linkedlist/LinkedList.java

```java
package linkedlist;

public class LinkedList {
    private Node head;
    private Node tail;
    private int length;

    class Node {
        int value;
        Node next;

        Node(int value) {
            this.value = value;
        }
    }

    public LinkedList(int value) {
        Node newNode = new Node(value);
        head = newNode;
        tail = newNode;
        length = 1;
    }

    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.println(temp.value);
            temp = temp.next;
        }
    }

    public void getHead() {
        System.out.println("Head: " + head.value);
    }

    public void getTail() {
        System.out.println("Tail: " + tail.value);
    }

    public void getLength() {
        System.out.println("Length: " + length);
    }

    public void append(int value) {
        Node newNode = new Node(value);
        if (length == 0) {
            head = newNode;
            tail = newNode;
        } else {
            tail.next = newNode;
            tail = newNode;
```

```
52          }
53          length++;
54      }
55
56      public Node removeLast() {
57          if (length == 0)
58              return null;
59          Node temp = head;
60          Node pre = head;
61          while (temp.next != null) {
62              pre = temp;
63              temp = temp.next;
64          }
65          tail = pre;
66          tail.next = null;
67          length--;
68          if (length == 0) {
69              head = null;
70              tail = null;
71          }
72          return temp;
73
74      }
75
76      public void prepend(int value) {
77          Node newNode = new Node(value);
78          if (length == 0) {
79              head = newNode;
80              tail = newNode;
81          } else {
82              newNode.next = head;
83              head = newNode;
84          }
85          length++;
86      }
87
88      public Node removeFirst() {
89          if (length == 0)
90              return null;
91          Node temp = head;
92          head = head.next;
93          temp.next = null;
94          length--;
95          if (length == 0) {
96              tail = null;
97          }
98          return temp;
99      }
100
101     public Node get(int index) {
102         if (index < 0 || index >= length) {
103             return null;
104         }
105         Node temp = head;
```

```
106         for (int i = 0; i < index; i++) {
107             temp = temp.next;
108         }
109         return temp;
110     }
111 
112     public boolean set(int index, int value) {
113         Node temp = get(index);
114         if (temp != null) {
115             temp.value = value;
116             return true;
117         }
118         return false;
119     }
120 
121     public boolean insert(int index, int value) {
122         if(index < 0 || index > length) return false;
123         if(index == 0) {
124             prepend(value);
125             return true;
126         }
127         if(index == length) {
128             append(value);
129             return true;
130         }
131         Node newNode = new Node(value);
132         Node temp = get(index - 1);
133         newNode.next = temp.next;
134         temp.next = newNode;
135         length++;
136         return true;
137     }
138 
139     public Node remove(int index) {
140         if(index < 0 || index >= length) return null;
141         if(index == 0) return removeFirst();
142         if(index == length - 1) return removeLast();
143 
144         Node prev = get(index - 1);
145         Node temp = prev.next;
146 
147         prev.next = temp.next;
148         temp.next = null;
149         length--;
150         return temp;
151     }
152 
153     public void reverse() {
154         Node temp = head;
155         head = tail;
156         tail = temp;
157         Node after = temp.next;
158         Node before = null;
159         for(int i = 0; i < length; i++) {
```

```
160             after = temp.next;
161             temp.next = before;
162             before = temp;
163             temp = after;
164         }
165     }
166 
167 }
168 
```