Aufgabenblatt 6

Chat-System mit Java RMI

Sie sollen ein Chat-System unter Verwendung von Java RMI implementieren. Das System besteht aus einem Server, an den beliebig viele Clients angeschlossen werden können. Das System soll nach bekanntem Prinzip arbeiten: Nachrichten werden auf einem Client eingegeben und über den Server an alle angeschlossenen Clients weitergeleitet.

Prinzipiell gibt es zwei Möglichkeiten, ein solches System zu realisieren:

1. Die Nachrichten werden vom Server vorgehalten. Verbundene Clients überprüfen in zyklischen Abständen das Vorliegen neuer Nachrichten und rufen diese ab (*Polling*).
2. Clients registrieren im Server jeweils ein sog. *Callback*-Objekt. Dieses Objekt realisiert Methoden zum Empfang und zur Anzeige von Nachrichten.

Verwenden Sie für Ihre Lösung den letztgenannten Ansatz. Es wird dazu vorgeschlagen, 3 Klassen von Remote-Objekten vorzusehen:

- Das Interface **`ChatServer`** stellt nach außen eine Methode zur Verfügung, mit der sich Clients beim Server unter Angabe eines Namens und einer Referenz auf das Client-Callback-Objekt (mit dem Remote-Interface **`ClientProxy`**) registrieren und de-registrieren können:

```
public interface ChatServer extends Remote {
    public ChatProxy subscribeUser (String username,
      ClientProxy handle) throws RemoteException;
    public boolean unsubscribeUser (String username)
        throws RemoteException;
}
```

- **`ClientProxy`** repräsentiert den Client im Server und unterstützt eine Methode zum Empfang von Nachrichten:

```
public interface ClientProxy extends Remote {
     public void receiveMessage (String username,
          String message) throws RemoteException;
}
```

- **`ChatProxy`** repräsentiert den Kommunikationskanal im Client und beinhaltet eine Methode zum Versenden von Nachrichten an den Server. Für jeden angemeldeten Client muss ein Remote-Objekte mit diesem Interface im Server realisiert werden:

```
public interface ChatProxy extends Remote {
    public void sendMessage (String message)
      throws RemoteException;
}
```

Die entsprechenden Implementierungsklassen realisieren die Schnittstellen-Methoden sowie weitere

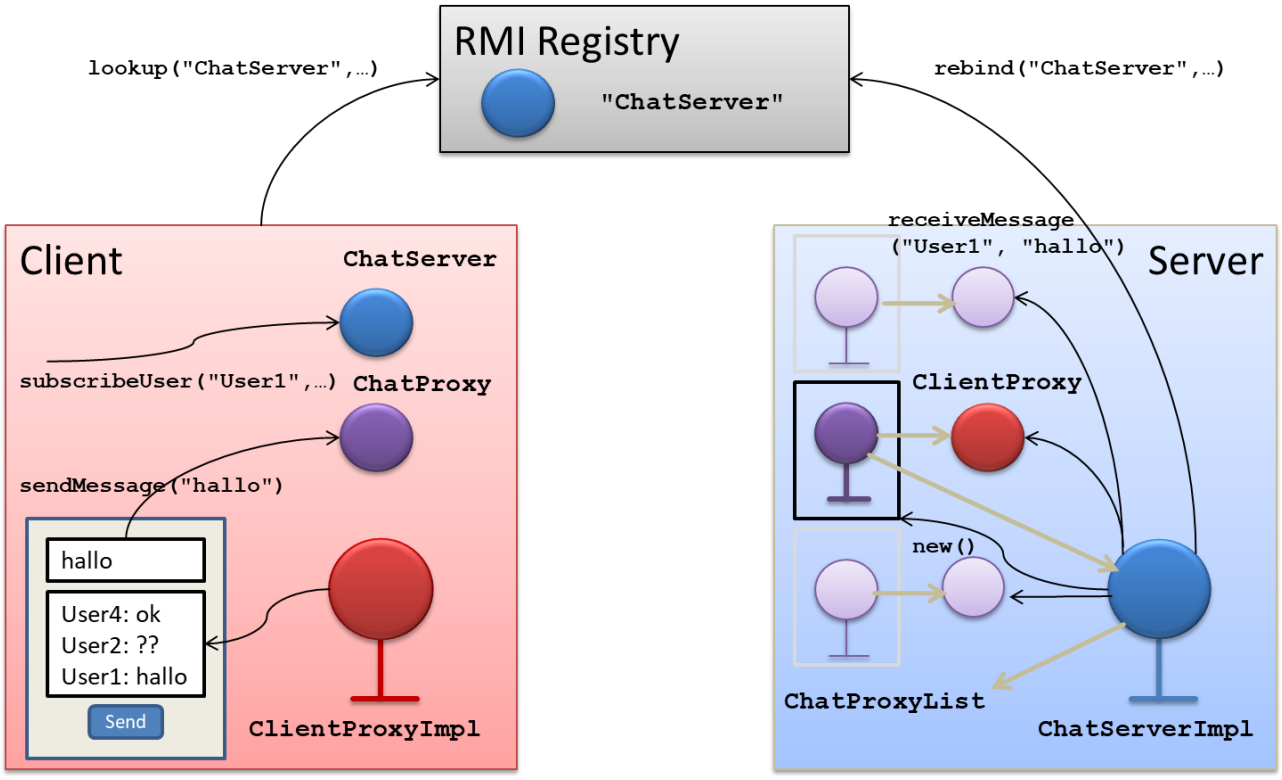
Methoden. Die Implementierung der Schnittstelle **`ChatServer`** erzeugt in der **`main()`** Routine ein neues **`ChatServerImpl`** Objekt und registriert das Interface in der RMI Registry unter einem frei wählbaren Namen. Daneben benötigen Sie noch einen Client (GUI oder interaktive Shell). Dieser liest eine Referenz auf das **`ChatServer`** Objekt aus der Registry aus und ermöglicht es einem User, sich unter Angabe eines Namens per **`subscribeUser()`** beim Chat zu registrieren.

Machen Sie sich zunächst den Ablauf klar: zeichnen Sie dazu ein Sequenzdiagramm des Ablaufs beim Registrieren eines Clients Ordnen Sie die Instanzen der Klassen folgendermaßen (von links nach rechts) an:

aClient: *ChatClient*	ClientProxyImpl: *ClientProxyImpl*	theRegistry: *Registry*	***Netz-werk***	theServer: *ChatServer*	theList: *ChatProxyList*	aChatProxy: *ChatProxy*

Überlegen Sie, welche Informationen über einen neu registrierten Client Sie im `ChatProxy` auf der Serverseite speichern müssen. Fügen Sie nach Erzeugung eines **`ChatProxy`** beim Server, diesen einer **`ChatProxyList`** zu.

Testen Sie Ihr System, in dem Sie einen Chat mit mehreren Clients aufsetzen.



Hinweis:

Für das Testat ist ein doppelseitiger Ausdruck (inkl. Namen der Gruppenmitglieder) der Dokumentation abzugeben.

Testierung: 9./10.5.2023