|  |  |
| --- | --- |
| Computer Science 211 Data StructuresSpring, 2015 |  |

**Lab Report – Week *[15]* - *[Graphs Assignment]***

*[Malik Van Kirk]*

*[CSCI 211] [Spring 2015]*

**Assignment Analysis and Design**

*My Program uses a Graph cityProject provided. I created a linked list class and a method that uses the class to find the shortest distance between two locations. The method takes the first location, the final location and cities array as a formal parameter. It then compares all possible distances and returns the path.*

**Assignment Code**

*public static LinkedL shortP(City source, City destination, City[] cities)*

*{*

*City current = source;*

*current.setBestDistance(0);*

*current.setImmediatePredecessor(null);*

*//variable to hold adjacency list head*

*AdjacencyNode adj;*

*boolean finished = false;*

*//while not finsihed*

*while (finished == false)*

*{*

*//set the adjacency list head to be the one or the current city*

*adj = current.getAdjacencyListHead();*

*//while adjacent cities not visited*

*while (!(adj == null))*

*{*

*int dist = adj.getcDistance() + current.getBestDistance();*

*if (dist < adj.getCity().getBestDistance())*

*{*

*adj.getCity().setBestDistance(dist);*

*adj.getCity().setImmediatePredecessor(current);*

*}//end if*

*adj = adj .getNext();*

*}//end inner while*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*current.setVisited(true);*

*current = newCurrent(cities);*

*if (current == null) //if newCurrent() is null then must be no unvisited cities)*

*{*

*finished = true;*

*}//end if*

*}//end While*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*//set the current City to be the destination city*

*current = destination;*

*//Create a new Linked List, with the head node to be the destination city*

*LinkedL path = new LinkedL(destination);*

*while (current.getImmediatePredecessor()!=null)*

*{*

*path.addFirst(current.getImmediatePredecessor());*

*current = current.getImmediatePredecessor();*

*}*

*//return the completed linked list*

*return path;*

*}//end ShortP*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*static public class LinkedL*

*{*

*public LinkedL.Node head = null; //head of the linked list*

*public LinkedL.Node first = null; //first node of the linked list*

*//null constructor*

*public LinkedL()*

*{*

*}//end constrcut*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*public LinkedL(City c)*

*{*

*first = new LinkedL.Node(c, null);*

*head = first;*

*}//end LinkedL*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*public void addFirst(City c)*

*{*

*LinkedL.Node newFirst = new LinkedL.Node(c, head);*

*head = newFirst;*

*}//end AddFirst*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*public LinkedL.Node getHead()*

*{*

*return this.head;*

*}//end getHead*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*public String toString()*

*{*

*String info = ""; //set string to null*

*StringBuffer s = new StringBuffer(info); //create stringbuffer for string*

*LinkedL.Node pointer = this.getHead(); //set pointer node to be the list's head*

*while (pointer != null) //append each city's name to the string until there aren't any more cities*

*{*

*info = pointer.data.getName();*

*if (pointer.pointer == null)*

*{*

*s.append(info + ".");*

*}//end if*

*else*

*{*

*s.append(info + ", ");*

*}//end else*

*pointer = pointer.pointer;*

*}//end while*

*return s.toString();*

*}//end toString*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*class Node*

*{*

*public City data; //holds City data*

*public LinkedL.Node pointer; //holds node data*

*public Node()*

*{*

*this.data = null;*

*this.pointer = null;*

*}//end Node*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*public Node(City d, LinkedL.Node n)*

*{*

*this.data = d;*

*this.pointer = n;*

*}//end Node*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*public String toString()*

*{*

*String info = data + ", {" + pointer + "}";*

*return info;*

*}//end toString*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*} //End Node*

*} //End LinkedL*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**Assignment Testing**

*The only problem I had was if the user input was not formatted exactly as it is in the Cities array then an error will occur.*

**Assignment Evaluation**

*I learned…or rather I should have known an error would occur if it was not inputted correctly*