```java
package com.timbuchalka;

import java.util.Iterator;
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.Scanner;

/**
 * Demo
 *
 * main(String[] args)
 *
 * printList(LinkedList<String> linkedList)
 * addInOrder(LinkedList<String> linkedList, String newCity)
 * visit(LinkedList cities)
 */
public class Demo {
    public static void main(String[] args) {

        /*
         * LinkedList
         * Each item in a LinkList is linked to the next item in a row.
         */
        LinkedList<String> placesToVisit = new LinkedList<String>();
        addInOrder(placesToVisit, "Sydney");
        addInOrder(placesToVisit, "Melbourne");
        addInOrder(placesToVisit, "Brisbane");
        addInOrder(placesToVisit, "Perth");
        addInOrder(placesToVisit, "Canberra");
        addInOrder(placesToVisit, "Adelaide");
        addInOrder(placesToVisit, "Darwin");
        printList(placesToVisit);

        addInOrder(placesToVisit, "Alice Springs");
        addInOrder(placesToVisit, "Darwin");
        printList(placesToVisit);
        visit(placesToVisit);
    }


    /**
     * printList()
     * @param linkedList
     */
    private static void printList(LinkedList<String> linkedList) {

        /*
         * Iterator
         */
        Iterator<String> i= linkedList.iterator();
        while(i.hasNext()) {
            System.out.println("Now visiting " + i.next());
        }
        System.out.println("========================");
    }


    /**
     * addInOrder()
     * @param linkedList
     * @param newCity
     * @return
     */
    private static boolean addInOrder(LinkedList<String> linkedList, String newCity) {

        /*
         * ListIterator
         */
        ListIterator<String> stringListIterator = linkedList.listIterator();

        /*
         * ListIterator.hasNext()
         */
        while(stringListIterator.hasNext()) {

            /*
             * ListIterator.next().compereTo(String)
             * Returns the value {@code 0} if the argument string is equal to
             * this string; a value less than {@code 0} if this string
             * is lexicographically less than the string argument; and a
             * value greater than {@code 0} if this string is
             * lexicographically greater than the string argument.
             */
            int comparison = stringListIterator.next().compareTo(newCity);
            if(comparison == 0) {
                // equal, do not add the same element twice
                System.out.println(newCity + " is already included as a destination");
                return false;
            } else if(comparison > 0) {
                // new City should appear before .next()
                /*
                 * ListIterator.previous()
                 */
                stringListIterator.previous(); // switch back to the previous entry, as .next() already called
                /*
                 * ListIterator.add(element)
                 */
```

```
                stringListIterator.add(newCity);
                return true;
            } else if(comparison < 0) {
                // move on next city
            }
        }

        // if end of the list reached by the while loop, add the city at the very end of the list
        stringListIterator.add(newCity);
        return true;
    }


    /**
     * visit()
     * @param cities
     */
    private static void visit(LinkedList cities) {
        Scanner scanner = new Scanner(System.in);
        boolean quit = false;
        boolean goingForward = true;
        ListIterator<String> listIterator = cities.listIterator();

        if(cities.isEmpty()) {
            System.out.println("No cities in the itenerary");
            return;
        } else {
            System.out.println("Now visiting " + listIterator.next());
            printMenu();
        }

        while (!quit) {
            int action = scanner.nextInt();
            scanner.nextLine();
            switch(action) {
                case 0:
                    System.out.println("Holiday (Vacation) over");
                    quit = true;
                    break;

                case 1:
                    if(!goingForward) {
                        if(listIterator.hasNext()) {
                            listIterator.next();
                        }
                        goingForward = true;
                    }
                    if(listIterator.hasNext()) {
                        System.out.println("Now visiting " + listIterator.next());
                    } else {
                        System.out.println("Reached the end of the list");
                        goingForward = false;
                    }
                    break;

                case 2:
                    if(goingForward) {
                        if(listIterator.hasPrevious()) {
                            listIterator.previous();
                        }
                        goingForward = false;
                    }
                    if(listIterator.hasPrevious()) {
                        System.out.println("Now visiting " + listIterator.previous());
                    } else {
                        System.out.println("We are at the start of the list");
                        goingForward = true;
                    }
                    break;

                case 3:
                    printMenu();
                    break;

            }

        }
    }

    private static void printMenu() {
        System.out.println("Available actions:\npress ");
        System.out.println("0 - to quit\n" +
                "1 - go to next city\n" +
                "2 - go to previous city\n" +
                "3 - print menu options");
    }
}

```