

```
/* Svolgimento Compito 1, 3, 5, 7 - 21/12/05 */
```

```
import java.rmi.*;
```

```
public class Server extends UnicastRemoteObject implements RemOp {  
    private static final int N = 10;
```

```
    static Stanza s[] = null;
```

```
    public Server() throws RemoteException {  
        super();  
    }
```

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

```
        s = new Stanza[N];
```

```
        for (int i = 0; i < N; i++) {
```

```
            s[i] = new Stanza();
```

```
        }
```

```
        final int REGISTRYPORT = 1099;
```

```
        String registryHost = "localhost";
```

```
        String serviceName = "ServerRMI";
```

```
        // Impostazione del SecurityManager
```

```
        if (System.getSecurityManager() == null) {
```

```
            System.setSecurityManager(new RMISecurityManager());
```

```
        }
```

```
        // Registrazione del servizio RMI
```

```
        String completeName = "/" + registryHost + ":" + REGISTRYPORT + "/"  
            + serviceName;
```

```
        try {
```

```
            Server serverRMI = new Server();
```

```
            Naming.rebind(completeName, serverRMI);
```

```
            System.out.println("Server RMI: Servizio \"" + serviceName  
                + "\" registrato");
```

```
        }
```

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

```
            System.err.println("Server RMI \"" + serviceName + "\": "  
                + e.getMessage());
```

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

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

```
        }
```

```
    }
```

```
    public synchronized boolean aggiungi_stanza(String nomeStanza, String tipo)  
        throws RemoteException {
```

```
        int indicePrimaLibera = -1;
```

```
        // controllo che non esista un'altra stanza con lo stesso nome
```

```
        for (int i = 0; i < N; i++) {
```

```
            if (s[i].getNome().equals(nomeStanza))
```

```
                return false;
```

```
            if (s[i].getNome().equals("L") && indicePrimaLibera == -1)  
                indicePrimaLibera = i;
```

```
        }
```

```
        if (indicePrimaLibera == -1)
```

```
            // non ci sono posizioni libere, restituisco false
```

```
            return false;
```

```
        else {
```

```
            s[indicePrimaLibera].setNome(nomeStanza);
```

```
            s[indicePrimaLibera].setStato(tipo);
```

```
            return true;
```

```
        }
```

```
    }
```

```
    public synchronized Stanza[] elimina_utente(String nomeUtente)
```

```

        throws RemoteException {
    Stanza temp[] = new Stanza[N];
    boolean isUtente = false;
    int cont = 0;
    for (int i = 0; i < N; i++) {
        if (s[i].rimuovi_utente(nomeUtente)) {
            temp[cont] = s[i];
            cont++;
            isUtente = true;
        }
    }
    // se l'utente non c'e' restituisco un puntatore a null
    if (!isUtente)
        return null;
    else {
        Stanza risultato[] = new Stanza[cont];
        // copio le sole stanze liberate
        for (int i = 0; i < cont; i++)
            risultato[i] = temp[i];
        return risultato;
    }
}

public synchronized boolean aggiungi_utente(String nomeStanza,
    String nomeUtente) throws RemoteException {
    int indiceStanza = -1;
    // controllo che esista la stanza alla quale si vuole aggiungere l'utente
    for (int i = 0; i < N; i++)
        if (s[i].getNome().equals(nomeStanza)) {
            indiceStanza = i;
            break;
        }
    if (indiceStanza >= 0)
        return s[indiceStanza].aggiungi_utente(nomeUtente);
    else
        return false; // la stanza non e' presente
}

public synchronized boolean elimina_stanza(String nomeStanza)
    throws RemoteException {
    int indiceStanza = -1;
    // controllo che esista la stanza che si vuole cancellare
    for (int i = 0; i < N; i++)
        if (s[i].getNome().equals(nomeStanza)) {
            indiceStanza = i;
            break;
        }
    if (indiceStanza >= 0) {
        s[indiceStanza].libera();
        return true;
    } else
        return false; // la stanza non e' presente
}

public Stanza[] visualizza_utente(String nomeUtente) throws RemoteException {
    Stanza temp[] = new Stanza[N];
    boolean isUtente = false;
    int cont = 0;
    for (int i = 0; i < N; i++) {
        if (s[i].is_utente(nomeUtente)) {
            temp[cont] = s[i];
            cont++;
            isUtente = true;
        }
    }
    // se l'utente non c'e' restituisco un puntatore a null

```

```
    if (!isUtente)
        return null;
    else {
        Stanza risultato[] = new Stanza[cont];
        // copio le sole stanze liberate
        for (int i = 0; i < cont; i++)
            risultato[i] = temp[i];
        return risultato;
    }
}
```