```java
//Programma che riceve una stringa su socket TCP, la inoltra su porta seriale, attende la risposta dalla seriale
// e inoltra la stringa ricevuta (di risposta dalla seriale) indietro sul socket TCP (ossia al client che si è connesso via TCP).

//Importazione Librerie
import gnu.io.*; //libreria per utilizzo della comunicazione seriale attraverso libreria RXTXlib-java
import java.io.*;
import java.util.*;

//Librerie per la parte del server TCP
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class RaspArdu {
    //parte SERIALE
    private static InputStream inStream;
    private static OutputStream outStream;
    //parte TCP
    private static int port;
    private static ServerSocket serverSocket;

    public static void main(String[] args) {

        //SERVIZIO TCP
        //Imposto la porta di ascolto del socket TCP
        //NB: sul Raspberry (ovvero notoriamente) le porte fino alla 1024 sono "riservate" al S.O.
        //allora scelgo la prima porta "libera" utile
        port = 1025;
        System.out.println("Port: " + port);
        //Socket di ricezione delle comunicazioni TCP
        Socket socket = null;

        try {
            //Inizializzo il socket TCP mettendolo in ascolto sulla porta precedentemente impostata
            serverSocket = new ServerSocket(port);

            while(true){
                //Accetto (l'eventuale) connessione in arrivo dal Client
                socket = serverSocket.accept();
                //Struttura per leggere i dati (quindi la stringa) inviata dal Client tramite il socket TCP
                Scanner scanner = new Scanner(socket.getInputStream());
                //Struttura per rispondere successivamente al Client via socket TCP
                PrintWriter printWriter = new PrintWriter(socket.getOutputStream(), true);

                //Catturo in una Stringa quanto ricevuto dal Client TCP
                String line = scanner.nextLine();

                System.out.println("received: " + line);

                //QUI VALUTO QUANTO RICEVUTO SU TCP
                //Ossia ricevo e mando direttamente alla Seriale
                //Però la seriale non accetta stringhe, quindi la scompongo in un array di byte (caratteri)
                byte[] buffer = line.getBytes();

                System.out.println("Connecting to serial...");

                //Richiamo la funzione successiva di Gestione della Comunicazione Seriale
                // e memorizzo la risposta nella Stringa "ris_serial"
                String ris_serial = gestioneSeriale(buffer);
                System.out.println("Serial reply: " + ris_serial);

                //RISPONDO AL CLIENT SUL SOCKET TCP
                //ovviamente inviandogli la risposta ricevuta dalla seriale e contenuta nella stringa "ris_serial"
```

```
64                  printWriter.println(ris_serial);
65
66                  //INFINE "chiudo" tutte le strutture precedentemente create
67                  // ossia: Struttura per ricezione dati su socket TCP; Struttura per
                    invio dati su socket TCP;
68                  // Struttura del socket TCP
69                  scanner.close();
70                  printWriter.close();
71                  socket.close();
72              }
73          } catch (IOException ex) {
74              ex.printStackTrace();
75          }finally {
76              //Istruzioni da eseguire comunque, anche in caso di eventuale errore
77              //Se il socket è eventualmente ancora aperto e/o attivo provvedo alla
                chiusura
78              if (socket != null) {
79                  try {
80                      socket.close();
81                  } catch (IOException ex) {
82                      ex.printStackTrace();
83                  }
84              }
85          }
86
87      } //Fine dalla funzione principale main
88
89      //FUNZIONE PER GESTIONE COMUNICAZIONE SERIALE
90      public static String gestioneSeriale(byte[] buffer){
91          //Stringa per ritorno dati al main
92          String reading = "";
93
94          try {
95              //Struttura per identificazione della Porta Seriale
96              //NB: *unix like le porte seriali si identificano come "/dev/tty" + nome
                porta impostata dal S.O.
97              // in questo caso il Raspberry ha chiamato la porta "S0"
98              CommPortIdentifier portId =
                CommPortIdentifier.getPortIdentifier("/dev/ttyS0");
99              SerialPort serialPort = (SerialPort) portId.open("PROVA application",
                5000);
100             //Parametri di configurazione della Porta Seriale
101             //Nel caso seguente si ha:
102             //  - Velocità di collegamento: 115200
103             //  - Data Bits: 8 (ossia si invia un byte alla volta)
104             //  - Controllo di Flusso: none (ossia disabilitato)
105             //  - Timeout: null (nessun timeout impostato per questa comunicazione
106             serialPort.setSerialPortParams(115200, SerialPort.DATABITS_8,
107                  SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
108             serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
109             serialPort.enableReceiveThreshold(1);
110             serialPort.disableReceiveTimeout();
111
112             //Struttura per leggere i dati ricevuti dalla seriale
113             inStream = serialPort.getInputStream();
114             //Struttura per inviare i dati verso la seriale
115             outStream = serialPort.getOutputStream();
116
117             System.out.println("INVIO a serial...");
118             //Invio il buffer dati (passati dal main a questa funzione) alla seriale
119             outStream.write(buffer, 0, buffer.length);
120             //Provvedo a "chiudere" subito la struttura di invio verso seriale
121             outStream.close();
122
123             System.out.println("RICEVO da serial...");
124
125             //Buffer di byte (provvisorio) di ricezione dalla seriale
126             byte[] buffer2 = new byte[1024];
127             int len = 0;
128             int data;
129             //Leggo dalla seriale (come impostato precedentemente 1 byte alla volta),
130             // finchè la funzione di READ non ritorna il carattere speciale "-1",
                ossia i dati da ricevere sono terminati
```

```
131             while ( ( data = inStream.read()) > -1 )
132             {
133                 //Controllo interno di ricezione: se ricevo il carattere speciale
                    "\n"
134                 //termino forzatamente la ricezione tramite l'istruzione "break"
135                 if ( data == '\n' ) {
136                     break;
137                 }
138                 //Byte per byte riempio il buffer provvisorio di ricezione
139                 //NB: notare il casting del singolo dato ricevuto a
                    byte
140                 buffer2[len++] = (byte) data;
141             }
142             System.out.print(new String(buffer2,0,len));
143             //Inserisco nella stringa di ritorno l'array di byte ricevuto dalla
                seriale
144             //convertendolo direttamente in Stringa (notare il new String(......)
145             reading = new String(buffer2,0,len);
146 
147             //Provvedo a "chiudere" la struttura di ricezione da seriale e la
                struttura dati della Porta Seriale
148             inStream.close();
149             serialPort.removeEventListener();
150             serialPort.close();
151 
152             System.out.println("valore di ritorno: " + reading);
153 
154             //Ritorno quanto ricevuto da seriale alla funzione main
155             return reading;
156 
157         } catch (Exception ex) {
158                 ex.printStackTrace();
159                 return reading;
160         }
161     } //Fine funzione di gestione della comunicazione Seriale
162 } //Fine della Classe RaspArdu
```