*Distributed Systems: Assignment 2*



*Multicast Handling System*

| **ID** | **Student Name** | **E-mail** |
| --- | --- | --- |
| **196280** | Ashraf Adel | ashraf196280@bue.edu.eg |

Table Of Contents

[**Introduction**](#_heading=h.tyjcwt) **3**

[**Design Decisions**](#_heading=h.3dy6vkm) **3**

[Chosen Task & Rules](#_heading=h.1t3h5sf) 3

[Design Visualisation](#_heading=h.4d34og8) 4

[Big Picture](#_heading=h.h9dvd12zez5p) 4

[The Output](#_heading=h.akf0wmbt945y) 5

[Multicast Handler Classes](#_heading=h.ln4mth44h9gl) 8

[Subscriber Classes](#_heading=h.pulteko3yl7) 9

[Design Mapping To Code](#_heading=h.q0utw68cew8s) 10

[Multicast Handler Interface (mhIF)](#_heading=h.roed5xih8o7o) 10

[Multicast Handler Class (mhC)](#_heading=h.nu9tei1h6i8g) 11

[Multicast Handler GUI (mhGUI)](#_heading=h.mewjia495b30) 13

[Multicast Handler Main (mhM)](#_heading=h.dqytqof41v94) 14

[Subscriber Interface (subIF)](#_heading=h.z0ypokl1scpp) 14

[Subscriber Class (subC)](#_heading=h.saehdiy3jdwr) 15

[Subscriber GUI (subGUI)](#_heading=h.czma3dowomos) 16

[Assumptions](#_heading=h.3j2qqm3) 21

[Advantages & Disadvantages of Multicast Handling System](#_heading=h.1y810tw) 21

# 

# Introduction

Multicast is a method of group communication where the sender (subscriber) sends data to multiple receivers or nodes (other subscribers) present in the network simultaneously using a multicast handler as a middle-man. Multicasting is a type of one-to-many and many-to-many communication as it allows sender or senders to send data packets to multiple receivers at once across LANs or WANs. The multicast handler is responsible for storing information about subscribers and their respective groups to know which messages should be multicasted to which specific groups.

# Design Decisions

## Chosen Task & Rules

The task that was chosen for this multicast system is creating a hub for students to communicate about different topics throughout the semester using different groups chat boxes; these groups are the following:

1. Study group
2. Materials group
3. Jokes group

Since the university doesn’t want students to overload its server (multicast handler), a student (subscriber) can only subscribe to one group at any given time. If they want to talk in another group, they would unsubscribe from the current group first, then subscribe to the new group.

Also, due to privacy issues, some students requested that they don’t appear to other students in a group unless they participate in the chat. Therefore, the students won’t be able to see the other students currently in the group when they are sending messages, but they will be able to see those students who respond in the chat.

However, the university wants to keep track of the students (subscribers) currently joined in a group, so the multicast handler GUI will display the list of students joined in each group in a queue manner (first student that joins will be displayed first, and so forth).

In addition, once a student subscribes to a group, they will not be able to change their name and id (even if they change the fields and press subscribe again, their name and id won’t change in the other GUIs).

Finally, to keep the student’s interface uncluttered, the only chatbox that will be displayed is the chatbox related to the group that the student is currently subscribed to.

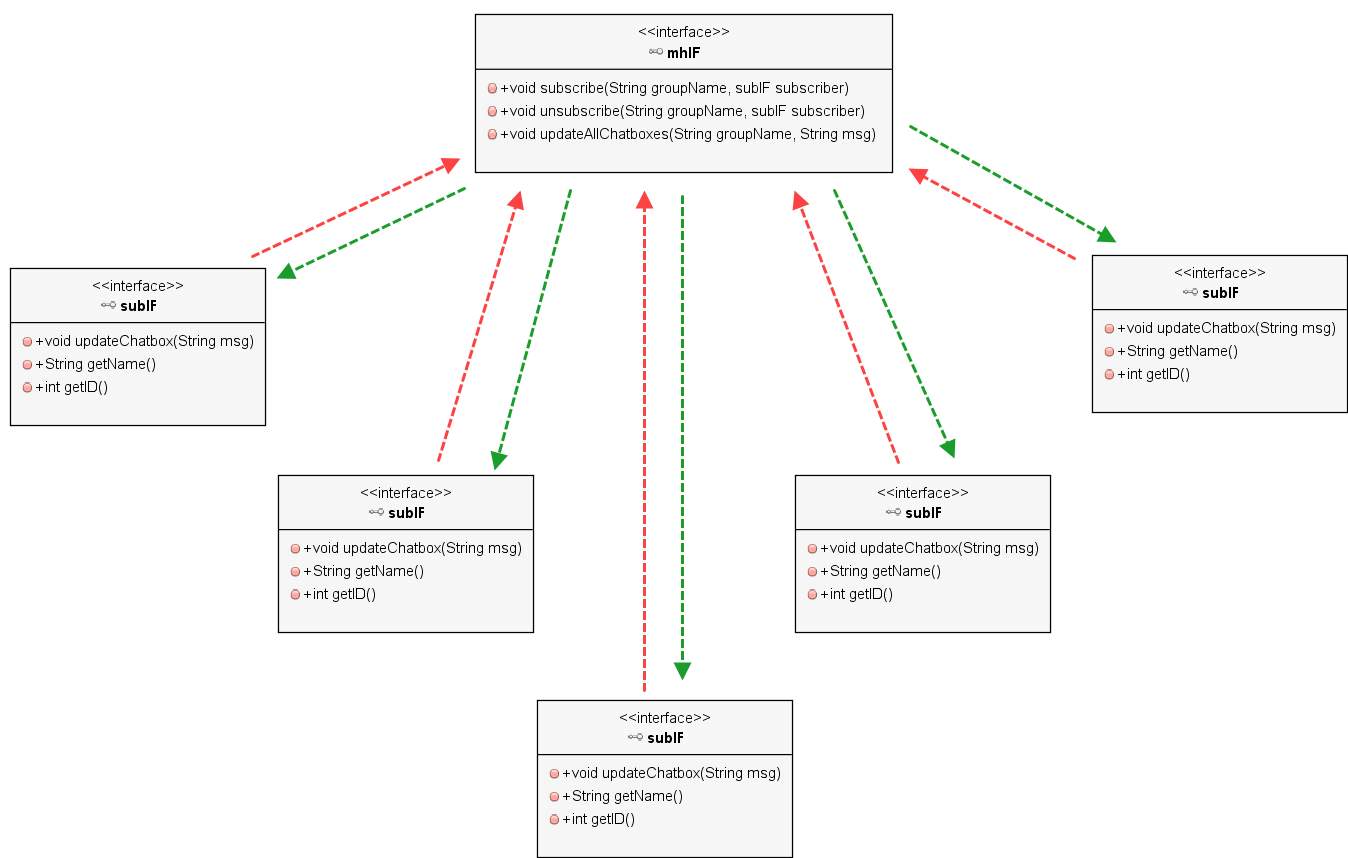
## Design Visualisation

**Update** notice: the code has been slightly modified such that there is an additional textArea for the announcments, where a subscriber can send a message and it will go to all other subscribers registered in the multihandler (in other groups).

Note that in the source code, there is only one subscriber java file (interface, remote class, GUI); that’s because it is the same code that will run no matter which subscriber is using the system, but in reality (and in the diagrams below), one can imagine that there is a java file for every subscriber device.

### Big Picture

Displaying the handler and five subscribers (students):



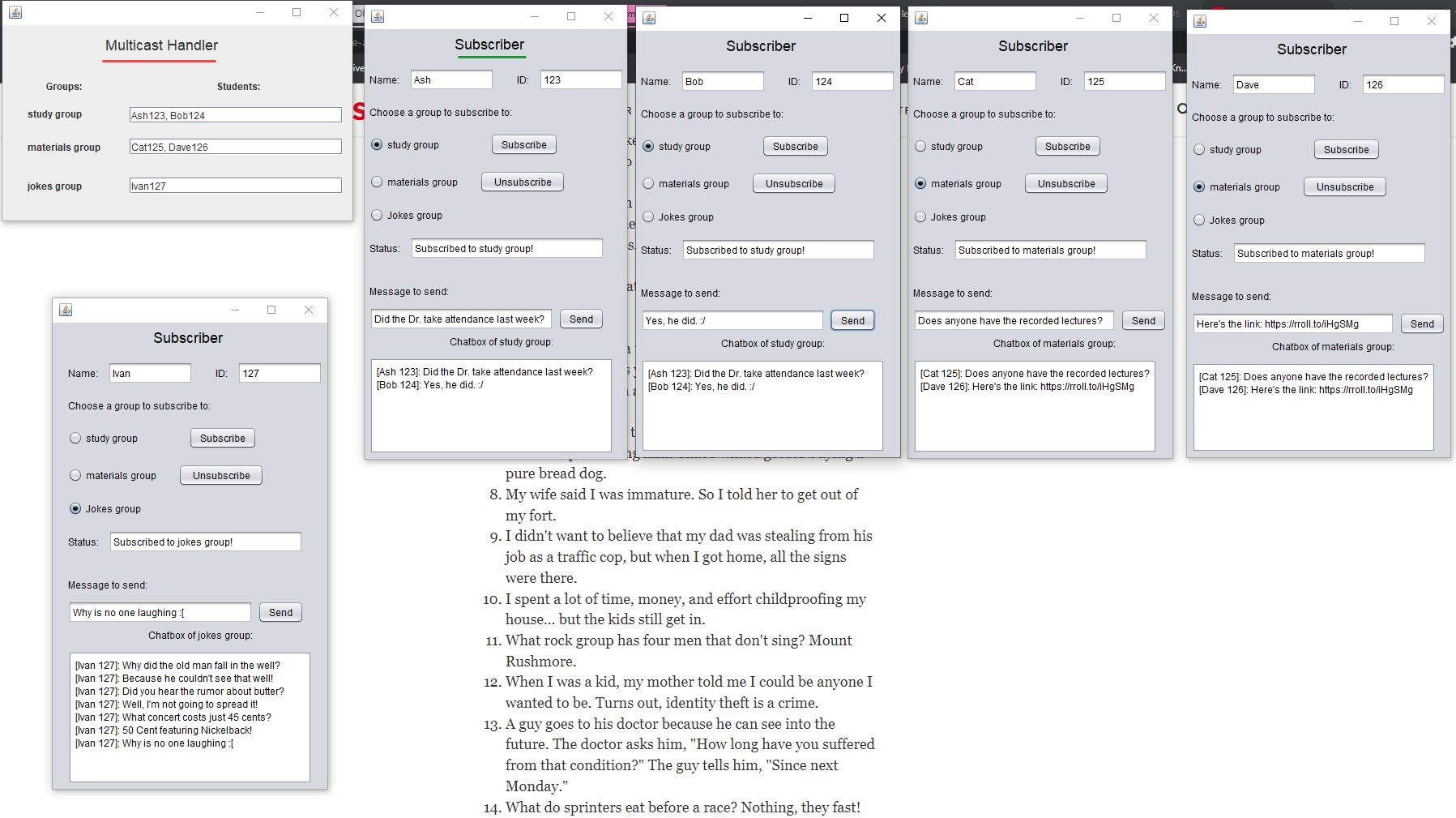
**Update**: updateChatbox(String msg, **boolean toAll**) where toAll indicates that this message will be displayed in the announcment text area (all subscribers), not to a specific group.

Note that only the interfaces are shown here, the rest of the classes are illustrated below.

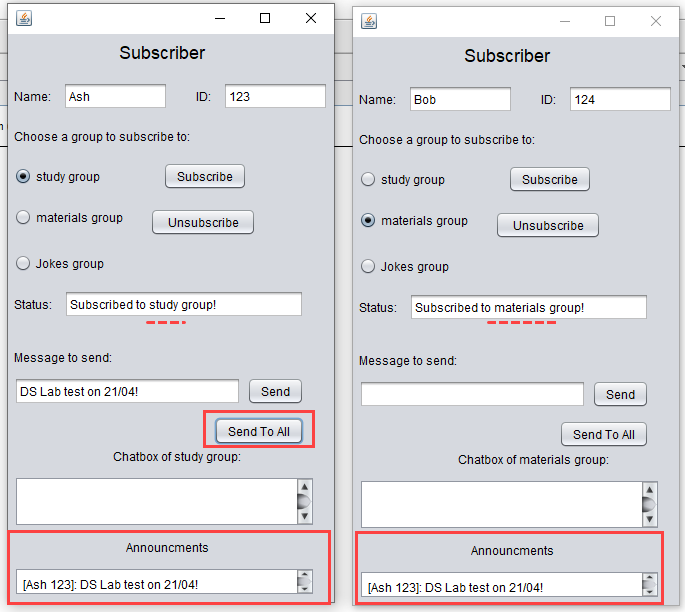
### 

### The Output

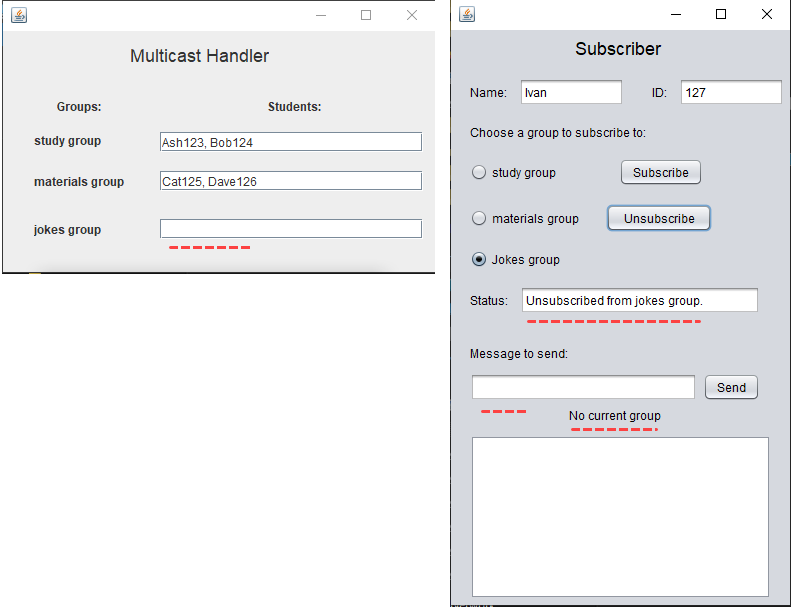
The output of the system for five subscribers looks like this:



**Update**: The announcement chatbox is added in the subscriber’s GUI:

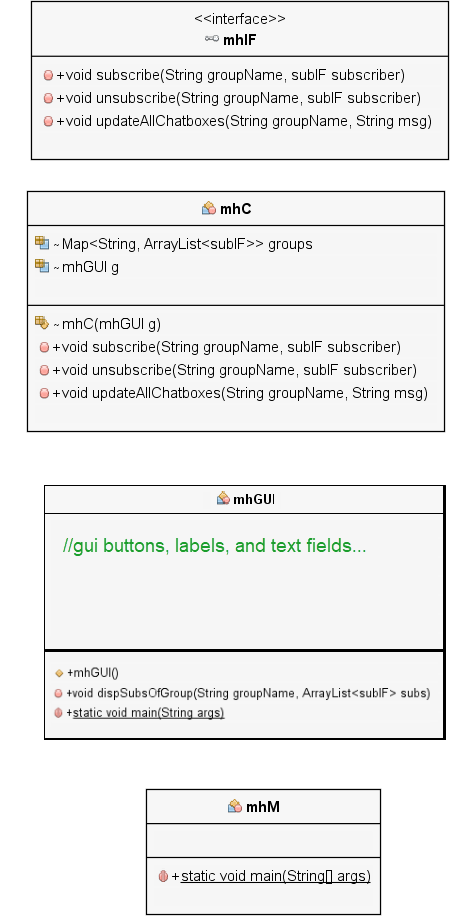


If a student (for example, Ivan) unsubscribes:



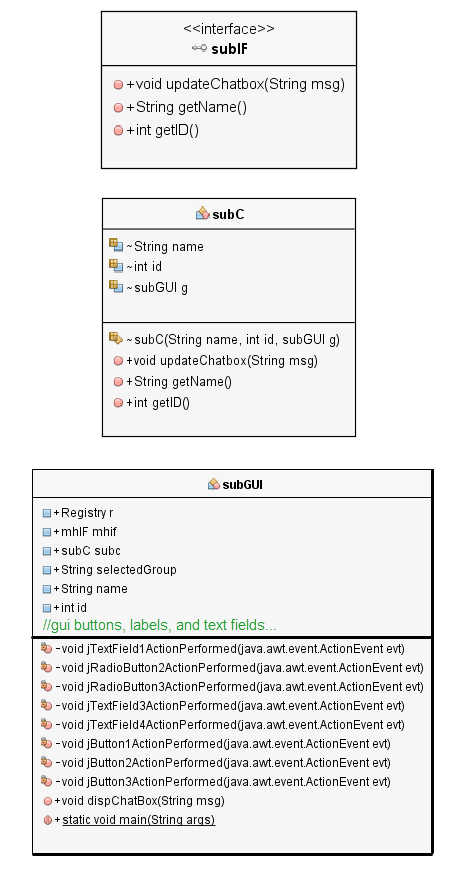
### Multicast Handler Classes

* mhIF → The interface which the remote class implements
* mhC → The remote class
* mhGUI → the Graphical User Interface of the server (GUI)



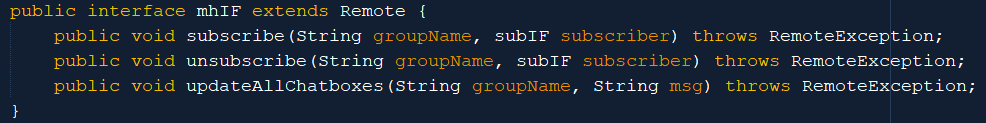
### Subscriber Classes

* subIF → The interface which the remote class implements
* subC → The remote class
* mhGUI → the Graphical User Interface (GUI) of the subscriber (student)

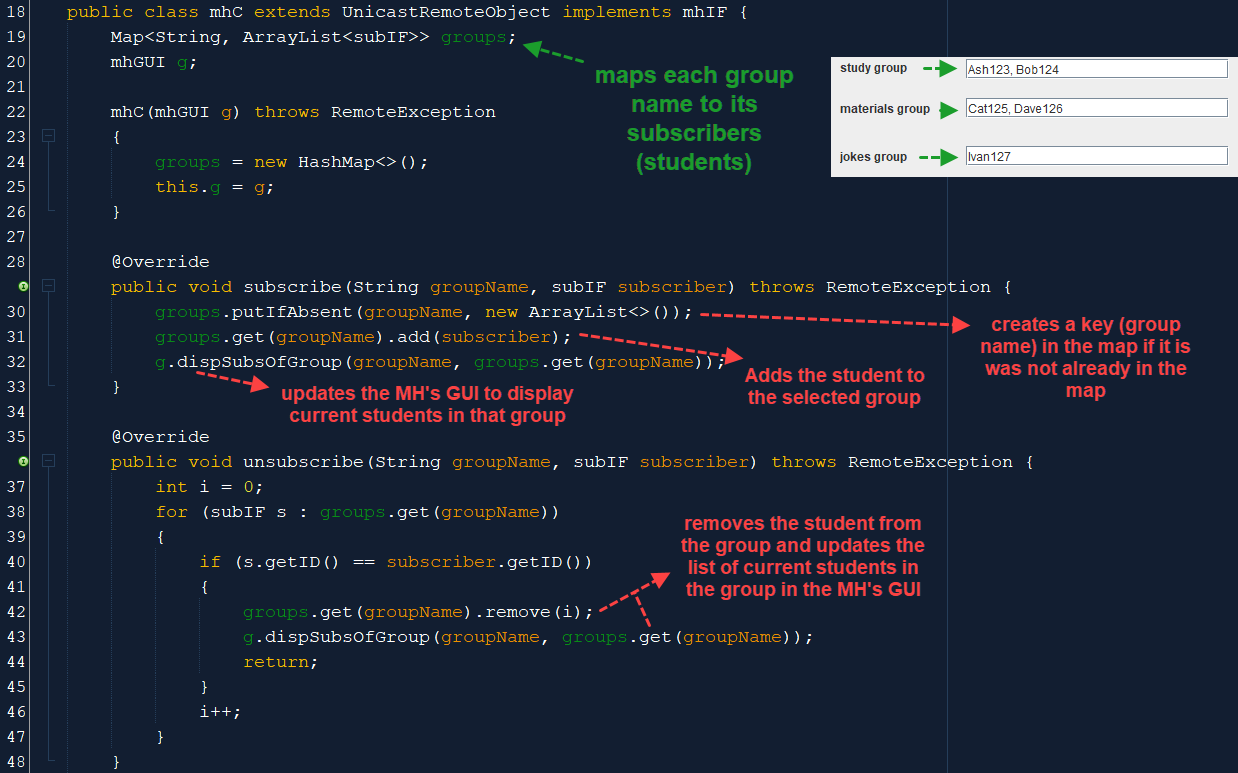


## Design Mapping To Code

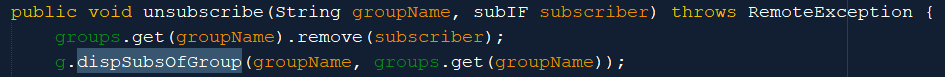
### Multicast Handler Interface (mhIF)

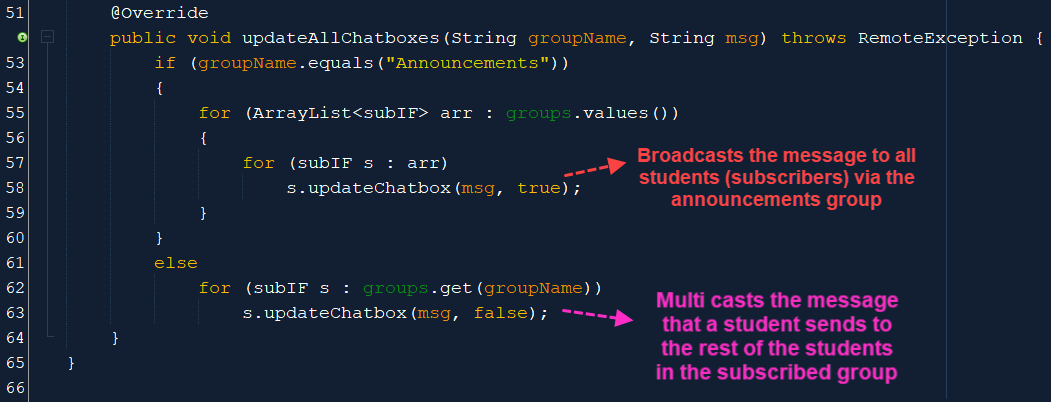


### Multicast Handler Class (mhC)

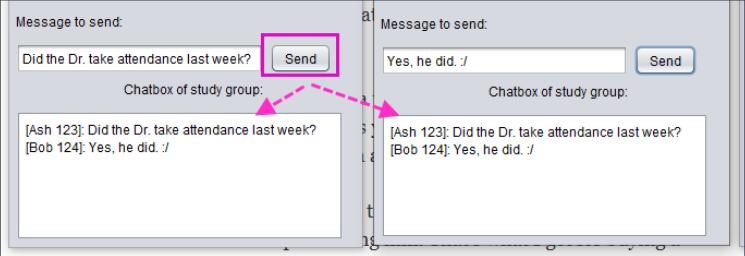
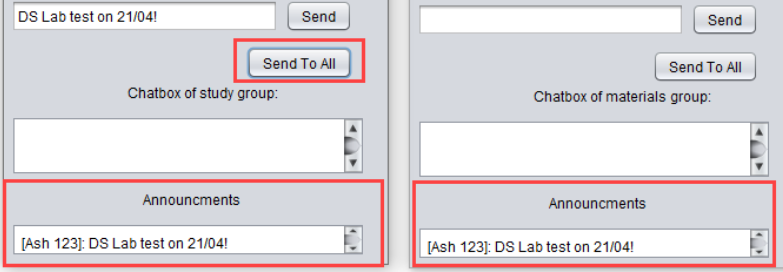


Alternative method for unsubcribing:

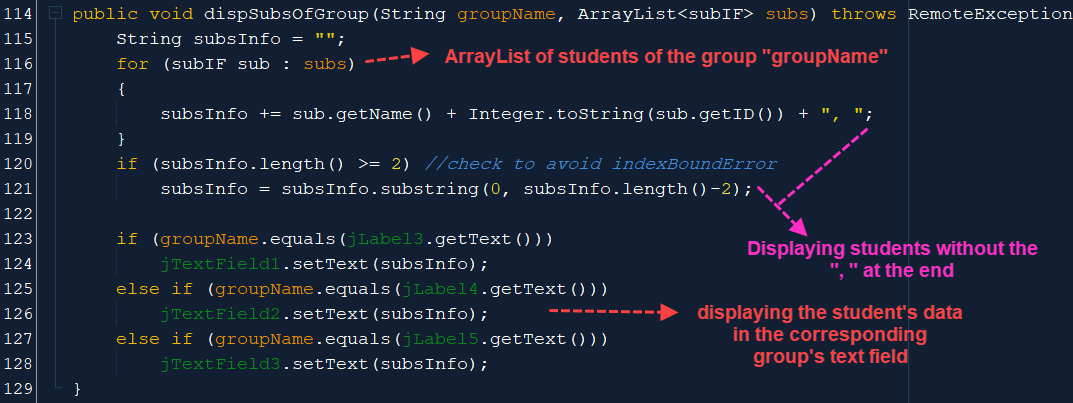


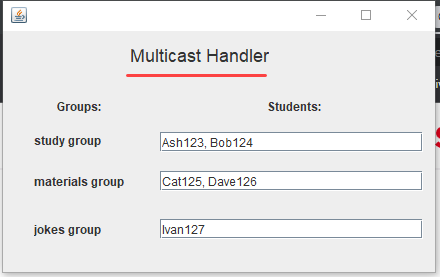


Example when if condition is true: Example when if condition is false:



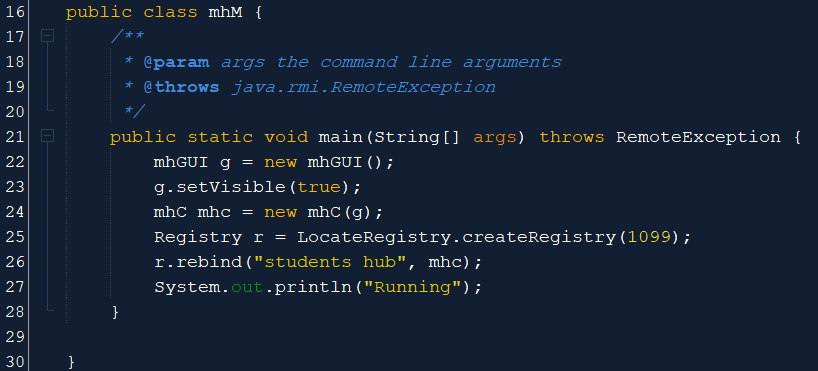
### Multicast Handler GUI (mhGUI)



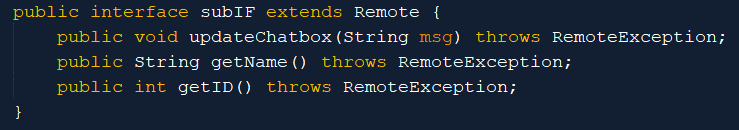
 (no **“, “** at the end)

Note that the rest of the code is the Java GUI’s automatically generated code (buttons, labels, etc).

### Multicast Handler Main (mhM)

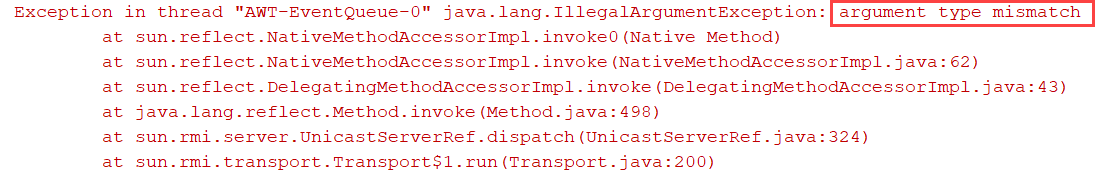


### Subscriber Interface (subIF)



**Update**: updateChatbox(String msg, **boolean toAll**)

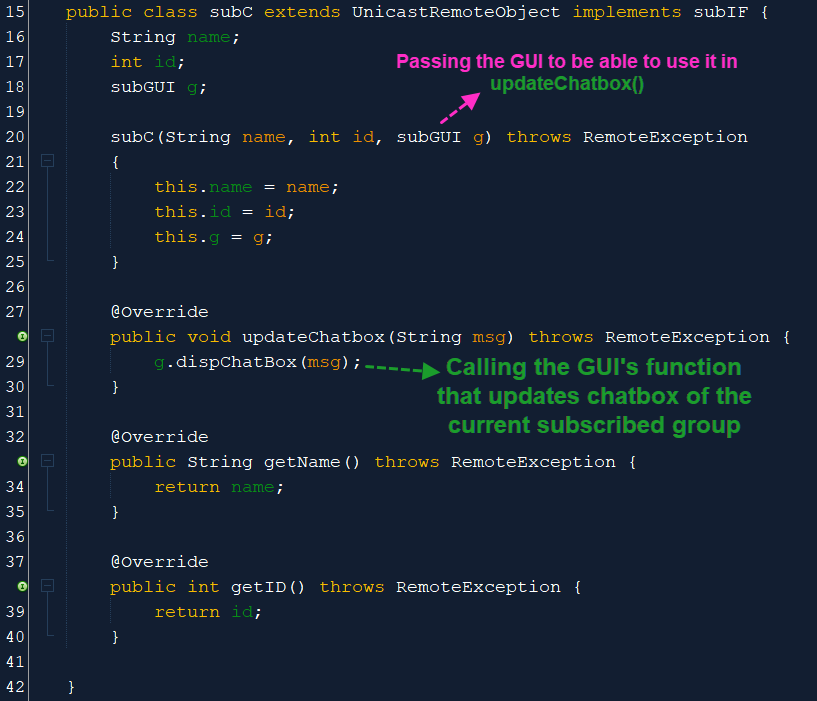
Debugging note: not writing “extends Remote” will cause this error:



When passing this subIF interface as an argument of another function like this:

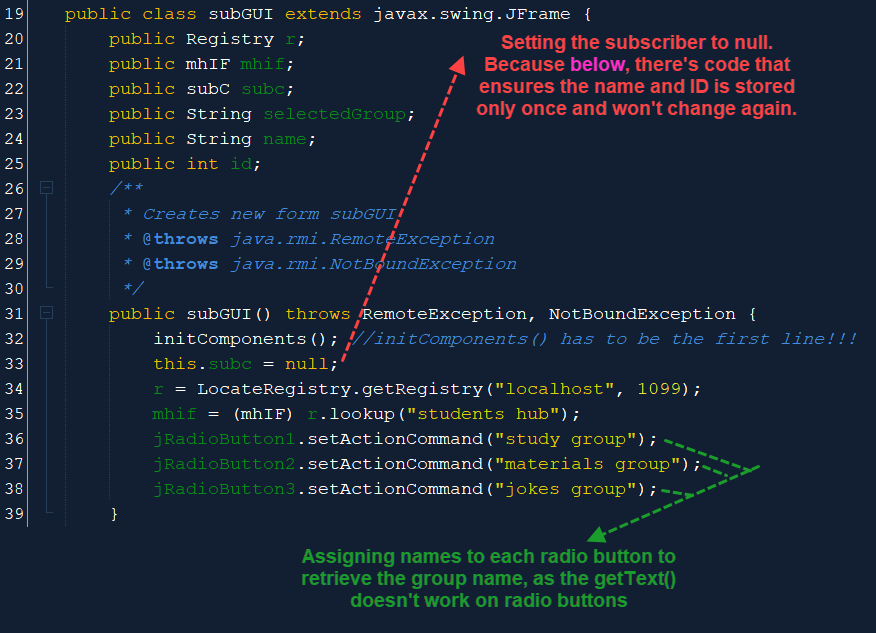


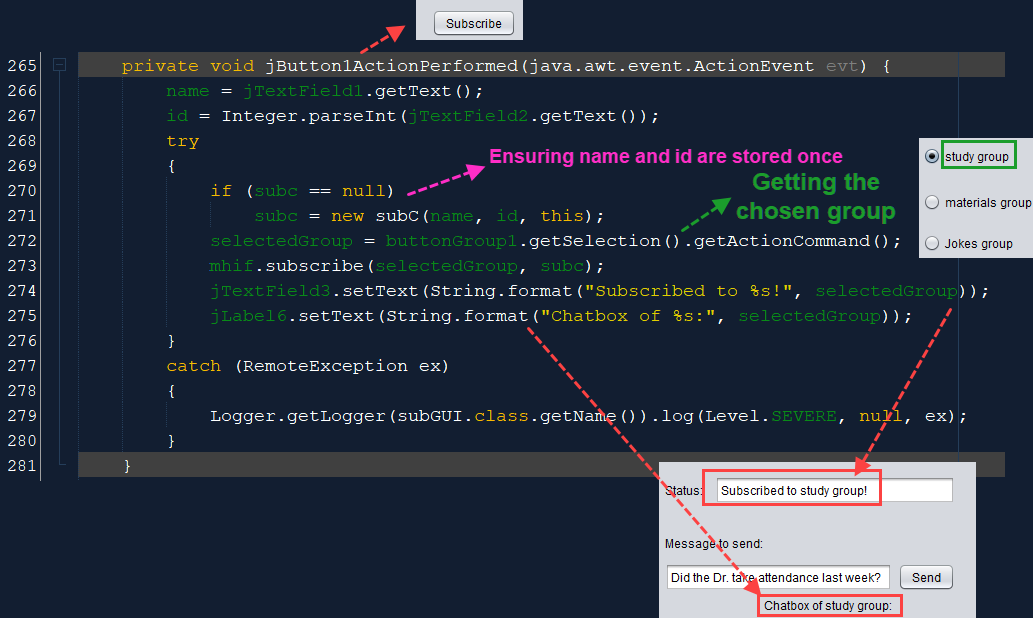
### Subscriber Class (subC)



**Update**: g.dispChatBox(msg, **toAll**) which is an argument in updateChatbox().

### Subscriber GUI (subGUI)

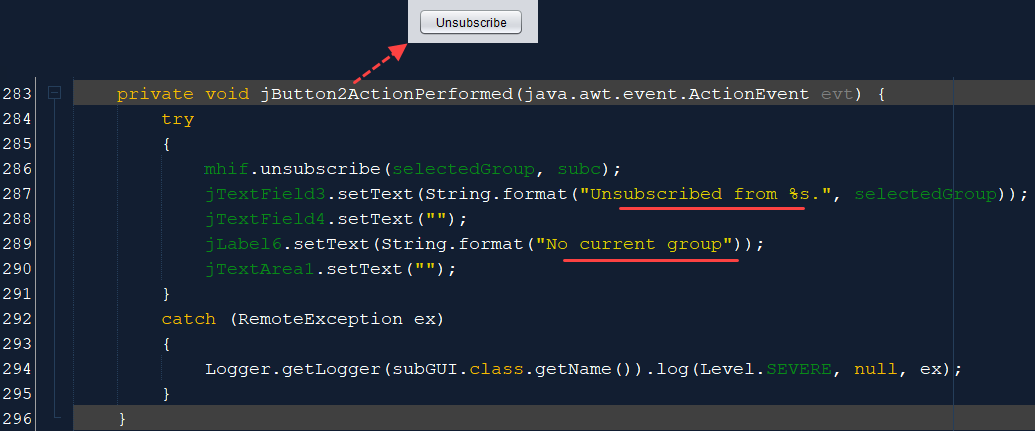


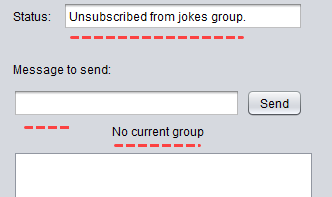


**Update 1**: in line 273, “subc” should be casted like this: “(subIF) subc”

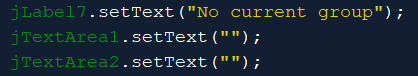
**Update 2**: under line 275:

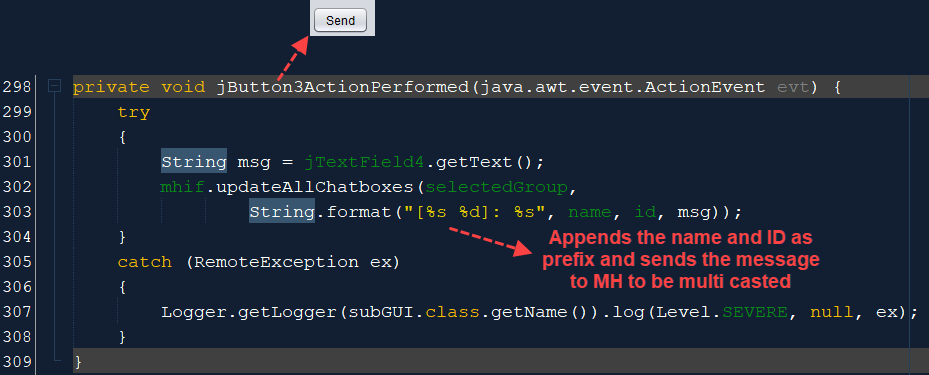


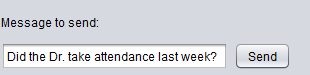




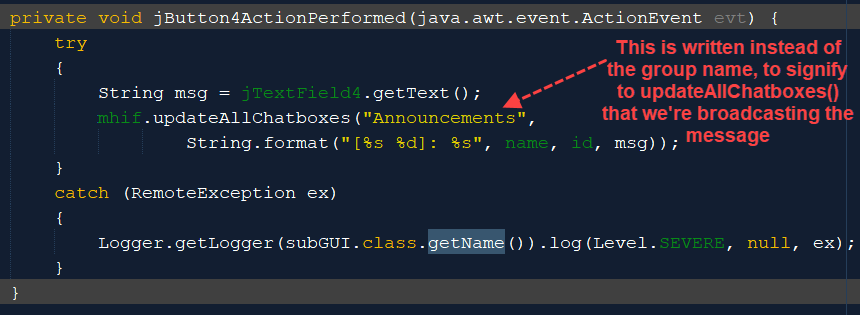
**Update**: under line 289:

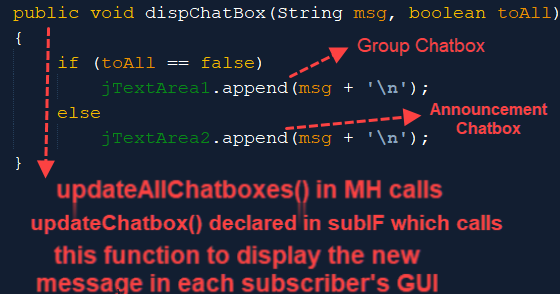






**Update**: “Send To All” button:





## Assumptions

* In the subscriber's GUI, the user will always write their name and ID, then click on a group from the listed groups, then press on subscribe, so handling cases like the user clicking on unsubscribe first or empty name/ID forms will not be implemented.
* Also, it is assumed that the user will always unsubscribe from their current group before subscribing to another group.
* It is also assumed that the user will only modify the designated text fields like "name:" , "ID:", and "Message to send:" and no attempts will be made to modify "Status:" and the "Chat Box" text area.
* Finally, it is assumed that the student will enter the correct information (name and ID) about themselves, as the university stated that any reports of an inappropriate name or false ID will result in a severe punishment to the student responsible for this (which will be known via their IP address).

## Advantages & Disadvantages of Multicast Handling System

The main advantage is the scalability compared to unicast traffic. Other advantages:

* Enhanced Efficiency: Controls network traffic and reduces server and CPU loads
* Optimised Performance: Eliminates traffic redundancy
* Distributed Applications: Makes multipoint applications possible

Main disadvantage is that it is required to have applications that support multicast and the network to support it needs to be configured. Other disadvantages:

* Best Effort Delivery: Drops are to be expected. Multicast applications should not expect reliable delivery of data and should be designed accordingly. Reliable Multicast is still an area for much research.
* No Congestion Avoidance: Lack of TCP windowing and “slowstart” mechanisms can result in network congestion. If possible, Multicast applications should attempt to detect and avoid congestion conditions.
* Duplicates: Some multicast protocol mechanisms result in the occasional generation of duplicate packets. Multicast applications should be designed to expect occasional duplicate packets.