CORBAChatApp.idl

```
module CORBAChatApp{
	interface CORBAChat{
		string connection(in string userName);
		void newMessages(in string roomName, in string message);
		string getMessages(in string roomName);
		void disconnect(in string userName, in string roomName);
	};
};
```

CORBAChatClient.java

```
import CORBAChatApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.util.*;
import java.io.*;
import java.util.regex.Pattern;

public class CORBAChatClient{
	static CORBAChat serverImpl;
	public static String lastMessage = "";
	public static String userName = "";
	public static String connectedRoom = "";
	public static String strResponse = "";

	public static void main(String []args){
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

		try{
			// create and initialize the ORB
			ORB orb = ORB.init(args, null);

			// get the root naming context
			org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");

			// Use NamingContextExt instead of NamingContext. This is part of the Interoperable naming
Service.
			NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

			// resolve the Object Reference in Naming
			String name="CORBAChat";
			serverImpl = CORBAChatHelper.narrow(ncRef.resolve_str(name));

			System.out.println("");
        System.out.print("Enter a name : ");
        String nameInput = br.readLine();
        System.out.println("");

			String connectionResponse = serverImpl.connection(nameInput);

			// get the response from the web server
```

```
        // if the response is 'failure'
        //    print a message saying the user needs to choose a different name
        // else, set the 'connectedRoom' to the default 'general'
        //    set the 'userName' to 'nameInput'
        //    print the 'connectedTime' to the terminal
                        if (connectionResponse.equals("failure")) {
                                System.out.println("Please choose a different name");
                        } else {
                                connectedRoom = "general";
                                strResponse = connectionResponse;
                                userName = nameInput;

                                String TimeStamp = new java.util.Date().toString();
                                String connectedTime = "Connected on " + TimeStamp;

                                System.out.println(connectedTime);
                                System.out.println("");

                                // get all the previously sent messages to this room and print it to the terminal
                                String[] strResponseParts = strResponse.split(Pattern.quote("|"));
                                for (int i = 1; i < strResponseParts.length - 1; i++) {
                                        String[] strArr = strResponseParts[i].split(" ", 2);
                                        System.out.println(strArr[1] + "\n");
                                }

// create a thread that will keep asking the server if there is a new message meant for the connected room every 500
milliseconds
                                Thread receivingMessages = new Thread(new Runnable() {
                                        public void run() {
                                                while (true) {
                                                        String serverResponse =
serverImpl.getMessages(connectedRoom);
                                                        if (!(serverResponse.equals(lastMessage)) &&
!(serverResponse.equals("")) && !(serverResponse.startsWith("@" + userName))) {
                                                                lastMessage = serverResponse;
                                                                System.out.println(serverResponse + "\n");
                                                        }
                                                        try {
                                                                Thread.sleep(500);
                                                        } catch (Exception e) {
                                                                System.out.println(e);
                                                        }
                                                }
                                        }
                                });
                                // start the receiving messages thread
                                receivingMessages.start();

                                while (true) {
                                        String input = br.readLine();
                                        System.out.println("");
```

```
					// if the 'input' starts with 'exit', disconnect the client by calling the remote
method 'disconnect'
					if (input.equals("exit")) {
						serverImpl.disconnect(userName, connectedRoom);
						System.exit(0);
					}

					// else send the message to all connected users by calling the remote method
'newMessages'
		// the message contain the room to send 'connectedRoom', the 'userName' and the message 'input'
					else {
						serverImpl.newMessages(connectedRoom, "@" + userName + ":" +
input);
					}
				}
			}
		}

		catch(Exception e){
			System.out.println("ERROR : "+e);
			e.printStackTrace(System.out);
		}
	}
}
```

CORBAChatServer.java

```
import CORBAChatApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import java.util.*;

public class CORBAChatServer{
	public static void main(String args[]){
		try{
			// create and initialize the ORB
			ORB orb=ORB.init(args,null);

			// get reference to rootpoa & activate the POAManager
			POA rootpoa=POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
			rootpoa.the_POAManager().activate();

			// create servant and register it with the ORB
			ServerImpl serverImpl=new ServerImpl();
			serverImpl.setORB(orb);

			// get object reference from the servant
			org.omg.CORBA.Object ref= rootpoa.servant_to_reference(serverImpl);
			CORBAChat href=CORBAChatHelper.narrow(ref);
```

```
                // get the root naming context
                org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");

                // Use NamingContextExt which is part of the Interoperable Naming Service (INS) specification.
                NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

                // bind the Object Reference in Naming
                String name="CORBAChat";
                NameComponent path[] = ncRef.to_name(name);
                ncRef.rebind(path,href);

                System.out.println("CORBAChatServer ready and waiting....");

                // wait for invocations from clients
                orb.run();

            }
            catch(Exception e){
                System.out.println("ERROR : "+e);
                e.printStackTrace(System.out);
            }
            System.out.println("CORBAChatServer Exiting...");
    }
}
class ServerImpl extends CORBAChatPOA {
    // create an ORB object
    private ORB orb;

    // initialize the ORB object
    public void setORB(ORB orb_val){
        orb=orb_val;
    }

    // list to store all sent messages log
    static List<String> messageLogs = new ArrayList<>();

    // list to store all users together with their connected rooms
    static List<String> roomUsers = new ArrayList<>();

    // list to store all users name
    private static List<String> names = new ArrayList<>();

    // list to store all available rooms, the default room is 'general'
    private static List<String> rooms = new ArrayList<String>() {
        {
            add("general");
        }
    };

    public String connection(String userName) {

        // create a 'StringBuilder sb' to store messages
        StringBuilder sb = new StringBuilder();
```

```
		// if names list already contains 'userName', return 'failure' message
	// else
	//	add 'userName' to 'names' list
	//	append default room name to 'userName' and add it to 'roomUsers' list
	//	create a new message indicating a new user is connected and add it to 'messageLogs' list
	//	get all the previously sent messages to this group and append it to a string builder (sb) object
	//	return a string builder (sb) object containing all previous messages in this group separated by | symbol
		if (names.contains(userName.toLowerCase())) {
			sb.append("failure");
		} else {
			String TimeStamp = new java.util.Date().toString();
			String connectedTime = "Connected on " + TimeStamp;
			names.add(userName);
			roomUsers.add("general " + userName);
			messageLogs.add("general @" + userName + " " + connectedTime);
			sb.append("joined");

			for (int i = 0; i < messageLogs.size(); i++) {
				if (messageLogs.get(i).startsWith("general")) {
					sb.append("|" + messageLogs.get(i));
				}
			}
		}
		return sb.toString();
	}
	// add a new message to the 'messageLogs' list, the new message contains 'roomName' followed by the
'message'
	public void newMessages(String roomName, String message) {
		messageLogs.add(roomName + " " + message);
	}

	// return the last message from 'roomName'
	public String getMessages(String roomName) {
		String valueToReturn = "";

		// get the last message from 'messageLogs'
		String message = messageLogs.get(messageLogs.size() - 1);

		// if the last message returned is for 'roomName', append it to 'valueToReturn'
		// otherwise, 'valueToReturn' will be null
		if (message.startsWith(roomName)) {
			valueToReturn = message.substring(message.indexOf(" ") + 1);
		}

		// return last message in 'roomName', or null if there is no new message for 'roomName'
		return valueToReturn;
	}

	// disconnect from the chat application
	public void disconnect(String userName, String roomName) {

		// remove 'userName' from 'names' list
```

```
                names.remove(userName);

                // send message to 'roomName' users indicating the user has left
                messageLogs.add(roomName + " " + userName + " has left");
        }
}
```

CMD Commands

```
idlj -fall CORBAChat.idl

javac *.java CORBAChatApp/*.java

start orbd -ORBInitialPort 1050

java CORBAChatServer -ORBInitialPort 1050 -ORBInitialHost localhost

java CORBAChatClient -ORBInitialPort 1050 -ORBInitialHost localhost
```