " ");
            current = current.next;
        } while (current != head);

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

```
public class Main {
    public static void main(String[] args) {
        CircularLinkedList list = new CircularLinkedList();

        list.insert(2);
        list.insert(4);
        list.insert(6);
        list.insert(8);

        System.out.println("Original list:");
        list.display();

        list.insert(5); // Inserting a new element

        System.out.println("List after inserting 5:");
        list.display();
    }
}
```