ICOM 4015-Advanced Programming
Spring 2014
Instructor: Dr. Amir H. Chinaei
TAs: Hector Franqui, Jose Garcia, and Antonio Tapia

Reference: **Big Java**
By Hortsmann, Ed 4

Lab 5
Introduction to Jar Files, RESTFUL Methods and Web Sockets.

Department
of
Electrical and Computer Engineering
University of Puerto Rico at Mayagüez

Before laboratory:

1. Review Java Jar Files
2. Revisit JPanels
3. Revisit Action Listeners
4. Review JComboBox,JTextField, JTextArea
5. Print (at least) the Evaluation sheet in the last page.

0 - Login to computer (1 minute)

1 - Using JAR Files(20 minutes)

- Download all the jar files from https://ece.uprm.edu/~ahchinaei/courses/2014jan/icom4015/lab5
- Create a new Java project by following the steps "Ctrl+n > Java Project" name it and click finish.
- Now we will create folder in the same level as the " *src*" folder in your Eclipse project and name it "*lib* . Select the project and follow these steps "Ctrl+n > Folder" now name it "*lib*". *Ambiguous: Where in eclipse? Workspace or somewhere else*
- Drag/Drop the Jar files into the *lib* folder.
- We need now to refresh our project so that Eclipse can acknowledge the jar files, "select the project > right mouse click > refresh ". *Ambiguous: What do you mean by refresh your project? Which project? Are we yet in any project?*
- Right click on each jar file in the folder and select the option "Build Path > Add To build path"
- In the end you should have had added all your jar files to your build path and your project should look as follows:

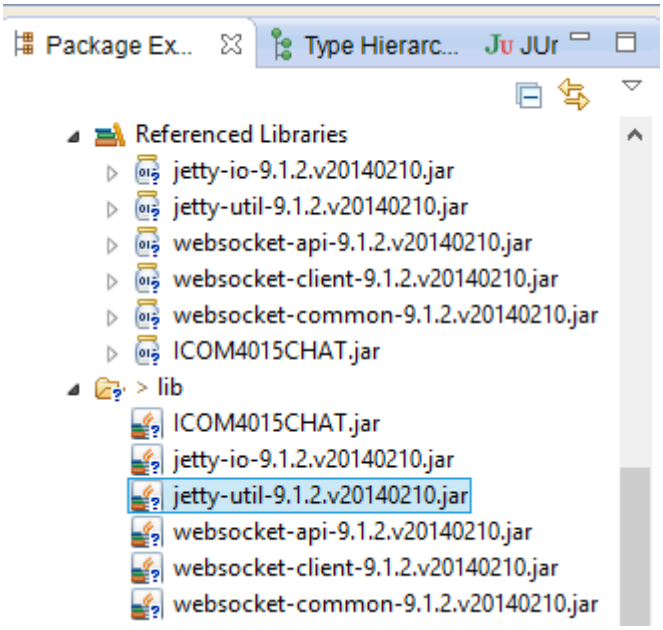

Figure 1

- **Show your project to instructor (20 points)**

2 - Build a GUI (25 minutes) – Compile in the JFrame and run it.

- Create a new class and call it *WelcomePanel* and add the following code to it: *why your codes are pictures? This way, students have to type it instead of copying it. Typing it has any advantage?*

```
@Override
void init() {
        JButton eButton = new JButton("Enter");
        eButton.setActionCommand("enterRoom");
        eButton.setName("enter");


        String[] roomNames = getChatNames();

        JComboBox<String> combo = new JComboBox<String>(roomNames);
        combo.setName("roomDropDown");
        combo.setActionCommand("checkRoom");

        JTextField field = new JTextField("Username", 2);
        field.setName("nameField");

        JPanel buttonPane = new JPanel();
        buttonPane.setLayout(new GridLayout(0,5));

        buttonPane.add(new JLabel("Name:"));
        buttonPane.add(field);
        buttonPane.add(new JLabel("Room Number:"));
        buttonPane.add(combo);
        buttonPane.add(eButton);
        buttonPane.setPreferredSize(new Dimension(630, 30));

        JPanel mainPanel = new JPanel();
        mainPanel.add(buttonPane,BorderLayout.CENTER);
        mainPanel.setPreferredSize(new Dimension(630,400));
        this.add(mainPanel,BorderLayout.CENTER);

        JLabel status = new JLabel("Welcome");
        status.setName("status");
        status.setPreferredSize(new Dimension(630, 30));
        this.add(status,BorderLayout.PAGE_END);

        this.setBorder(new EmptyBorder(20,20,20,20));
}

private String[] getChatNames() {
        String knownNames[] = {
                        "b/",
                        "irc",
                        "Mofongo"
        };
        String roomNames[] = new String[20];
        for(int i=1;i<=20;i++){
                roomNames[i-1]=knownNames[(i-1)%knownNames.length]+((i-
1)/knownNames.length);
```

```
        }
        return roomNames;
}
```

- Now add the following main method to test the code:

```
/**
 * Test UI
 * @param args
 */
public static void main(String... args){
        JFrame window = new JFrame();
        window.setContentPane(new EnterPanel());
        window.setVisible(true);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setSize(680, 520);
}
```

  - Run this class and you should see the window below:

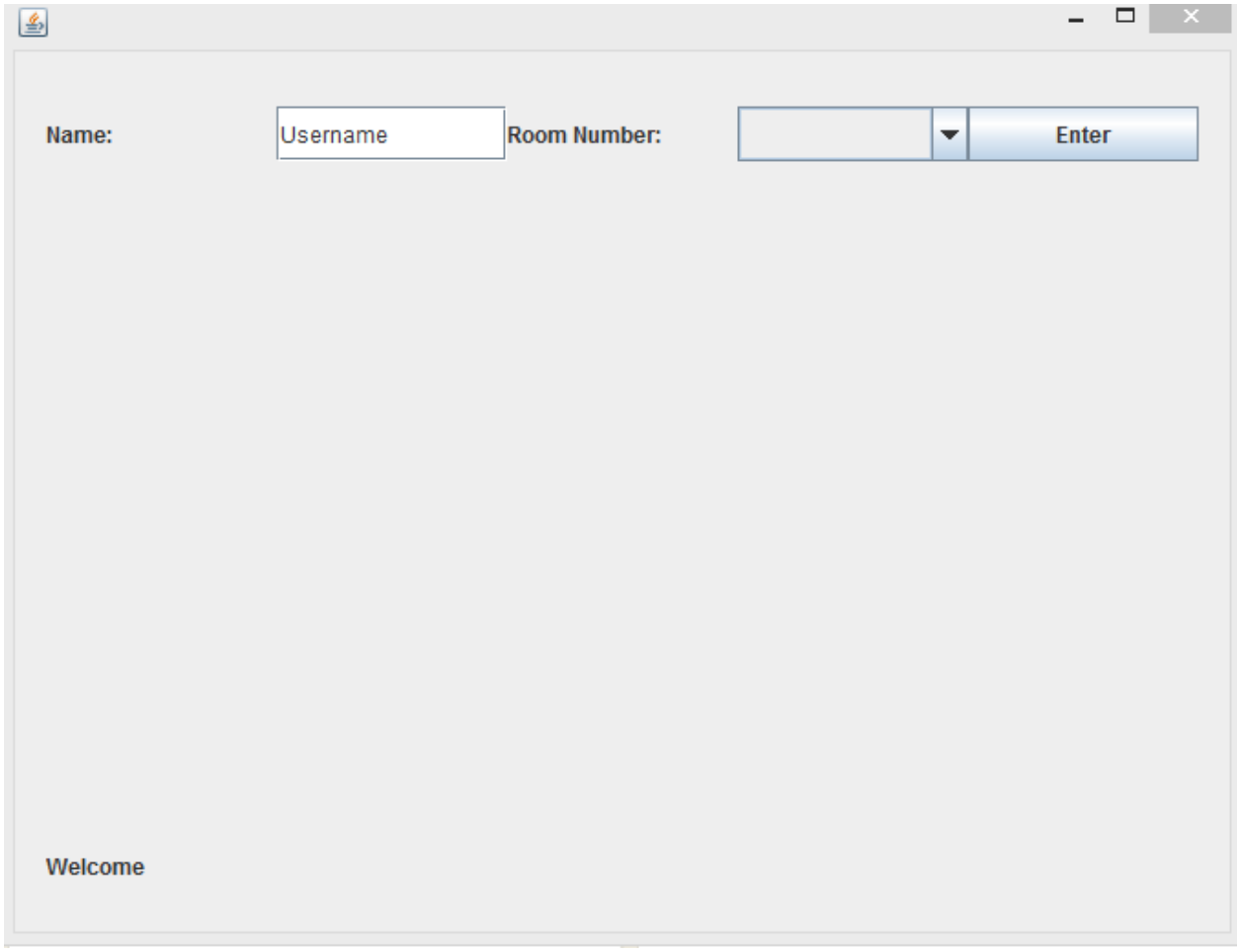

Figure 2

- Now create a new class and name it “ChatPanel” and add the following piece of code to it:

```
@Override
void init() {

        /**
         * Text Area
         */
        JTextArea chatArea = new JTextArea("Welcome");
        chatArea.setName("chatArea");
        chatArea.setPreferredSize(new Dimension(630,300));
        chatArea.setEditable(false);

        /**
         * Button Bar
         */
        JButton sButton = new JButton("Send!");
        sButton.setActionCommand("sendMessage");
        sButton.setName("send");
        sButton.setPreferredSize(new Dimension(50, 30));

        JTextField messageField = new JTextField("Hey, sup?");
        messageField.setName("messageField");
        messageField.setPreferredSize(new Dimension(550, 30));

        JPanel buttonPane = new JPanel();
        buttonPane.setLayout(new GridLayout(0,2));
        buttonPane.setPreferredSize(new Dimension(630, 30));
        buttonPane.add(messageField);
        buttonPane.add(sButton);

        /**
         * Main Pane
         */
        JPanel chatPane = new JPanel();
        chatPane.add(chatArea,BorderLayout.CENTER);
        chatPane.add(buttonPane,BorderLayout.PAGE_END);
        chatPane.setPreferredSize(new Dimension(630,350));

        /**
         * Status Label
         */
        JLabel status = new JLabel("Welcome");
        status.setName("status");
        status.setPreferredSize(new Dimension(630, 30));

        /**
         * Exit Button
         */
        JButton exitRoom = new JButton("Exit Room!");
        exitRoom.setName("exitRoom");
```

```
        exitRoom.setActionCommand("exitRoom");
        exitRoom.setPreferredSize(new Dimension(630,30));
        /**
         * Main Panel
         */
        this.add(exitRoom,BorderLayout.PAGE_START);
        this.add(chatPane,BorderLayout.CENTER);
        this.add(status,BorderLayout.PAGE_END);

        this.setBorder(new EmptyBorder(20,20,20,20));

    }
-
```

- Now add the following main method to test the code:

```
/**
 * Test UI
 * @param args
 */
public static void main(String... args){
        JFrame window = new JFrame();
        window.setContentPane(new ChatPanel());
        window.setVisible(true);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setSize(680, 520);
}
```

- Run this class and you should see the window below:

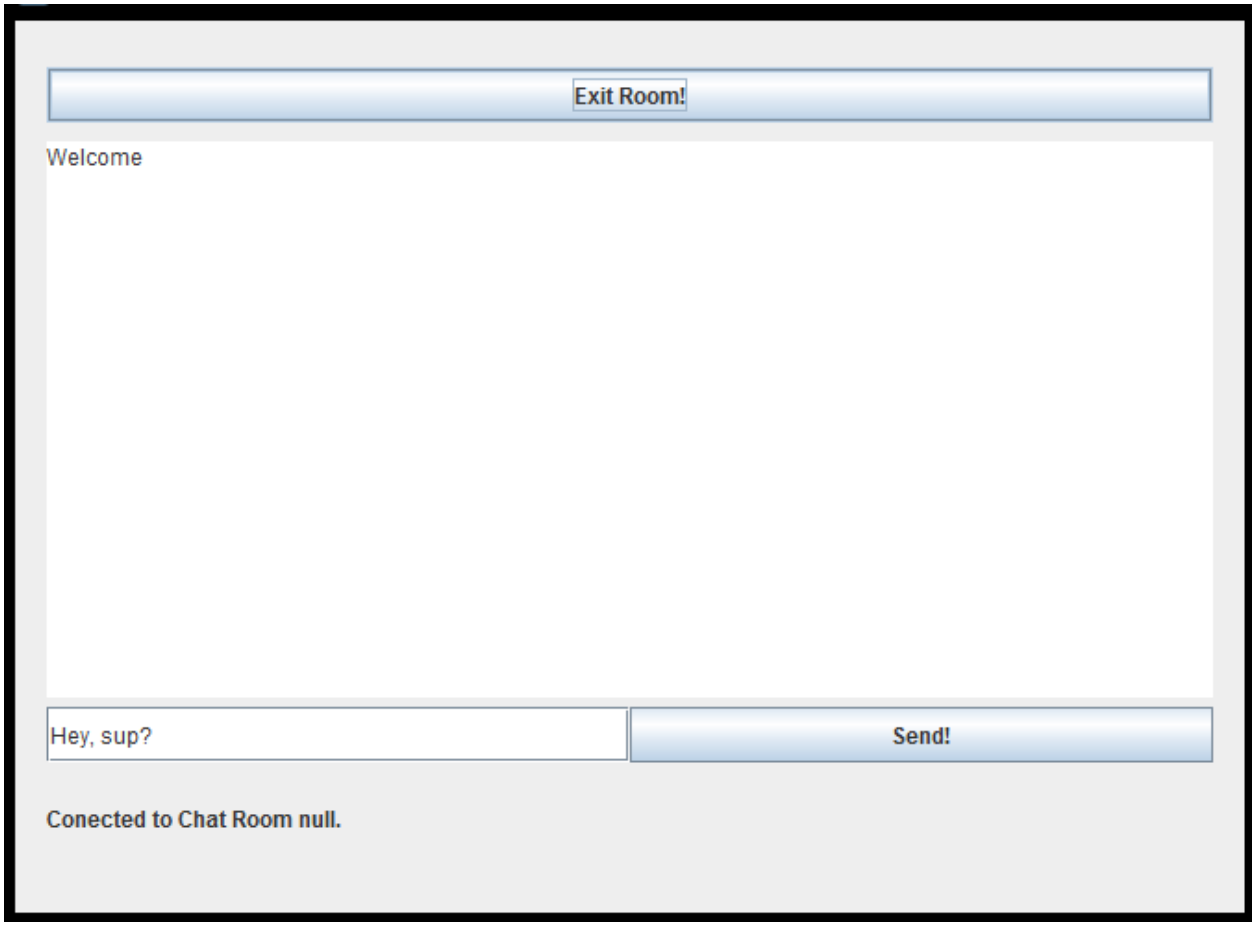

Figure 4

- Let us now customize the UI Code below so that it looks more personal (change button colors, background color...)
  .
- **Show your code to instructor (20 points)**

3 - Chat Engine (10 minutes)

- This an abstract class that already implements most of what we need to make our chat client. We now need to complete the implementation.

- Create a new class that extends the "ChatEngine" class, namely "MyClassEngine".

- Now Implement the method "setName()".

```
@Override
public void setName() {
```

```
        JTextField name = (JTextField)fetchComponent(null,
        "nameField");
        this.name = name.getText();
}
```

- Notice that the class chat engine has some objects and methods already implemented. The fetch Component methods searches for whatever swing component you need to find in the current window!

4 - HTTP . (35 minutes)

- The Hypertext Transfer Protocol is the most used protocol on the web for information sharing.

- TAs explain information on slides (15 minutes). For TAs note that the "HTTPConnector.sendGet()" and "HTTPConnector.sendPost()" method already takes care of all the details of creating the respective HTTP request.

- Now implement the "checkRoomImp" method.

```
@Override
public boolean checkRoomImp(String roomName, String url) {
        String[] parameters = {
                "room="+roomName
        };
        String result = HTTPConnetor.sendGet(url,parameters ,
        System.out);
        return result.contains("\"success\":true");
}
```

5 - Do a POST that adds User to the Room parsing the server response.

- Implement the "exitRoomImp"

```
@Override
public boolean exitRoomImp(String roomName, String name, String url) {
        String[] parameters = {
                "room="+roomName,
                "name="+name
        };
        String result = HTTPConnetor.sendPost(url,parameters ,
        System.out);
        return result.contains("\"success\":true");
}
```

- **Show your code to instructor (20 points)**

- Now it's up to you to implement the POST for "registerUserInRoomImp".

- **Show your code to instructor (20 points)**

6 - Sockets (19 minutes)

- Is another method of communication on the web and it is commonly used in chat services, games and all services needing duplex(two way) communication.

- TAs explain information on slides (10 minutes). Notice that the socket is passed as an object and that all messages send to the server must be send in the JSON format.

  JSON format in a nutshell:

```
{
    "name":name,
    "message":message,
    "room":roomName,
    "isMessage":true,
    "acknowledge":true
}
```

- Ask to implement a WEB SOCKET for "sendMessageImp" user clicks send message to chat(15 minutes).

- **Show your code to instructor (20 points)**

After Lab

- **Read on HTTP Methods**
- **Evaluation Sheet**
- **(optional)Check out the chat server on Github "https://github.com/Apo45ty/NodeSimpleChatServer"**

Evaluating Lab 4

Write Your Section# here: ……..

Please evaluate the quality of the lab and performance of the instructors by filling up the following table and give it to your lab representative. (Choose 5 as the highest and 1 as the lowest grade).

Items	5	4	3	2	1
The lab started **on time SHARP**.					
The instructor covered adequately the HTTP Methods, WebSockets and answered the group's questions thoroughly.					
The instructor covered adequately Jar files and answered the group's questions thoroughly.					
The instructor **overall** followed the specified **timeline** for each step					
You found the lab today **overall** Great (helpful, fruitful, interesting, etc.).					