Distributed Systems

Chat application with a Centralized server

Raggi hosni - 144711

**Application of Choice**

A chat application with a centralized  server using Java CORBA.

***Why is it a good distributed application?***

The very basic idea of how chat applications work is that messages are exchanged between nodes (clients) for communication. This fact can make such applications be a very good example of distributed systems, especially with the increase of more features and number of nodes which in turn will increase the messages exchanged and the demand for implementing it in a distributed manner.

**Key features:**

***Architecture***

**-** The architecture is a **centralized** client/server architecture.

- The server side contains a container that has hashmaps of all users and rooms(advanced).

- User identification is token based, each user has a unique token generated by Java SecureRandom, and is stored in the Token:User hashmap on the server side.

***Technical features***

**Basic**- Sending messages.  
- Changing name

**Advanced**- Sending messages  
- Changing name  
- Creating rooms   
- Joining rooms  
- Viewing room list  
- Leaving rooms.

**Simple Application Design**

**Server Side**

***IDL File (Interface):***Contains the needed methods to be implemented

***User Class :*** - Queue of strings to represent the messages received by user in a FIFO manner  
- Synchronized accessors and mutators to user variables (token, name, messages)

***Message Class:***  
- Message variables (sender,content,date)

***Server Class:***  
- Acts as a container for all users.  
- Has a user counter to set default user names.  
- Has a random token generator for each user using Java SecureRandom  
- Hashmap of all users and their tokens as keys, **i.e:** Token:User  
- Synchronized methods

***Server Implementation Class:***  
 - Implementation of all the IDL (Interface) methods  
 - CORBA Server listening logic

**Client Side**

***Chat Client Class:***   
Contains 2 threads plus the main thread  
**Input thread :  
-** Initializes the input GUI along with the needed Jbuttons,Jlabels..etc  
**-** Contains an action listener to take input on Enter key press  
- Button and different GUI Elements actionListeners  
  
**Output thread:  
-** Initialized the output box and adds it to the global Jframe  
**-** Keep receiving messages (dequeuing from user messages) by calling proper method from server implementation instance.  
  
**Main thread:  
-** Starts the output and the input threads  
**-** CORBA Client connection logic

***Text Area Output Stream Class:***Used to transform the system.out to JTextArea to display in the GUI

**Advanced Application Design**

**Server Side**

***IDL File (Interface):***Contains the needed methods to be implemented

***User Class :***- Queue of strings to represent the messages received by user in a FIFO manner  
- Synchronized accessors and mutators to user variables (token,current room , name, messages)

***Message Class:***  
- Message variables (sender,content,date)

***Server Class:***  
- Acts as a container for all users.  
- Has a user counter to set default user names.  
- Has a random token generator for each user using Java SecureRandom  
- Hashmap of all users and their tokens as keys, **i.e:** Token:User  
- Hashmap of all rooms and their names as keys, **i.e:** roomName:Room  
- Synchronized methods  
  
***Server Implementation Class:***  
 Implementation of all the IDL (Interface) methods  
 CORBA Server listening logic

**Client Side**

***Chat Client Class:***   
Contains 2 threads plus the main thread  
**Input thread :  
-** Initializes the input GUI along with the needed Jbuttons,Jlabels..etc  
- Contains an action listener to take input on Enter key press  
- Button and different GUI Elements actionListeners  
  
**Output thread:**- Initialized the output box and adds it to the global Jframe  
- Keep receiving messages (dequeuing from user messages) by calling proper method from server implementation instance.  
  
**Main thread:  
-** Starts the output and the input threads  
- CORBA Client connection logic

***Text Area Output Stream Class:***Used to transform the system.out to JTextArea to display in the GUI

**References**

Implementation of TextAreaOutputStream **:** [http://www.codejava.net/java-se/swing/redirect-standard-output-streams-to-jtextarea](http://www.codejava.net/java-se/swing/redirect-standard-output-streams-to-jtextarea%20)