Distributed Systems
Lab 03

Chirag Vankar
201651013

Implement the Gossip program by using RMI.

We create two interfaces- one for client and one for server. Additionally we create driver program for both client and server.

Finally we create two more classes containing main methods.

ChatServerInterface has 2 methods- one for registering client and one to broadcast messages to chat clients.
ChatClientInterface has one method to retrieve the messages.

ChatServerInterface.java

```
import java.rmi.*;

public interface ChatServerInterface extends Remote{
   void registerChatClient(ChatClientInterface chatClient) throws RemoteException;
   void broadcastMessage(String message) throws RemoteException;
}
```

Create interface for Server

ChatClientInterface.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ChatClientInterface extends Remote
{
   void retrieveMessage(String message) throws RemoteException;
}
```

Create interface for Client

ChatServer.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

public class ChatServer extends UnicastRemoteObject implements ChatServerInterface
{
   public static final long serialVersionUID = 1L;
   private ArrayList<ChatClientInterface> chatClients;
   protected ChatServer() throws RemoteException{
       chatClients = new ArrayList<ChatClientInterface>();
   }

   public synchronized void registerChatClient(ChatClientInterface chatClient) throws
RemoteException{
           this.chatClients.add(chatClient);
   }

   public synchronized void broadcastMessage(String message) throws RemoteException{
       int i = 0;
       while(i< chatClients.size()){
           chatClients.get(i++).retrieveMessage(message);
       }
   }
}
```

This class extends UnicastRemoteObject and implements ChatServerInterface. ArrayList stores the list of clients and ArrayList is instantiated. In method registerChatClient We are adding chatClient instance to chatClients arraylist. In method broadcastMessage we go through all chat clients and calling retrieveMessage method passing the message. So each client receives the message which is broadcasted.

ChatServerDriver.java

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;


public class ChatServerDriver{
   public static void main(String args[]) throws RemoteException, MalformedURLException{
       Naming.rebind("RMIChatServer", new ChatServer());
   }
}
```

We call rebind method which binds the instance of ChatServer to RMIChatServer.

ChatClient.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Scanner;

public class ChatClient extends UnicastRemoteObject implements ChatClientInterface,
Runnable{
   private static final long   serialVersionUID = 1L;
   private ChatServerInterface chatServer;
   private String name = null;

   protected ChatClient(String name, ChatServerInterface chatServer) throws
RemoteException{
       this.name = name;
       this.chatServer = chatServer;
       chatServer.registerChatClient(this);
   }

   public void retrieveMessage(String message) throws RemoteException{
       System.out.println(message);
   }

   public void run(){
       Scanner scanner = new Scanner(System.in);
       String message;
       while(true)
```

```
        {
            message = scanner.nextLine();
            try{
                chatServer.broadcastMessage(name + ":" + message);

            }
            catch(RemoteException e)
            {e.printStackTrace(); }
        }
    }
}
```

ChatClient extends UnicastRemoteObject and implements ChatClientInterface. Define chatServer and name and set in constructor. Register chatClient with chatServer instance.

When chatClient receives message print the message using System.out.

Since we are executing instances of this class as threads we implement Runnable and run() method which reads text from command line and broadcasts them to clients.

ChatClientDriver.java

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

public class ChatClientDriver{
   public static void main(String args[]) throws MalformedURLException, RemoteException,
NotBoundException{
        String chatServerURL = "rmi://localhost/RMIChatServer";
        ChatServerInterface chatServer = (ChatServerInterface) Naming.lookup(chatServerURL);
        new Thread(new ChatClient(args[0], chatServer)).start();
    }
}
```

Since we have bounded chatServer to RMIChatServer so we set chat server URL. Use chatServer instance to lookup chatServerURL. Instantiate a thread and call start method on it. Args[0] is the name user enters on command line.

Compiling :

1. **Use javac *.java in working directory**
2. **rmic ChatServer**
3. **rmic ChatClient**
4. **rmiregistry**
5. **java ChatServerDriver**
6. **java ChatClientDriver Clientname1**
7. **java ChatClientDriver Clientname2**