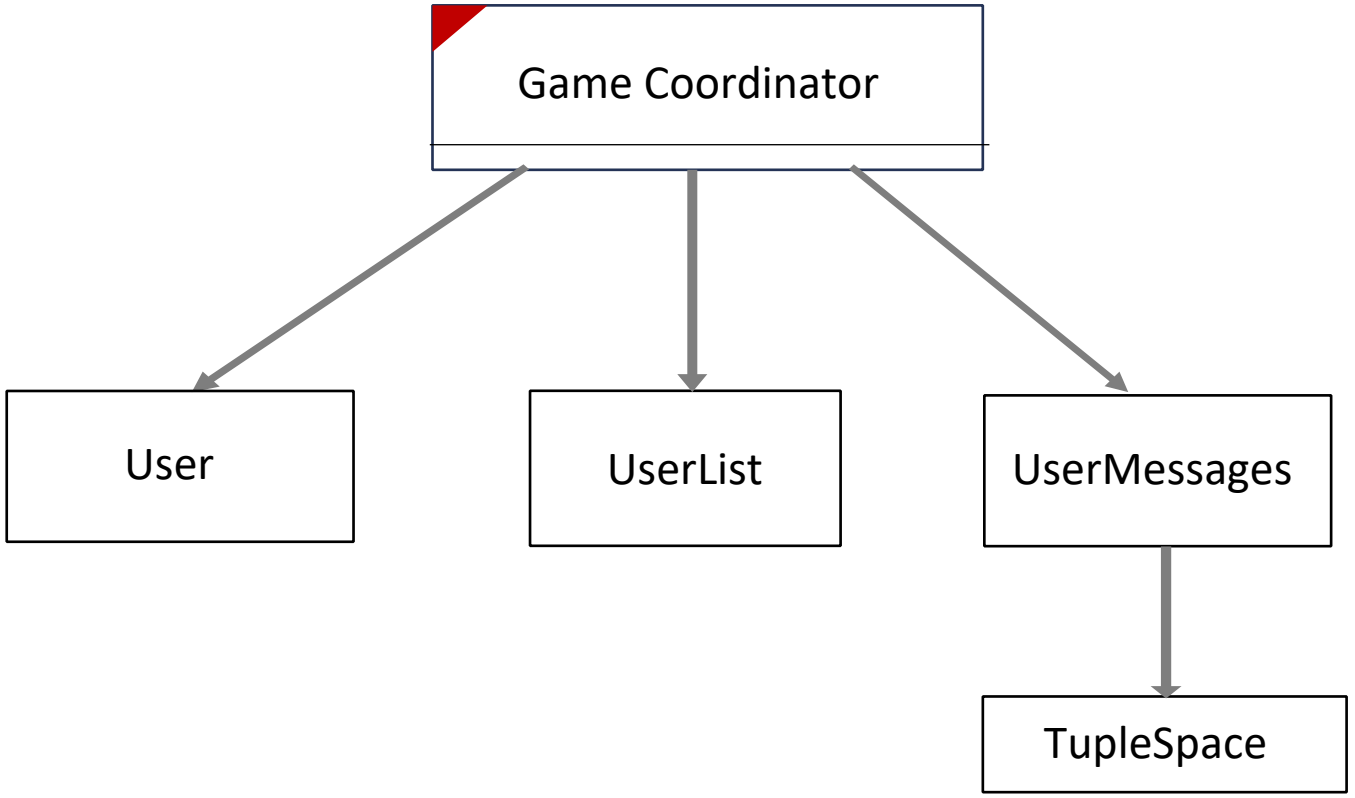
ChatRoom Design:



Design Details

Below are the methods explained that I used in User, UserList, UserMessages classes:

- **Note**: My TupleSpace is named (TrieSpace (efficient), LinkedSpace (naïve)). I have a TupleSpace interface, TrieSpace and LinkedSpace implements TupleSpace.
- Game Coordinator / Display is the main class. It has access to User, UserList, and UserMessages class.
  I. The way my design works is like User themselves have a name and whether or not they are active or not.
     - So, User has just getters and setters.
  II. UserList keeps track of all the users in the list, whether active or inactive.
     - UserList has methods to addUser(), getAllUsers() and getActiveUsers().

Design Details

- I have addUser() method because I want to store all the users so I can show them when the person using ChatRoom clicks “All” or “Active” button.
- getAllUsers() and getActiveUsers() becomes useful when the user clicks “All” , “Active” button, in game coordinator, they are called so it can display them.

III. UserMessages is a class which has access to TupleSpace.

- It has the functionality of adding a specific tuple to tupleSpace (in the case of ChatRoom App, the tuple it adds is < time, User, message >)
- It also has the functionality to be able to printLastTenMsgs() from the TupleSpace by keeping track of recent times in the list.

Design Details

- ChatRoom overall design implementation:
    a. Game Coordinator has display elements such as Vbox, Hbox, buttons, textFields, and there are certain number of actionListeners. The methods in User, UserList and UserMessages are used by the Coordinator.
    b. In setting up the GUI, those methods becomes useful for better user experience.
    c. WHY this design?
        - This design is good because there's encapsulation of TupleSpace (only UserMessages class knows about TupleSpace).
        - User can login/ logoff (user has Boolean field).

Design Details

- Users share information via shared tupleSpace, meaning in tupleSpace we have user's information (stored as tuple < time, **User,** message>) and if user1 typed in a message, and then the user2 typed; user1 can see user2's messages and vice versa.

- **Methods Used (Explained above):**
- **User:**
  I. getName();
  II. isActive();
  III. setActive(Boolean active);

Design Details

- **UserList:**
  I. addUser(User… user);
  II. getUsers();
  III. getActiveUsers();

- **UserMessages:**
  I. addMsg(String time, User user, String message);
  II. printLastTenMsgs();
    - This class has access to TupleSpace.