```
import java.net.*;
import java.util.*;
import java.io.*;


public class StorageNode_TCP extends Thread {
		ServerSocket TCPServiceSocket;
		String predIP, succIP, myIP; //indirizzi ip predecessore, successore ed il mio
		int predPort, succPort, myPort, myId; //come sopra, solo che fanno riferimento a porte TCP
		int node_number; //serve per capire se questo è il primo nodo ad esser stato creato o no
		Hashtable<Integer, String> dataTable; //la tabella dei dati che detiene il nodo in questione
		Hashtable<Integer, String> nodeTable; //la tabella dei nodi (inizializzata con oggetto multicast)
		ScannerPorte scanner;
		StorageNode_Multicast multicast; //oggetto che gestisce il multicast
		Thread chiamante; //il thread che richiama il metodo. Esso rimarrà in stato di join fino ad un
nostro segnale di interrupt.
		//in questo modo garantiamo che esso non rilasci la lock acquisita fino al quando non siamo
"operativi" come nodo
		StorageNode_TCPMonitor TCPM; //questo monitor serve per poter gestire richieste concorrenti nei
confronti del TCP di questo nodo
		StorageNode_TCPWriter TCPR; //ricopre il ruolo di Reader nel modello Readers & Writer del TCPM
		List<Socket> list; //le socket generate dagli accept del TCPR andranno aggiunte a questa lista
		int network; //settata a 0 se utilizzata la versione locale, 1 se utilizzata la versione di rete
		public StorageNode_TCP
		(Thread chiamante, ServerSocket TCPServiceSocket, String predIP, String succIP, String fakeip, int
predPort, int succPort, int myId, Hashtable<Integer, String> dataTable, int node_number, StorageNode_Multicast
multicast, ScannerPorte scanner, int network) {
			this.chiamante = chiamante;
			this.TCPServiceSocket = TCPServiceSocket;
			this.predIP = predIP;
			this.succIP = succIP;
			this.predPort = predPort;
			this.succPort = succPort;
			this.myId = myId;
			this.node_number = node_number;
			this.dataTable = dataTable;
			this.multicast = multicast;
			this.scanner = scanner;
			//ottengo il mio indirizzo ip
			int end = fakeip.indexOf("UDPport=");
			this.myIP = fakeip.substring(0, end);
			int start = fakeip.indexOf("TCPport=");
			//ottengo la mia porta TCP
			this.myPort = Integer.parseInt(fakeip.substring(start + 8, fakeip.length()));
			this.list = new ArrayList();
			this.TCPM = new StorageNode_TCPMonitor();
			this.TCPR = new StorageNode_TCPWriter(TCPM, this.TCPServiceSocket, list,
this.TCPServiceSocket.getLocalPort());
			this.network = network;
		}

		public void run() { //se non sono il primo nodo, contatto il mio successore

			if (node_number != 1)
				try {
					Socket socketSucc = new Socket();

					if (network == 0)
						socketSucc.connect(new InetSocketAddress("localhost",
succPort));
					else
						socketSucc.connect(new InetSocketAddress
(InetAddress.getByName(succIP), succPort));
```

```
                                        BufferedReader reader = new BufferedReader(new InputStreamReader
(socketSucc.getInputStream()));

                                        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter
(socketSucc.getOutputStream()));

                                        //comunico al mio successore il mio id, in modo che possa capire
quali dati inviarmi
                                        writer.write("nuovo nodo");

                                        writer.newLine();

                                        writer.write(myId);

                                        writer.flush();

                                        String data;

                                        int id_data;

                                        do {
                                                id_data = reader.read();
                                                data = reader.readLine();

                                                if (id_data != -1 && data != null)
                                                        dataTable.put(id_data, data);

                                        } while (id_data != -1);

                                        writer.close();

                                        reader.close();

                                } catch (Exception e) {
                                        e.printStackTrace();
                                }

                        chiamante.interrupt(); //ora che sono operativo, posso dire al Thread chiamante di
continuare
                        TCPR.start(); //attivo il TCPReader, il quale si occuperà di accettare le richieste in
ingresso ed aggiungerle in list
                        //con la sleep garantiamo che la stampa dei contenuti della mia tabella dati non
interferisca con quella degli altri nodi

                        try {
                                Thread.sleep(50*node_number);

                        } catch (InterruptedException e) {
                                e.printStackTrace();
                        }

                        System.out.println("Elementi presenti nel nodo con id=" + myId);
                        Enumeration <Integer> items = dataTable.keys();

                        while (items.hasMoreElements()) {
                                int idobj = items.nextElement();
                                System.out.println("id=" + idobj + " data=" + dataTable.get(idobj));
                        }

                        //rimango in ascolto
                        while (true) {
                                try { //tento di accedere alla lista delle richieste
                                        TCPM.StartRead();
```

```
                                        //scorro la lista delle richieste e soddisfo ciascuna di
esse

                                        for (int i = 0;i < list.size();i++) {
                                                Socket socket = list.get(i);
                                                BufferedReader reader = new BufferedReader
(new InputStreamReader(socket.getInputStream()));
                                                //se mi contatta un nuovo nodo devo aprire
anche uno stream in uscita

                                                if (reader.readLine().equals("nuovo nodo"))
{
                                                        BufferedWriter writer = new
BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                                                        int id_node = reader.read
();
                                                        System.out.println("suo
id=" + id_node);
                                                        System.out.println("mio
id=" + myId);
                                                        nodeTable =
multicast.getTable(); //ottengo la tabella aggiornata dei nodi presenti sull'anello
                                                        items = nodeTable.keys();
                                                        int min = items.nextElement
(), actual;

                                                        while
(items.hasMoreElements()) {
                                                                actual =
items.nextElement();

                                                                if (actual
< min)
                                                                        min
= actual;
                                                        }

                                                        items = dataTable.keys();

                                                        while
(items.hasMoreElements()) {
                                                                int id_data
= items.nextElement();
                                                                //casi
possibili in cui "cedo" i dati della mia tabella al mio nuovo predecessore:
                                                                //1. Il
nuovo nodo è il più grande nodo dell'anello, mentre io sono il più piccolo
                                                                //2. Caso
normale: l'id del nuovo nodo è compreso tra l'id del dato e il mio id
                                                                //3. Il
nuovo nodo è il più piccolo nodo dell'anello, quindi gli passo i "dati più grandi dell'anello"

                                                                if ((myId
< id_node && myId < id_data && id_data < id_node) || id_data < id_node && id_node < myId || id_node == min &&
id_data > myId) {
String data = dataTable.get(id_data);
writer.write(id_data);
writer.write(data);
writer.newLine();
```

```
writer.flush();
//cancello dalla mia tabella il dato che ho trasmesso al nuovo nodo
dataTable.remove(id_data);
                                                                                    }
                                                                            }
                                                                    //id_data=-1 significa che
non ci sono altri dati
                                                                    writer.write( -1);
                                                                    writer.flush();
                                                                    reader.close();
                                                                    writer.close();
                                                                    //stampa dataTable
aggiornata
                                                                    Thread.sleep(50
*node_number);
                                                                    System.out.println
("Elementi presenti nel nodo con id=" + myId);
                                                                    items = dataTable.keys();
                                                                    while
(items.hasMoreElements()) {
                                                                                int idobj =
items.nextElement();
System.out.println("id=" + idobj + " data=" + dataTable.get(idobj));
                                                                            }
                                                                    }
                                                            //se mi contatta un client allora non
necessito dello stream in uscita
                                                            //in realtà a contattarmi non è
direttamente il client
                                                            //ma una socket TCP creta dall'oggetto che
gestisce l'UDP del nodo di bootstrap che si è occupato del client
                                                            else {
                                                                    items = dataTable.keys();
                                                                    int id_data = reader.read
(); //leggo l'identificatore del dato che vuole il client
                                                                    System.out.println("Qui
parla il noddo " + myId + ", cerco il dato " + id_data + " nella mia tabella");
                                                                    String ipClient =
reader.readLine(); //leggo l'indirizzo del client
                                                                    String risposta; //stringa
su cui risponderò tramite UDP
                                                                    int start =
ipClient.indexOf(" UDPport="), UDPportClient;
                                                                    UDPportClient =
Integer.parseInt(ipClient.substring(start + 9)); //ottengo la porta UDP del Client
                                                                    ipClient =
ipClient.substring(0, start); //ed il suo indirizzo ip
                                                                    //controllo che il dato
sia presente nella tabella
                                                                    if (dataTable.containsKey
```

```
(id_data))
                                                                risposta =
dataTable.get(id_data);
                                                        else
                                                                risposta = "Dato
inesistente";

                                                        //rispondo al client
direttamente su UDP
                                                        DatagramSocket
UDPServiceSocket = scanner.UDPSocket();

                                                        ByteArrayOutputStream bout
= new ByteArrayOutputStream( );

                                                        DataOutputStream dos = new
DataOutputStream(bout);

                                                        dos.writeUTF(risposta);

                                                        byte [ ] data =
bout.toByteArray();

                                                        InetAddress ia;

                                                        if (network == 0)
                                                                ia =
InetAddress.getByName("localhost");
                                                        else {
                                                                        ia =
InetAddress.getByName(ipClient);

                                                                }

                                                        DatagramPacket dp = new
DatagramPacket(data, data.length, ia, UDPportClient);
                                                        UDPServiceSocket.send(dp);
                                                        UDPServiceSocket.close();
                                                } //fine else
                                        } //fine for for(int i=0;i<list.size();i++)

                                //chiudo tutte le socket della lista e le rimuovo da essa
                                if (list.size() != 0)
                                        for (int i = list.size() - 1;i >= 0;i--) {
                                                list.get(i).close();
                                                list.remove(i);
                                        }

                                //posso rilasciare la lock
                                TCPM.EndRead();
                        } //fine try
                catch (Exception e) {
                                e.printStackTrace();        Scanner scanner = new Scanner
(System.in);scanner.nextInt();
                        }
                } //fine while(true)
        }
}
```