

## Aufgabe 5

### 5.1

HelloWorld Stub:

```
public interface Hello extends Remote {
    String sayHello() throws RemoteException;
}
```

Server:

```
public String sayHello() {
    return "Hello, world!";
}

public static void main(String args[]) {
    Server obj = new Server();
    Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
    // Bind the remote object's stub in the registry
    Registry registry = LocateRegistry.getRegistry();
    registry.rebind("Hello", stub);
    System.err.println("Server ready");
}
```

Client:

```
String host = (args.length < 1) ? null : args[0];
Registry registry = LocateRegistry.getRegistry(host);
Hello stub = (Hello) registry.lookup("Hello");
String response = stub.sayHello();
System.out.println("response: " + response);
```

### 5.2

Kalkulator Stub:

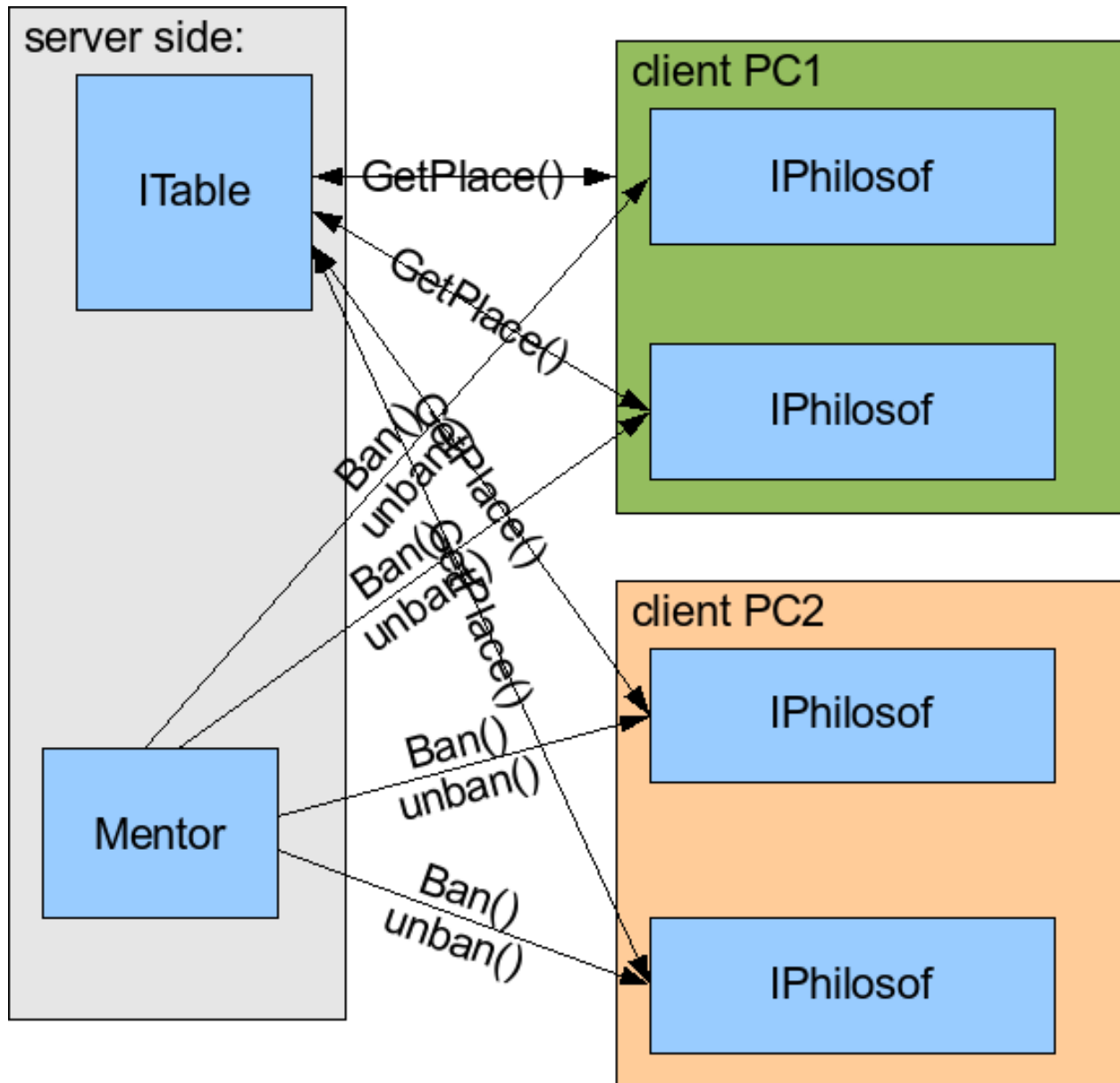
```
public interface Calculator extends Remote {
    float add(float i, float j) throws RemoteException;
    float sub(float i, float j) throws RemoteException;
    float mul(float i, float j) throws RemoteException;
    float div(float i, float j) throws RemoteException;
}
```

Client:

```
Registry registry = LocateRegistry.getRegistry(host);
Calculator stub = (Calculator) registry.lookup("Calculator");
while(true) {
```

```
String command = input.readLine();
float response = 0;
if (command.equalsIgnoreCase("exit") || command.equalsIgnoreCase("quit"))
    break;
String[] data = command.split(" ");
float i = Float.parseFloat(data[1]);
float j = Float.parseFloat(data[2]);
if (data[0].equalsIgnoreCase("add"))
    response = stub.add(i, j);
else if (data[0].equalsIgnoreCase("sub"))
    response = stub.sub(i, j);
else if (data[0].equalsIgnoreCase("mul"))
    response = stub.mul(i, j);
else if (data[0].equalsIgnoreCase("div"))
    response = stub.div(i, j);
System.out.println("Result: " + response);
}
```

### 5.3



### 5.4

Chat Stub for Server:

```
public interface Chat extends Remote {
    void say(String nickName, String msg) throws RemoteException;
    void registerForNotification(String nickName, Notifiable n) throws
    RemoteException;
    void removeSession(Notifiable n) throws RemoteException;
}
```

Notifiable Stub for client:

```
public interface Notifiable extends Remote {
    void onMessage(String nickname, String msg) throws RemoteException;
    String getNickName() throws RemoteException;
}
```

Every Client have to register itself on server:

```
public void registerForNotification(String nickName, Notifiable n)
    throws RemoteException {
    clientList.addElement(n);
    say("***", nickName+" joined server");
    System.out.println("registered");
}
```

And unregister if them will go out:

```
@Override
    public void removeSession(Notifiable n) throws RemoteException {
        String nickName = n.getNickName();
        clientList.remove(n);
        say("***", nickName+" leaved server");
        System.out.println("removed");
    }
```

To send message to other Clients registered on Server :

```
public void say(String nickName, String msg) throws RemoteException {
    System.out.println("say msg");
    for(Enumeration<Notifiable> clients = clientList.elements();
clients.hasMoreElements();) {
        Notifiable thingToNotify = clients.nextElement();
        if(!nickName.equalsIgnoreCase(thingToNotify.getNickName()))
            try {
                thingToNotify.onMessage(nickName, msg);
            } catch (RemoteException ex) {
                removeSession(thingToNotify);
            }
    }
}
```

Und Klient Seite:

```
Chat server;
boolean exit = false;
Registry registry = LocateRegistry.getRegistry(host);
server = (Chat) registry.lookup("Chat");
Client client = new Client(nickName, server);
while(!exit) {
    String command = input.readLine();
    if(command.equalsIgnoreCase("/exit") || command.equalsIgnoreCase("/quit")) {
        exit = true;
        break;
    }
}
```

```
        client.sendMessage (command) ;  
    }  
    server.removeSession (client) ;  
    client.close () ;
```

Klient OnMessage (call back from Server):

```
public void onMessage (String nickName, String msg) throws RemoteException {  
    System.out.println ("<"+nickName+">"+msg) ;  
}
```

PS. für eine Verschlüsselung könnte man einen AES symmetrisches Verfahren nehmen, wo alle Klienten beim starten noch einen Password eingeben müssen, mit dem die Nachrichten kodiert und dekodiert werden.