```java
import javax.swing.*;

public class RouterNode {
  private int myID;
  private GuiTextArea myGUI;
  private RouterSimulator sim;
  private int[] costs = new int[RouterSimulator.NUM_NODES];
  // Possible routes to take represented by ID of nodes
  private int[] routes = new int[RouterSimulator.NUM_NODES];
  // Node ID connected to cost
  private int[][] vectors = new int[RouterSimulator.NUM_NODES][RouterSimulator.NUM_NODES];

  boolean poisonReverse = false;

  //--------------------------------------------------
  public RouterNode(int ID, RouterSimulator sim, int[] costs) {
    myID = ID;
    this.sim = sim;
    myGUI =new GuiTextArea("  Output window for Router #"+ ID + "  ");

    // Assign your own costs.
    System.arraycopy(costs, 0, this.costs, 0, RouterSimulator.NUM_NODES);

    // Go through amount of nodes on network.
    for(int i = 0; i < RouterSimulator.NUM_NODES; i++)
    {
      // Add the node to the network.
      routes[i] = i;
      for(int j = 0; j < RouterSimulator.NUM_NODES; j++)
      {
        // Insert your own costs in neighbour table.
        // INFINITY to non-neighbours.
        if( i == myID )
        {
          vectors[i][j] = costs[j];
        }
        else
        {
          vectors[i][j] = RouterSimulator.INFINITY;
        }
      }
    }
    sendToNeighbours();
  }
  //--------------------------------------------------
  private void sendToNeighbours() {
    int[] sendCosts = new int[RouterSimulator.NUM_NODES];

    // go through all *nodes*
    for (int neighbour = 0; neighbour <RouterSimulator.NUM_NODES; neighbour++)
    {
      // make sure we only go through *neighbours*
      if ( neighbour != myID && costs[neighbour] != RouterSimulator.INFINITY)
      {
        // copy vectors from current node in case we want to alter them.
        System.arraycopy(vectors[myID], 0, sendCosts, 0, RouterSimulator.NUM_NODES);
```

```
57
58          if ( poisonReverse )
59          {
60            // loop through all possible destinations.
61            for (int dest = 0; dest < RouterSimulator.NUM_NODES; dest++)
62            {
63              // if our route to destination takes us through the
64              //  neighbour we're on right now, send our cost to
65              //  destination to infinity.
66              if ( routes[dest] == neighbour && dest != neighbour )
67              {
68                sendCosts[dest] = RouterSimulator.INFINITY;
69              }
70            }
71          }
72          // send current packet.
73          RouterPacket packet = new RouterPacket(myID, neighbour, sendCosts);
74          sendUpdate(packet);
75        }
76      }
77    }
78
79    //--------------------------------------------------
80    private boolean modifyCost() {
81      boolean costChanged = false;
82
83      // Go through all nodes on the network
84      //  to find the cheapest path to the destination.
85      for (int destination = 0; destination < RouterSimulator.NUM_NODES; destination++)
86      {
87        // you're not supposed to go to yourself
88        if ( destination != myID )
89        {
90          // set cheapest route to initial cost
91          int cheapest = costs[destination];
92          // set route to current destination
93          int route = destination;
94          // go through possible routes to destination
95          for (int neighbour = 0; neighbour < RouterSimulator.NUM_NODES; neighbour++)
96          {
97            // route is not to yourself nor a non-neighbour
98            if ( neighbour != myID && costs[neighbour] != RouterSimulator.INFINITY )
99            {
100             // calculate cost based on cost to neighbour and
101             //  neighbours cost to destination
102             int altCost = costs[neighbour] + vectors[neighbour][destination];
103             // check if the possible path is cheaper than current cheapest path
104             if (altCost < cheapest)
105             {
106               cheapest = altCost;
107               route = neighbour;
108             }
109           }
110         }
111
112         // if a route has changed, update vectors' route and cost
113         if (cheapest != vectors[myID][destination])
```

```
114           {
115             vectors[myID][destination] = cheapest;
116             routes[destination] = route;
117             costChanged = true;
118           }
119         }
120       }
121       // A cost has been modified or not
122       return costChanged;
123     }
124
125     //--------------------------------------------------
126     public void recvUpdate(RouterPacket pkt) {
127       // overwrite the old vectors from the source with new ones.
128       System.arraycopy(pkt.mincost, 0, vectors[pkt.sourceid], 0, RouterSimulator.
        NUM_NODES);
129
130       // check if anything has changed.
131       if ( modifyCost() )
132       {
133         sendToNeighbours();
134       }
135     }
136
137     //--------------------------------------------------
138     private void sendUpdate(RouterPacket pkt) {
139       sim.toLayer2(pkt);
140     }
141
142     //--------------------------------------------------
143     public void printDistanceTable() {
144       // oof. trust us. it works. minimize and move on.
145       myGUI.println(" Current state for " + myID +
146         "  at time " + sim.getClocktime());
147
148       myGUI.println();
149
150       myGUI.println(" Distancetable:");
151       String strHeader = F.format(" dest |", 10);
152       for (int i = 0; i < RouterSimulator.NUM_NODES; i++)
153       {
154         strHeader+= F.format(Integer.toString(i), 6);
155       }
156       myGUI.println(strHeader);
157       myGUI.println(new String(new char[strHeader.length()]).replace("\0", "-"));
158
159       // Setup printing for neighbours
160       for (int i = 0; i < RouterSimulator.NUM_NODES; i++)
161       {
162           String str = F.format("nbr ", 5);
163           str+=F.format(Integer.toString(i), 3);
164           str+=F.format("|", 2);
165           for (int j = 0; j < RouterSimulator.NUM_NODES; j++)
166           {
167             str+=F.format(Integer.toString(vectors[i][j]), 6);
168           }
169           myGUI.println(str);
```

```
170       }
171       myGUI.println();
172       myGUI.println(" Our distance vector and routes:");
173
174       strHeader = F.format(" dest |", 10);
175       for (int i = 0; i < RouterSimulator.NUM_NODES; i++)
176       {
177         strHeader+= F.format(Integer.toString(i), 6);
178       }
179       myGUI.println(strHeader);
180       // Print a line
181       myGUI.println(new String(new char[strHeader.length()]).replace("\0", "-"));
182
183       // Print routes double woohoo
184       String strRoutes = F.format("route", 6);
185       strRoutes+= F.format("|", 4);
186       for (int i = 0; i < RouterSimulator.NUM_NODES; i++)
187       {
188         if ( i == myID )
189         {
190           strRoutes+=F.format("-", 6);
191         }
192         else
193         {
194           strRoutes+=F.format(routes[i], 6);
195         }
196       }
197       myGUI.println(strRoutes);
198
199       // Print costs woohoo
200       String strCosts = F.format("cost", 5);
201       strCosts+= F.format("|", 5);
202       for (int i = 0; i < RouterSimulator.NUM_NODES; i++)
203       {
204          strCosts+= F.format(Integer.toString(vectors[myID][i]), 6);
205       }
206       myGUI.println(strCosts);
207
208       myGUI.println();
209     }
210
211     //--------------------------------------------------
212     public void updateLinkCost(int dest, int newcost) {
213       // alter the cost to the affected node.
214       // only time it's allowed to alter the variable 'costs'.
215       costs[dest] = newcost;
216
217       if ( modifyCost() )
218       {
219         sendToNeighbours();
220       }
221     }
222   }
223
```