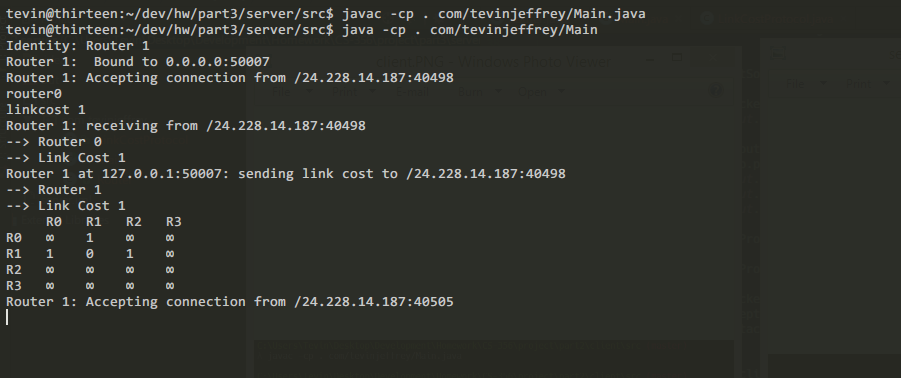
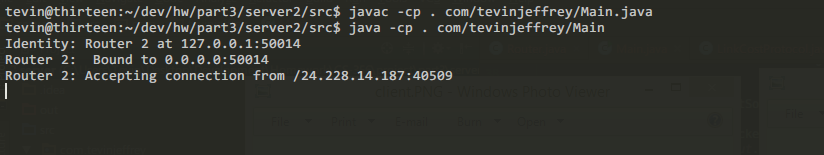
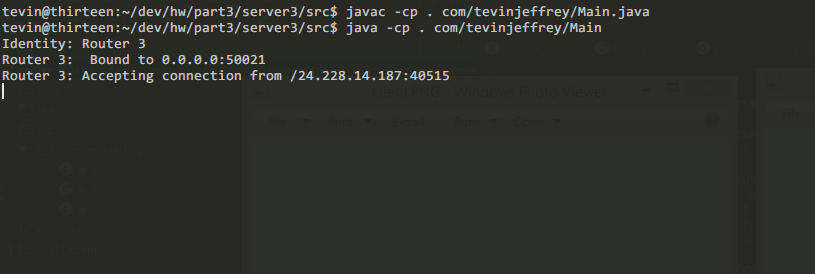
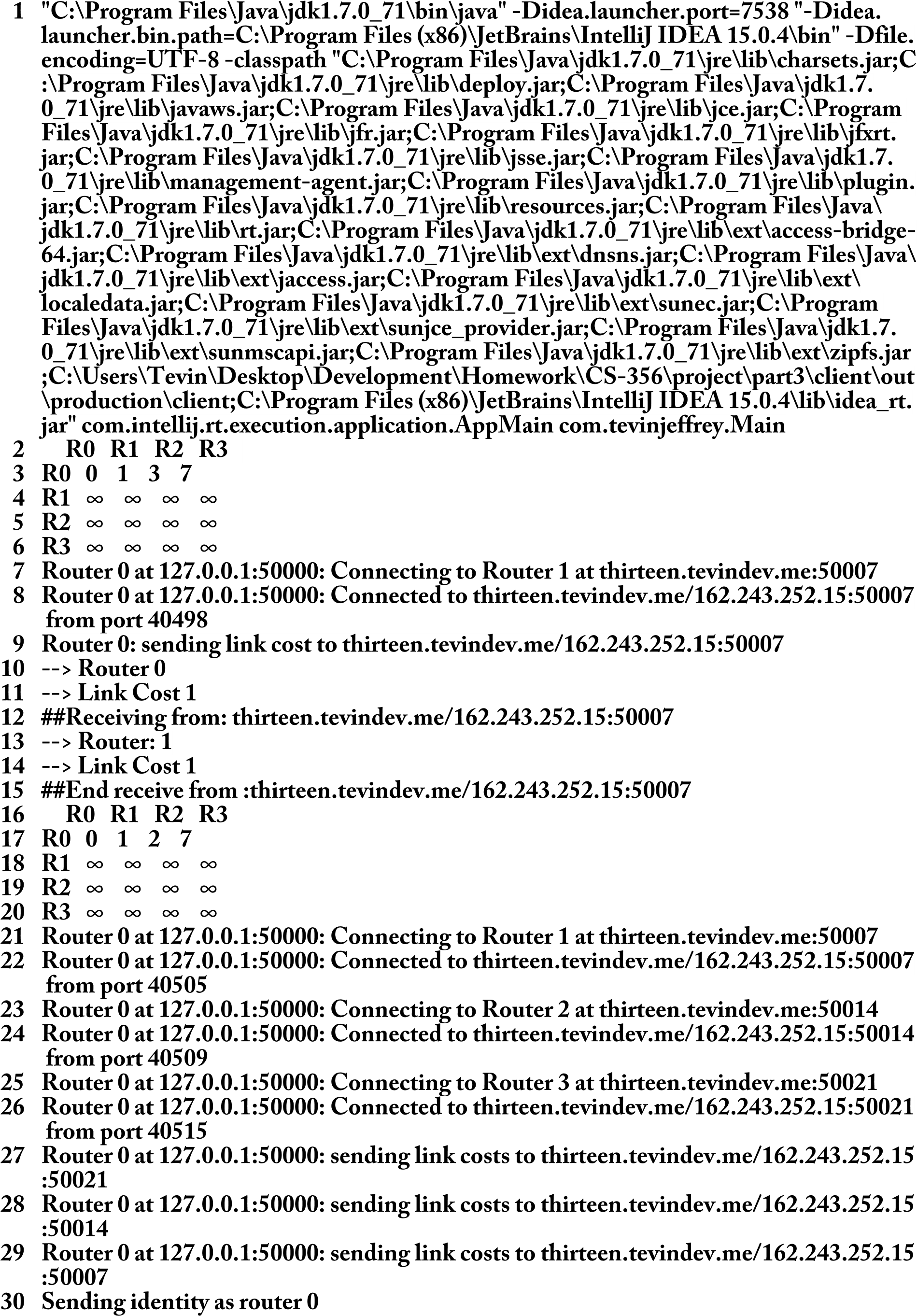
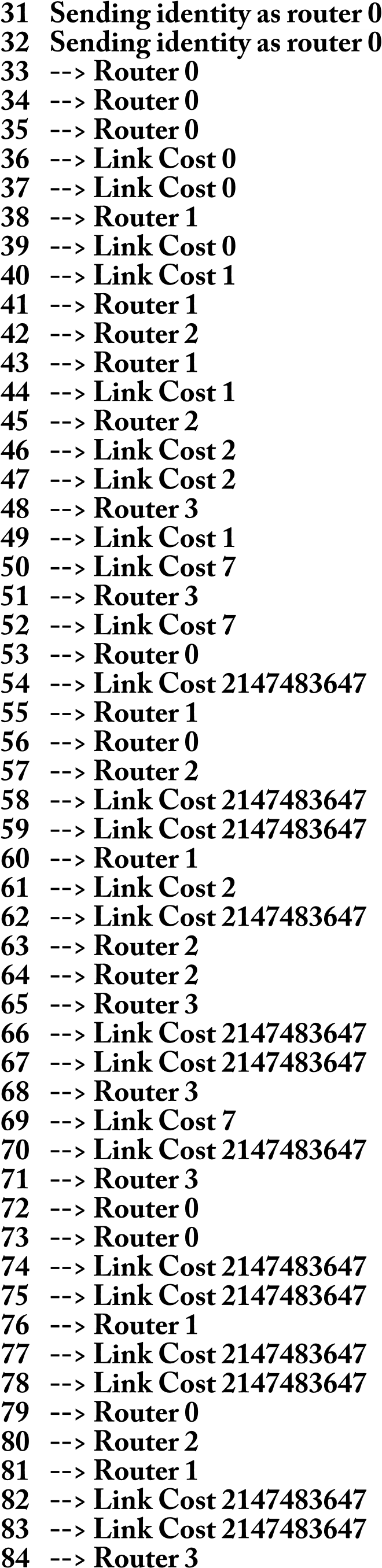
Stage 3

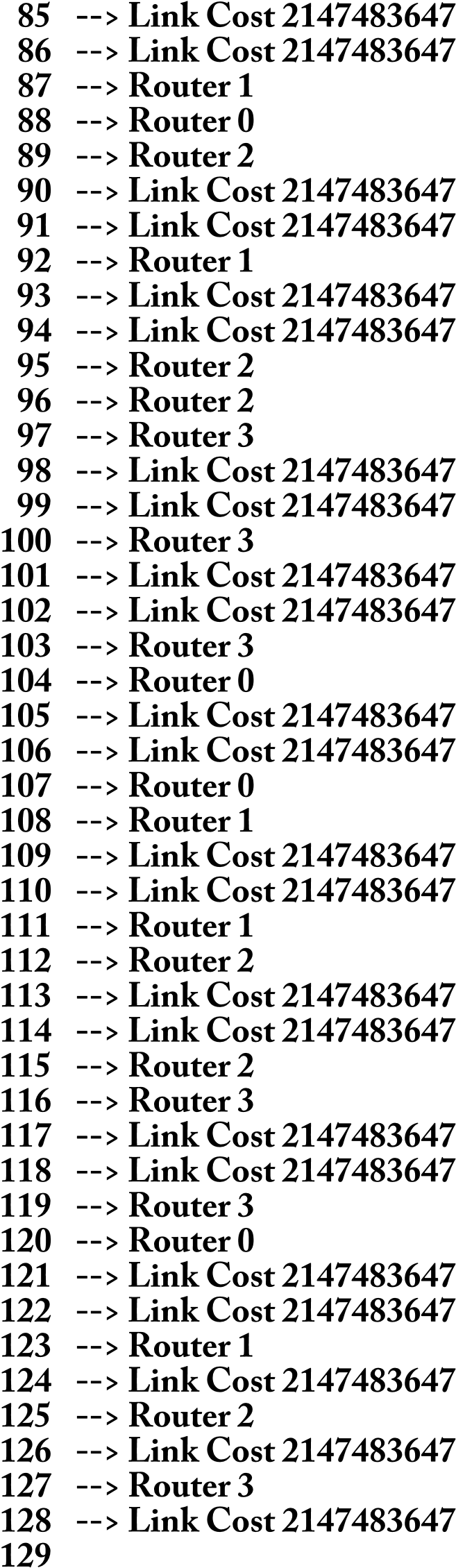
Intro to Networking

Tevin Jeffrey









package com.tevinjeffrey;  
  
import java.net.Socket;  
import java.util.HashMap;  
import java.util.Map;  
  
public class Main {  
 public static void main(String[] args) {  
 String who = "Router 0";  
  
 Router router0 = new Router("Router 0", "127.0.0.1", 50000);  
 Router router1 = new Router("Router 1", "thirteen.tevindev.me", 50007);  
 Router router2 = new Router("Router 2", "thirteen.tevindev.me", 50014);  
 Router router3 = new Router("Router 3", "thirteen.tevindev.me", 50021);  
  
 Map<Integer, Router> routerMap = new HashMap<>();  
 routerMap.put(router0.getNumber(), router0);  
 routerMap.put(router1.getNumber(), router1);  
 routerMap.put(router2.getNumber(), router2);  
 routerMap.put(router3.getNumber(), router3);  
  
 router0.addLinkCost(router0.getNumber(), 0);  
 router0.addLinkCost(router1.getNumber(), 1);  
 router0.addLinkCost(router2.getNumber(), 3);  
 router0.addLinkCost(router3.getNumber(), 7);  
  
 LinkCostProtocol.*printLinkCost*(routerMap);  
  
 try {  
 Socket router1Client = LinkCostProtocol.*estab*(router1, router0);  
  
 //Send link cost to to router 1  
 LinkCostProtocol.*send*(router1Client, router0, router1, who);  
 //Receive table from router 1  
 LinkCostProtocol.*recv*(router1Client, router0);  
  
 routerMap.get(0).addLinkCost(2,2);  
 LinkCostProtocol.*printLinkCost*(routerMap);  
 //router1Client.close();  
  
 LinkCostProtocol.*broadcast*(router0, routerMap);  
  
 } catch (Exception e) {  
 System.*out*.println(who +": Could not establish connection. " + e.getMessage());  
 }  
 }  
}

package com.tevinjeffrey;  
  
import java.io.\*;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
import java.util.concurrent.ThreadPoolExecutor;  
import java.util.concurrent.TimeUnit;  
  
public class LinkCostProtocol {  
  
 InputStream inputStream;  
 OutputStream outputStream;  
  
 public LinkCostProtocol(InputStream inputStream, OutputStream outputStream) {  
 this.inputStream = inputStream;  
 this.outputStream = outputStream;  
 }  
  
 public static void recv(Socket clientSocket, Router receivingRouter) throws Exception {  
 BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
 System.*out*.println("##Receiving from: " + clientSocket.getInetAddress().toString() + ":"+clientSocket.getPort());  
  
 String routerNumber = in.readLine();  
 String linkCost = in.readLine();  
  
 if (routerNumber == null || linkCost == null) {  
 throw new Exception("Router number or Link cost not received.");  
 }  
  
 *printRandL*(routerNumber, linkCost);  
 System.*out*.println("##End receive from :" + clientSocket.getInetAddress().toString() + ":"+clientSocket.getPort());  
 receivingRouter.addLinkCost(Integer.*parseInt*(routerNumber), Integer.*parseInt*(linkCost));  
 }  
  
 public static Socket estab(Router router, Router who) throws IOException {  
 System.*out*.println(who +": Connecting to " + router);  
 Socket client = new Socket(router.getHost(), router.getPort());  
 System.*out*.println(who +": Connected to " + client.getRemoteSocketAddress() + " from port " + client.getLocalPort());  
 return client;  
 }  
  
  
 public static void recvAll(Socket clientSocket, Map<Integer, Router> routerMap) throws Exception {  
 BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
 Router receivingRouter;  
 //Get identity  
 String identity = in.readLine();  
 switch (identity){  
 case "0":  
 receivingRouter = routerMap.get(0);  
 break;  
 case "1":  
 receivingRouter = routerMap.get(1);  
 break;  
 case "2":  
 receivingRouter = routerMap.get(2);  
 break;  
 case "3":  
 receivingRouter = routerMap.get(3);  
 break;  
 default:  
 throw new Exception("Can't identify router.");  
 }  
 System.*out*.println("##Receiving from: " + clientSocket.getInetAddress().toString() + ":"+clientSocket.getPort());  
  
 System.*out*.println("Remote identity: " + receivingRouter);  
 System.*out*.println("table contents...");  
 *recvOne*(in, receivingRouter);  
  
 System.*out*.println("##End receive from :" + clientSocket.getInetAddress().toString() + ":"+clientSocket.getPort());  
  
 }  
  
 public static void recvOne(BufferedReader in, Router receivingRouter) throws Exception {  
 String routerNumber = in.readLine();  
 String linkCost = in.readLine();  
  
 *printRandL*(routerNumber, linkCost);  
 if (routerNumber == null || linkCost == null) {  
 throw new Exception("Router number or Link cost not received.");  
 } else {  
 receivingRouter.addLinkCost(0, Integer.*valueOf*(linkCost));  
 }  
  
  
 routerNumber = in.readLine();  
 linkCost = in.readLine();  
 *printRandL*(routerNumber, linkCost);  
  
 if (routerNumber == null || linkCost == null) {  
 throw new Exception("Router number or Link cost not received.");  
 } else {  
 receivingRouter.addLinkCost(1, Integer.*valueOf*(linkCost));  
 }  
  
 routerNumber = in.readLine();  
 linkCost = in.readLine();  
 *printRandL*(routerNumber, linkCost);  
  
 if (routerNumber == null || linkCost == null) {  
 throw new Exception("Router number or Link cost not received.");  
 } else {  
 receivingRouter.addLinkCost(2, Integer.*valueOf*(linkCost));  
 }  
  
 routerNumber = in.readLine();  
 linkCost = in.readLine();  
 *printRandL*(routerNumber, linkCost);  
  
 if (routerNumber == null || linkCost == null) {  
 throw new Exception("Router number or Link cost not received.");  
 } else {  
 receivingRouter.addLinkCost(3, Integer.*valueOf*(linkCost));  
 }  
 }  
  
 private static void printRandL(String routerNumber, String linkCost) {  
 System.*out*.println("--> Router: " + routerNumber);  
 System.*out*.println("--> Link Cost " + linkCost);  
 }  
  
 public static void send(Socket client, Router router, Router toRouter, String who) throws IOException {  
 PrintWriter printWriter = new PrintWriter(client.getOutputStream());  
 printWriter.print(String.*valueOf*(router.getNumber()) + "\n" + router.getLinkCost(toRouter.getNumber()) + "\n");  
 printWriter.flush();  
 System.*out*.println(who + ": sending link cost to " + client.getRemoteSocketAddress());  
 System.*out*.println("--> Router " + router.getNumber());  
 System.*out*.println("--> Link Cost " + router.getLinkCost(toRouter.getNumber()));  
 }  
  
 public static void broadcast(final Router fromRouter, final Map<Integer, Router> routerMap) throws IOException {  
 List<Socket> socketList = *getConnections*(routerMap, fromRouter);  
  
 ExecutorService service = Executors.*newFixedThreadPool*(4);  
 for (final Socket client : socketList) {  
 Runnable runnable = new Runnable() {  
  
 @Override  
 public void run() {  
 System.*out*.println(fromRouter + ": sending link costs to " + client.getRemoteSocketAddress());  
  
 PrintWriter printWriter = null;  
 try {  
 printWriter = new PrintWriter(client.getOutputStream());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 *sendAll*(routerMap, fromRouter, printWriter);  
  
 }  
 };  
 service.submit(runnable);  
 }  
 }  
  
 public static List<Socket> getConnections(Map<Integer, Router> routerMap, Router router) {  
 List<Socket> sockets = new ArrayList<>();  
 for (int i = 0; i < routerMap.size(); i++) {  
 Router curr = routerMap.get(i);  
 if (router.equals(curr)){  
 continue;  
 } else {  
 try {  
 sockets.add(*estab*(curr, router));  
 } catch (IOException e) {  
 System.*out*.println("Connection refused");  
 }  
 }  
 }  
 return sockets;  
 }  
  
 public static void sendAll(Map<Integer, Router> routerMap, Router fromRouter, PrintWriter printWriter) {  
 System.*out*.println("Sending identity as router "+ fromRouter.getNumber());  
 printWriter.print(fromRouter.getNumber() + "\n");  
 *sendOne*(routerMap.get(0), printWriter);  
 *sendOne*(routerMap.get(1), printWriter);  
 *sendOne*(routerMap.get(2), printWriter);  
 *sendOne*(routerMap.get(3), printWriter);  
 }  
  
 public static void sendOne(Router router, PrintWriter printWriter) {  
 System.*out*.println("--> Router " + 0);  
 System.*out*.println("--> Link Cost " + router.getLinkCost(0));  
 printWriter.print(String.*valueOf*(0) + "\n" + router.getLinkCost(0) + "\n");  
  
 System.*out*.println("--> Router " + 1);  
 System.*out*.println("--> Link Cost " + router.getLinkCost(1));  
 printWriter.print(String.*valueOf*(1) + "\n" + router.getLinkCost(1) + "\n");  
  
 System.*out*.println("--> Router " + 2);  
 System.*out*.println("--> Link Cost " + router.getLinkCost(2));  
 printWriter.print(String.*valueOf*(2) + "\n" + router.getLinkCost(2) + "\n");  
  
 System.*out*.println("--> Router " + 3);  
 System.*out*.println("--> Link Cost " + router.getLinkCost(3));  
 printWriter.print(String.*valueOf*(3) + "\n" + router.getLinkCost(3) + "\n");  
 }  
  
 public static void printLinkCost(Map<Integer, Router> routerMap) {  
 Router router0 = routerMap.get(0);  
 Router router1 = routerMap.get(1);  
 Router router2 = routerMap.get(2);  
 Router router3 = routerMap.get(3);  
  
 System.*out*.printf(" %-5s%-5s%-5s%-5s\n", "R0", "R1", "R2", "R3");  
 System.*out*.printf("%-5s%-5s%-5s%-5s%-5s\n", "R0", router0.getStringLinkCost(0), router0.getStringLinkCost(1), router0.getStringLinkCost(2), router0.getStringLinkCost(3));  
 System.*out*.printf("%-5s%-5s%-5s%-5s%-5s\n", "R1", router1.getStringLinkCost(0), router1.getStringLinkCost(1), router1.getStringLinkCost(2), router1.getStringLinkCost(3));  
 System.*out*.printf("%-5s%-5s%-5s%-5s%-5s\n", "R2", router2.getStringLinkCost(0), router2.getStringLinkCost(1), router2.getStringLinkCost(2), router2.getStringLinkCost(3));  
 System.*out*.printf("%-5s%-5s%-5s%-5s%-5s\n", "R3", router3.getStringLinkCost(0), router3.getStringLinkCost(1), router3.getStringLinkCost(2), router3.getStringLinkCost(3));  
 }  
}

package com.tevinjeffrey;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class Router {  
  
 private String name;  
 private int number;  
 private String host;  
 private int port;  
 private Map<Integer, Integer> linkCost = new HashMap<>();  
  
 public Router(String name, String host, int port) {  
 this.name = name;  
 this.host = host;  
 this.port = port;  
 number = Integer.*parseInt*(name.substring(name.length() -1));  
  
 addLinkCost(0, Integer.*MAX\_VALUE*);  
 addLinkCost(1, Integer.*MAX\_VALUE*);  
 addLinkCost(2, Integer.*MAX\_VALUE*);  
 addLinkCost(3, Integer.*MAX\_VALUE*);  
  
 }  
  
 public Router(String name) {  
 this.name = name;  
 number = Integer.*parseInt*(name.substring(name.length() -1));  
 }  
  
 public Router() {  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getHost() {  
 return host;  
 }  
  
 public int getPort() {  
 return port;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
  
 public void addLinkCost(int router, int linkCost) {  
 this.linkCost.put(router, linkCost);  
 }  
  
 public int getLinkCost(int router) {  
 return linkCost.get(router);  
 }  
  
 public String getStringLinkCost(int router) {  
 if (linkCost.get(router) == Integer.*MAX\_VALUE*) {  
 return"\u221e";  
 }  
 return String.*valueOf*(linkCost.get(router));  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (obj == null) return false;  
 if (obj == this) return true;  
 return obj instanceof Router && this.name.equals(((Router) obj).getName());  
 }  
  
 @Override  
 public String toString() {  
 return name + " at " + host +":" + port;  
 }  
}

package com.tevinjeffrey;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class Main {  
 public static void main(String[] args) {  
 String who = "Router 1";  
  
 List<Router> knownRouters = new ArrayList<>();  
 Router router0 = new Router("Router 0", "127.0.0.1", 50000);  
 Router router1 = new Router("Router 1", "127.0.0.1", 50007);  
 Router router2 = new Router("Router 2", "127.0.0.1", 50014);  
 Router router3 = new Router("Router 3", "127.0.0.1", 50021);  
  
 Map<Integer, Router> routerMap = new HashMap<>();  
 routerMap.put(router0.getNumber(), router0);  
 routerMap.put(router1.getNumber(), router1);  
 routerMap.put(router2.getNumber(), router2);  
 routerMap.put(router3.getNumber(), router3);  
  
 router1.addLinkCost(router0.getNumber(), 1);  
 router1.addLinkCost(router1.getNumber(), 0);  
 router1.addLinkCost(router2.getNumber(), 1);  
 router1.addLinkCost(router3.getNumber(), Integer.*MAX\_VALUE*);  
  
 System.*out*.println("Identity: " + router1.getName());  
  
 try {  
 ServerSocket serverSocket = new ServerSocket(router1.getPort());  
 System.*out*.println(who +": Bound to " + serverSocket.getInetAddress().getHostAddress()  
 + ":"+serverSocket.getLocalPort());  
  
 while(true) {  
 Socket clientSocket = null;  
 try {  
 clientSocket = serverSocket.accept();  
 System.*out*.println(who +": Accepting connection from " + clientSocket.getRemoteSocketAddress());  
  
 Router router = LinkCostProtocol.*recv*(clientSocket, router1);  
 routerMap.put(router.getNumber(), router);  
 System.*out*.println(who +": receiving from "+clientSocket.getRemoteSocketAddress().toString());  
 System.*out*.println("--> Router " + router.getNumber());  
 System.*out*.println("--> Link Cost " + router1.getLinkCost(router.getNumber()));  
  
 LinkCostProtocol.*send*(clientSocket, router1, router);  
  
 LinkCostProtocol.printLinkCost(routerMap);  
  
 clientSocket.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (clientSocket != null) {  
 clientSocket.close();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 }  
  
 } catch (IOException e) {  
 System.*out*.println("Failed to bind to "+router1.getPort());  
 }  
 }  
}

package com.tevinjeffrey;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class Main {  
 public static void main(String[] args) {  
 String who = "Router 2";  
  
 List<Router> knownRouters = new ArrayList<>();  
 Router router0 = new Router("Router 0", "127.0.0.1", 50000);  
 Router router1 = new Router("Router 1", "127.0.0.1", 50007);  
 Router router2 = new Router("Router 2", "127.0.0.1", 50014);  
 Router router3 = new Router("Router 3", "127.0.0.1", 50021);  
  
 Map<Integer, Router> routerMap = new HashMap<>();  
 routerMap.put(router0.getNumber(), router0);  
 routerMap.put(router1.getNumber(), router1);  
 routerMap.put(router2.getNumber(), router2);  
 routerMap.put(router3.getNumber(), router3);  
  
 router2.addLinkCost(router0.getNumber(), 3);  
 router2.addLinkCost(router1.getNumber(), 1);  
 router2.addLinkCost(router2.getNumber(), 0);  
 router2.addLinkCost(router3.getNumber(), 2);  
  
 System.*out*.println("Identity: " + router2);  
  
 try {  
 ServerSocket serverSocket = new ServerSocket(router2.getPort());  
 System.*out*.println(who +": Bound to " + serverSocket.getInetAddress().getHostAddress()  
 + ":"+serverSocket.getLocalPort());  
  
 while(true) {  
 Socket clientSocket = null;  
 try {  
 clientSocket = serverSocket.accept();  
 System.*out*.println(who +": Accepting connection from " + clientSocket.getRemoteSocketAddress());  
  
 LinkCostProtocol.*recvAll*(clientSocket, routerMap);  
  
 LinkCostProtocol.*printLinkCost*(routerMap);  
  
 clientSocket.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (clientSocket != null) {  
 clientSocket.close();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 }  
  
 } catch (IOException e) {  
 System.*out*.println("Failed to bind to "+router2.getPort());  
 }  
 }  
}

package com.tevinjeffrey;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class Main {  
 public static void main(String[] args) {  
 String who = "Router 3";  
  
 List<Router> knownRouters = new ArrayList<>();  
 Router router0 = new Router("Router 0", "127.0.0.1", 50000);  
 Router router1 = new Router("Router 1", "127.0.0.1", 50007);  
 Router router2 = new Router("Router 2", "127.0.0.1", 50014);  
 Router router3 = new Router("Router 3", "127.0.0.1", 50021);  
  
 Map<Integer, Router> routerMap = new HashMap<>();  
 routerMap.put(router0.getNumber(), router0);  
 routerMap.put(router1.getNumber(), router1);  
 routerMap.put(router2.getNumber(), router2);  
 routerMap.put(router3.getNumber(), router3);  
  
 router3.addLinkCost(router0.getNumber(), 7);  
 router3.addLinkCost(router1.getNumber(), Integer.*MAX\_VALUE*);  
 router3.addLinkCost(router2.getNumber(), 2);  
 router3.addLinkCost(router3.getNumber(), 0);  
  
 System.*out*.println("Identity: " + router3.getName());  
  
 try {  
 ServerSocket serverSocket = new ServerSocket(router3.getPort());  
 System.*out*.println(who +": Bound to " + serverSocket.getInetAddress().getHostAddress()  
 + ":"+serverSocket.getLocalPort());  
  
 while(true) {  
 Socket clientSocket = null;  
 try {  
 clientSocket = serverSocket.accept();  
 System.*out*.println(who +": Accepting connection from " + clientSocket.getRemoteSocketAddress());  
  
 LinkCostProtocol.*recvAll*(clientSocket, routerMap);  
  
 LinkCostProtocol.*printLinkCost*(routerMap);  
  
 clientSocket.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (clientSocket != null) {  
 clientSocket.close();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 }  
  
 } catch (IOException e) {  
 System.*out*.println("Failed to bind to "+router1.getPort());  
 }  
 }  
}