

```

1 package com.timbuchalka;
2
3 import java.util.Iterator;
4 import java.util.LinkedList;
5 import java.util.ListIterator;
6 import java.util.Scanner;
7
8 /**
9  * Demo
10  */
11 * main(String[] args)
12 *
13 * printList(LinkedList<String> linkedList)
14 * addInOrder(LinkedList<String> linkedList, String newCity)
15 * visit(LinkedList cities)
16 */
17 public class Demo {
18     public static void main(String[] args) {
19
20         /*
21          * LinkedList
22          * Each item in a LinkList is linked to the next item in a row.
23          */
24         LinkedList<String> placesToVisit = new LinkedList<String>();
25         addInOrder(placesToVisit, "Sydney");
26         addInOrder(placesToVisit, "Melbourne");
27         addInOrder(placesToVisit, "Brisbane");
28         addInOrder(placesToVisit, "Perth");
29         addInOrder(placesToVisit, "Canberra");
30         addInOrder(placesToVisit, "Adelaide");
31         addInOrder(placesToVisit, "Darwin");
32         printList(placesToVisit);
33
34         addInOrder(placesToVisit, "Alice Springs");
35         addInOrder(placesToVisit, "Darwin");
36         printList(placesToVisit);
37         visit(placesToVisit);
38     }
39
40
41     /**
42      * printList()
43      * @param linkedList
44      */
45     private static void printList(LinkedList<String> linkedList) {
46
47         /*
48          * Iterator
49          */
50         Iterator<String> i= linkedList.iterator();
51         while(i.hasNext()) {
52             System.out.println("Now visiting " + i.next());
53         }
54         System.out.println("=====");
55     }
56
57
58     /**
59      * addInOrder()
60      * @param linkedList
61      * @param newCity
62      * @return
63      */
64     private static boolean addInOrder(LinkedList<String> linkedList, String newCity) {
65
66         /*
67          * ListIterator
68          */
69         ListIterator<String> stringListIterator = linkedList.listIterator();
70
71         /*
72          * ListIterator.hasNext()
73          */
74         while(stringListIterator.hasNext()) {
75
76             /*
77              * ListIterator.next().compareTo(String)
78              * Returns the value {@code 0} if the argument string is equal to
79              * this string; a value less than {@code 0} if this string
80              * is lexicographically less than the string argument; and a
81              * value greater than {@code 0} if this string is
82              * lexicographically greater than the string argument.
83              */
84             int comparison = stringListIterator.next().compareTo(newCity);
85             if(comparison == 0) {
86                 // equal, do not add the same element twice
87                 System.out.println(newCity + " is already included as a destination");
88                 return false;
89             } else if(comparison > 0) {
90                 // new City should appear before .next()
91                 /*
92                  * ListIterator.previous()
93                  */
94                 stringListIterator.previous(); // switch back to the previous entry, as .next() already called
95                 /*
96                  * ListIterator.add(element)
97                  */

```

```

98         stringListIterator.add(newCity);
99         return true;
100     } else if(comparison < 0) {
101         // move on next city
102     }
103 }
104
105 // if end of the list reached by the while loop, add the city at the very end of the list
106 stringListIterator.add(newCity);
107 return true;
108 }
109
110
111 /**
112  * visit()
113  * @param cities
114  */
115 private static void visit(LinkedList cities) {
116     Scanner scanner = new Scanner(System.in);
117     boolean quit = false;
118     boolean goingForward = true;
119     ListIterator<String> listIterator = cities.listIterator();
120
121     if(cities.isEmpty()) {
122         System.out.println("No cities in the itinerary");
123         return;
124     } else {
125         System.out.println("Now visiting " + listIterator.next());
126         printMenu();
127     }
128
129     while (!quit) {
130         int action = scanner.nextInt();
131         scanner.nextLine();
132         switch(action) {
133             case 0:
134                 System.out.println("Holiday (Vacation) over");
135                 quit = true;
136                 break;
137
138             case 1:
139                 if(!goingForward) {
140                     if(listIterator.hasNext()) {
141                         listIterator.next();
142                     }
143                     goingForward = true;
144                 }
145                 if(listIterator.hasNext()) {
146                     System.out.println("Now visiting " + listIterator.next());
147                 } else {
148                     System.out.println("Reached the end of the list");
149                     goingForward = false;
150                 }
151                 break;
152
153             case 2:
154                 if(goingForward) {
155                     if(listIterator.hasPrevious()) {
156                         listIterator.previous();
157                     }
158                     goingForward = false;
159                 }
160                 if(listIterator.hasPrevious()) {
161                     System.out.println("Now visiting " + listIterator.previous());
162                 } else {
163                     System.out.println("We are at the start of the list");
164                     goingForward = true;
165                 }
166                 break;
167
168             case 3:
169                 printMenu();
170                 break;
171
172         }
173     }
174 }
175
176 private static void printMenu() {
177     System.out.println("Available actions:\npress ");
178     System.out.println("0 - to quit\n" +
179         "1 - go to next city\n" +
180         "2 - go to previous city\n" +
181         "3 - print menu options");
182 }
183 }
184 }
185

```