\*\*For Grading

1 import java.util.\*;  
 2 import java.io.\*;  
 3   
 4 public class UPGMA  
 5 {  
 6 private double[][] distanceM;  
 7 private List<String> names;  
 8 private List<String> updated\_names;  
 9 private int min\_row; //row of min  
 10 private int min\_column; //column of min in matrix  
 11 private int pos; //index of most recent OTU  
 12   
 13 public static void main(String[] args)  
 14 {  
 15 double[][] dis = {  
 16 null,  
 17 {19},  
 18 {27, 31},  
 19 {8, 18, 26},  
 20 {33, 36, 41, 31},  
 21 {18, 1, 32, 17, 35},  
 22 {13, 13, 29, 14, 28, 12}  
 23 };  
 24 List<String> list = new ArrayList<String>();  
 25 list.add("turtle");  
 26 list.add("man");  
 27 list.add("tuna");  
 28 list.add("chicken");  
 29 list.add("moth");  
 30 list.add("monkey");  
 31 list.add("dog");   
 32 UPGMA u = new UPGMA(dis, list);  
 33 u.treeBuild();   
 34 }  
 35 public UPGMA(double[][] d, List<String> m)  
 36 {  
 37 distanceM = d;  
 38 names = m;  
 39 }  
 40 public void treeBuild() //full UPGMA algorithm with printing  
 41 {  
 42 int count = 1;  
 43 while(distanceM.length>1)  
 44 {  
 45 System.out.println("iteration " + count);  
 46 toTable(); //print to table  
 47   
 48 //algorithm  
 49 minimum();  
 50 joinLabels();  
 51 System.out.println(names.get(min\_row)+" and " + names.get(min\_column)+ " are the nearest");  
 52 System.out.println("The distance between them is " + distanceM[min\_row][min\_column]);  
 53 rebuild();  
 54   
 55 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 56 System.out.println();  
 57 count++;  
 58 }  
 59 }  
 60   
 61 public void rebuild() //updates distance matrix after merging taxa  
 62 {  
 63 double d1,d2 = 0;  
 64 double [][] d = new double[updated\_names.size()][updated\_names.size()];  
 65   
 66   
 67 for(int i = 1; i<d.length-1;i++)  
 68 {  
 69 for(int j = 0;j<i;j++)  
 70 {   
 71 d[i][j] = distanceM[names.indexOf(updated\_names.get(i))][names.indexOf(updated\_names.get(j))];  
 72 }   
 73 }  
 74 for(int k = 0; k < d.length-1; k++)  
 75 {  
 76 d1 = calc(names.indexOf(updated\_names.get(k)), min\_column);  
 77 d2 = calc(min\_row, names.indexOf(updated\_names.get(k)));  
 78 d[d.length-1][k] = (d1 + d2) \* 0.5;  
 79 }  
 80 distanceM = d;  
 81 names = updated\_names;  
 82 }  
 83 public double calc(int a, int b)   
 84 {  
 85 int temp = 0;  
 86 if(b > a) //all data in left corner triangle  
 87 {  
 88 temp = a;  
 89 a = b;  
 90 b = temp;  
 91 }  
 92 return distanceM[a][b];  
 93 }  
 94 public void joinLabels() //merges two taxa together  
 95 {  
 96 updated\_names = new ArrayList<String>(names);  
 97 updated\_names.add(names.get(min\_row)+ ", " + names.get(min\_column));  
 98 if(min\_column<min\_row)  
 99 {  
100 updated\_names.remove(min\_column);  
101 updated\_names.remove(min\_row-1);  
102 } else {  
103 updated\_names.remove(min\_row);  
104 updated\_names.remove(min\_column);  
105 }  
106 }   
107 public void minimum() //finds minimum distance in entire array  
108 {  
109 double min = distanceM[1][0];  
110 min\_row = 1;  
111 min\_column = 0;  
112   
113 for(int i = 1; i<distanceM.length; i++) //start at row 1  
114 {  
115 for(int j = 0; j<i; j++)  
116 {  
117 if(distanceM[i][j] < min)  
118 {  
119 min = distanceM[i][j];  
120 min\_row = i;  
121 min\_column = j;  
122 }  
123 }  
124 }  
125 }  
126 public void toTable()  
127 {  
128 List<List> rows = new ArrayList<List>();  
129 names.add(0," ");  
130 rows.add(names);  
131   
132 //finds max length in name list  
133 int max = 0;  
134 for(int i = 0; i<names.size(); i++)  
135 {  
136 if(names.get(i).length()>max)  
137 max = names.get(i).length();  
138 }  
139   
140 for (int i=0; i<distanceM.length; i++) {  
141 List<String> column = new ArrayList<String>();  
142 column.add(names.get(i+1));//add label as first element  
143 for (int j=0; j<i; j++) {  
144 column.add(new Double(distanceM[i][j]).toString());  
145   
146 }  
147 rows.add(column);  
148 }  
149 for (int i = 0; i < rows.size(); i++)//rows  
150 {  
151 List column = rows.get(i);  
152 for (int j = 0; j < rows.get(i).size(); j++) //columns  
153 {  
154 if(j == 0)  
155 System.out.printf("%" + max +"s", rows.get(i).get(j));  
156 else  
157 System.out.printf("%" + (((String)rows.get(0).get(j)).length() + 5) +"s", rows.get(i).get(j));  
158 }  
159   
160 System.out.println();  
161 }  
162 System.out.println();  
163 names.remove(0);   
164 }  
165 }

Output of HW

Graphical user interface, text, application, email

Description automatically generatedA picture containing text, screenshot, receipt

Description automatically generated