

```
/* Svolgimento Compito 4 - 21/12/05 */
```

```
import java.io.BufferedReader;
```

```
public class Client {
```

```
    public static void main(String[] args) throws IOException {
```

```
        InetAddress hostServerStream = null;
```

```
        int portServerStream = -1;
```

```
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
        String req;
```

```
        Socket ssocket;
```

```
        DataInputStream inSock;
```

```
        DataOutputStream outSock;
```

```
        try {
```

```
            if (args.length == 2) {
```

```
                hostServerStream = InetAddress.getByName(args[0]);
```

```
                // Controllare anche porta!!
```

```
                portServerStream = Integer.parseInt(args[1]);
```

```
            } else {
```

```
                System.out
```

```
                    .println("Usage: java Client hostServerStream portServerStream");
```

```
                System.exit(1);
```

```
            }
```

```
        } //try
```

```
        // Per esercizio si possono dividere le diverse eccezioni
```

```
        catch (Exception e) {
```

```
            System.out.println("Problemi, i seguenti: ");
```

```
            e.printStackTrace();
```

```
            System.out
```

```
                .println("Usage: java Client hostServerStream portServerStream");
```

```
            System.exit(2);
```

```
        }
```

```
        System.out.println("Esecuzione del servizio, ctrl+c per terminare");
```

```
        System.out
```

```
            .print("Premi V per la visualizzazione delle stanze, S per la sospensione di  
una stanza (ctrl+z exit): ");
```

```
        while ((req = br.readLine()) != null) {
```

```
            if (!req.equals("V") && !req.equals("S")) {
```

```
                System.out.println("Errore di sintassi");
```

```
                System.out
```

```
                    .print("Premi V per la visualizzazione delle stanze, S per la soppressione  
di una stanza (ctrl+c exit): ");
```

```
                continue;
```

```
            }
```

```
        // creazione socket
```

```
        try {
```

```
            ssocket = new Socket(hostServerStream, portServerStream);
```

```
            // setto il timeout per non bloccare indefinitivamente il client
```

```
            ssocket.setSoTimeout(30000);
```

```
            System.out.println("Creata la socket: " + ssocket);
```

```
            inSock = new DataInputStream(ssocket.getInputStream());
```

```
            outSock = new DataOutputStream(ssocket.getOutputStream());
```

```
        }
```

```
        catch (IOException ioe) {
```

```
            System.out.println("Problemi nella creazione degli stream su socket: ");
```

```
            ioe.printStackTrace();
```

```
            System.out.print("\n^D(Unix)/^Z(Win)+invio per uscire,"
```

```
                + " solo invio per continuare: ");
```

```
            // il client continua l'esecuzione riprendendo dall'inizio del ciclo
```

```
            continue;
```

```

}

/* Servizio di (V)isualizzazione */
if (req.equals("V")) {
    int numStanze;
    String stanzaCorr;

    // utilizzo socket
    try {
        // invio il tipo del servizio
        outSock.writeUTF(req);

        // chiudo l'output
        ssocket.shutdownOutput();

        // leggo il numero di stanze da visualizzare
        numStanze = inSock.readInt();

        // leggo le righe della matrice le le stampo a video
        for (int i = 0; i < numStanze; i++) {
            stanzaCorr = inSock.readUTF();
            System.out.println(stanzaCorr);
        }

        // libero le risorse
        ssocket.shutdownInput();
    }
    catch (IOException ioe) {
        System.out
            .println("Problemi nella creazione degli stream su socket: ");
        ioe.printStackTrace();
        System.out.print("\n^D(Unix)/^Z(Win)+invio per uscire,"
            + " solo invio per continuare: ");
        // il client continua l'esecuzione riprendendo dall'inizio del ciclo
        continue;
    }
    catch (Exception e) {
        System.out.println("Problemi nella creazione della socket: ");
        e.printStackTrace();
        System.out.print("\n^D(Unix)/^Z(Win)+invio per uscire,"
            + " solo invio per continuare: ");
        // il client continua l'esecuzione riprendendo dall'inizio del ciclo
        continue;
    }
}

System.out.println("\n\n---FINE RICEZIONE STANZE---\n\n");

/* Servizio di (S)ospensione */
} else {
    System.out.print("Inserisci il nome della stanza da sospendere: ");
    String stanza = br.readLine();
    int risposta;

    // utilizzo socket
    try {
        // invio il tipo del servizio
        outSock.writeUTF(req);

        // invio il nome della stanza da sospendere
        outSock.writeUTF(stanza);
        // non devo piu' scrivere; chiudo la socket in output
        ssocket.shutdownOutput();

        // leggo la risposta
        risposta = inSock.readInt();
    }
}

```

```

        // libero le risorse
        ssocket.shutdownInput();
    }
    catch (IOException ioe) {
        System.out
            .println("Problemi nella creazione degli stream su socket: ");
        ioe.printStackTrace();
        System.out.print("\n^D(Unix)/^Z(Win)+invio per uscire,"
            + " solo invio per continuare: ");
        // il client continua l'esecuzione riprendendo dall'inizio del ciclo
        continue;
    }
    catch (Exception e) {
        System.out.println("Problemi nella creazione della socket: ");
        e.printStackTrace();
        System.out.print("\n^D(Unix)/^Z(Win)+invio per uscire,"
            + " solo invio per continuare: ");
        // il client continua l'esecuzione riprendendo dall'inizio del ciclo
        continue;
    }
    }

    if (risposta == 0) {
        System.out.println("Stanza soppressa");
    } else {
        System.out.println("Errori nella soppressione...");
    }
} // else

System.out
    .print("Premi V per la visualizzazione delle stanze, S per la soppressione di
una stanza (ctrl+c exit): ");

    } //while
} //main

} //class Client

```