Massachusetts Institute of Technology
6.005: Elements of Software Construction
Fall 2011
Quiz 2
November 21, 2011

Name: **SOLUTIONS**

Athena User Name: ______________________________

Instructions

This quiz is 50 minutes long. It contains 7 pages (including this page) for a total of 100 points. The quiz is closed-book, closed-notes.

Please check your copy to make sure that it is complete before you start. Turn in all pages, together, when you finish. Write your name on the top of every page. Please write neatly. No credit will be given if we cannot read what you write. Good luck!

Question Name	Page	Maximum Points	Points Given
Design Patterns	2	20	
Interpreter/Visitor	3	16	
Map/Filter/Reduce	4	12	
Concurrency	5	16	
Deadlock	6	16	
Thread Safety	7	20	

Name: ______________________________

Design Patterns [20 pts]

For each of the following statements, name the design pattern that it **best** describes, from the list below.

Interpreter
Visitor
Event Listener
Map/Filter/Reduce
Client/Server
Model/View/Controller
Composite

You may use a design pattern more than once in your answers. If you're torn between two best answers, you can give both, but in that case you should justify both answers.

(a) This design pattern produces tree-like data structures.

Composite

Interpreter or Visitor given partial credit, because these are often used to write producers of a tree-like datatype

(b) This design pattern is used for operating over sequences of elements.

Map/Filter/Reduce

(c) This design pattern uses higher-order functions.

Map/Filter/Reduce

Visitor also given full credit, because a Visitor is a functional object

(d) This design pattern is used to separate concerns in user interfaces.

Model/View/Controller

Event Listener also given full credit, because events decouple models from views and input from output

(e) This design pattern is used for message passing over a network.

Client/Server

Name: ______________________________

Interpreter/Visitor [16 pts]

You want to write a program to perform operations on all your Foos.

A Foo can perform lots of different tricks, like *bazzle* and *glibble*.

There are (and will always be) exactly 4 types of Foo, each of which does something different when they *bazzle* or *glibble*.

But every so often your Foos learn a new trick, and you must update your program to include the new operation.

For example, last week your Foos learned how to *joople*.

a) Would it be better to use the interpreter pattern or the visitor pattern for implementing the datatype representing a Foo?

Visitor, because the variants of the Foo datatype are fixed, but new operations appear from time to time.

b) Assuming you designed your program according to your choice in part (a), now you want to add the *joople* operation. Explain what classes and methods you will change, or what classes and methods you would add, in order to support the *joople* operation.

Add a JoopleVisitor class implementing the Foo's Visitor interface. JoopleVisitor needs to define a visit() or on() method for each of the four variants of Foo. (Optional: also add a static method that makes it easy to call *joople* on a Foo, by encapsulating the construction of the JoopleVisitor.)

Name: ____________________________________

Map/Filter/Reduce [12 pts]

Suppose you want to rewrite the following Python code using map, filter, and reduce:

```
def ssp(list):      # sum of squares of positive numbers in list
    result = 0
    for x in list:
        if x > 0:
            result += x*x
    return result
```

Fill in the blanks in the map/filter/reduce version below.

```
def ssp(list):      # sum of squares of positive numbers in list
    return reduce(r, map(m, filter(f, list)), 0)


def f(    x                                        ):

    return   x > 0


def m(   x                                         ):

    return   x * x


def r(    x, y                                         ):

    return  x + y
```

Name: ____

Concurrency [16 pts]

Read the following code:

```
public static void main() {
    Thread t1 = new Thread(new Runnable() {
        public void run() {
            System.out.print("O");
            System.out.print ("Y");
        }
    });
    Thread t2 = new Thread(new Blue());
    System.out.print("R");
    t1.start();
    System.out.print("G");
    t2.start();
    System.out.print("I");
    t1.join();
    System.out.print("V");
    t2.join();
    System.out.print("K");
}

public static class Blue implements Runnable {
    public void run() {
        System.out.print("B");
    }
}
```

Assume that print() is threadsafe and atomic. Which of the following sequences can be printed by this code? Circle possible or impossible.

ROYGBIVK	[✓ possible]	impossible
ROYBGIVK	possible	[✓ impossible]
RGOYIBVK	[✓ possible]	impossible
OYBRGIVK	possible	[✓ impossible]

Name: ____

Deadlock [16 pts]

You have two threads (T0 and T1) and two locks (X and Y). Which of the following situations can lead to deadlock? If deadlock can occur, circle the method call in each thread where the thread would stop in the event of deadlock. If deadlock is impossible, circle "no deadlock."

a)

```
T0:                     T1:
X.acquire();            X.acquire();
Y.acquire();            Y.acquire();
Y.release();            X.release();
X.release();            Y.release();
```

[✓ no deadlock]

b)

```
T0: (same as T0 above)  T1:
X.acquire();            Y.acquire();
[✓ Y.acquire();]        [✓ X.acquire();]
Y.release();            X.release();
X.release();            Y.release();
```

no deadlock

c)

```
T0: (same as T0 above)  T1:
X.acquire();            Y.acquire();
Y.acquire();            Y.release();
Y.release();            X.acquire();
X.release();            X.release();
```

[✓ no deadlock]

Name: ____

Thread Safety [20 pts]

Consider the following code, and answer the questions on the next page.

```
public class Widget extends Thread {
    public static List<String> strings = new ArrayList<String>();
    public int count;
    public List<Integer> numbers;

    public Widget() {
        count = 0;
        numbers = new ArrayList<Integer>();
    }

    public void run() {
        for (int i = 0; i < 1000; ++i) {
            synchronized (this) {
                count++;
                synchronized (numbers) {
                    numbers.add(i);
                }
                synchronized (Widget.strings) {
                    Widget.strings.add(“x”);
                }
            }
        }
    }

    public static void main(String[] args) {
        List<Widget> widgets = new ArrayList<Widget>();
        for (int i = 0; i < 1000; ++i) {
            Widget w = new Widget();
            widgets.add(w);
            w.start();
        }
        for (Widget w : widgets) {
            synchronized (w) {
                w.count++;
                synchronized (w.numbers) {
                    w.numbers.add(1000);
                }
            }
            synchronized (Widget.strings) {
                Widget.strings.clear();
            }
        }
        for (Widget w : widgets) {
            w.join();
        }
    }
}
```

You are reviewing a concurrency argument about this code. Circle whether you agree or disagree with each of the following statements in the concurrency argument, and **add a brief (1 sentence) justification of your answer**.

(a) Accesses to the `widgets` list are safe because the list is confined to the main thread.

Name: ____

AGREE DISAGREE

The only reference to the widgets list is the local variable widgets in main(), which is never shared with any other thread.

(b) Accesses to the `numbers` list are safe because they acquire the list's lock.

[✓ AGREE] DISAGREE

All accesses to the numbers list happen inside a synchronized(numbers) block.

(c) Assuming that the program terminates without throwing an exception, `count` for every widget is 1001 at the end of main.

[✓ AGREE] DISAGREE

All accesses to count are guarded by the Widget object's lock, and the Widget's run() increments it 1000 times while the main() loop increments it once, producing 1001.

(d) Assuming that the program terminates without throwing an exception, `strings` has size 0 at the end of main.

AGREE [✓ DISAGREE]

The strings.clear() in main() races against the strings.add() calls in run(); even though both are threadsafe, it may be that the last operation executed against strings is an add().

END OF QUIZ

6.005 Elements of Software Construction
Fall 2011
Project 2: Instant Messaging
Monday, November 21

Due Dates:

Milestone 1: midnight, Tuesday, November 29
Milestone 2: midnight, Tuesday, December 6
Possible amendment: Wednesday, December 7
Prize consideration: 11am, Tuesday, December 13
Final version: midnight, Wednesday, December 14
Reflection: midnight, Thursday, December 15

Problem

Instant messaging (IM) is a staple of the web and has been around almost since its inception, starting with simple text-based programs like talk and IRC and progressing to today's GUI-based IM clients from Google, Yahoo, Microsoft, AOL, etc. In this project you will design and implement an IM system, including both the client and the server. The following characteristics constrain the design space of an IM system:

- **Real-time communication.** An IM conversation happens in real time: one person types some text, presses "enter," and the other person (almost) immediately sees the text.
- **Number of parties.** An IM conversation can happen between two or more people. Some systems only allow two people to communicate; others allow more than two people. Most systems allow a person to be involved in multiple conversations at the same time.
- **Based on typed text.** The main mode of communication is via text, as opposed to voice or video.
- **Connected over a network.** The parties involved in the communication may be in physically remote locations, and are connected over the internet.

Your task will be to design an instant messaging system with the above properties, as well as additional properties that you will incorporate into your design. This system will include a server component that handles the transfer of messages and other data, and a client component with a graphical user interface.

Server as middle!

Purpose

The purpose of this project is twofold. First, you will use several Java technologies, including networking (to support connectivity over a network), sockets and I/O (to support real-time, text-based communication), and threads (to support two or more people communicating concurrently). State machines may be useful to specify certain aspects of the system's behavior.

Second, you will have to think about the best way to present your chat system, this will required a graphical user interfaces. You will:

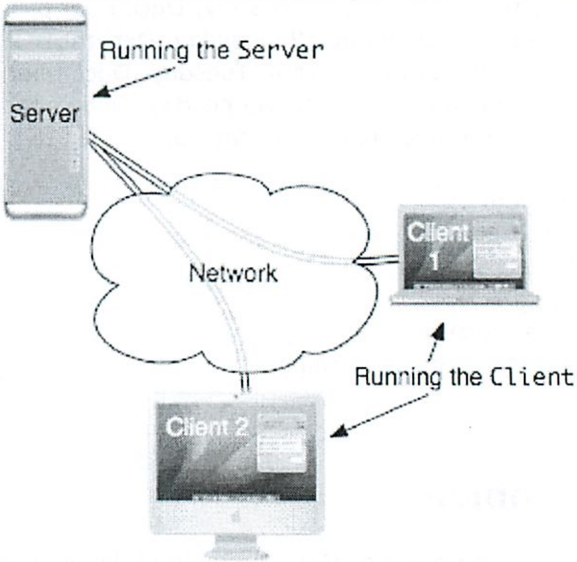
- become more familiar with Swing, a graphical user interface (GUI) toolkit for Java, that is similar to

many other such toolkits;
- use important GUI programming concepts, including the notion of a *view hierarchy* and the *model-view-controller* design pattern;
- use the event-listening design pattern in several ways, not only in your GUI but also in the more general publish-subscribe sense.

Throughout the project, you will need to design and implement mutable datatypes, paying particular attention to their specifications, how they interact with one another, and concurrency issues.

Specification

Implement an IM system in Java with the following properties:



- **Client.** The client is a program that opens a network connection with the IM server at a specified IP address and port number. The client should have a way of specifying the server IP, port, and a username. Once the connection is open, the client program presents a graphical user interface for performing the interactions listed below.
- **Server.** The server is a program that accepts connections from clients. A server should be able to maintain a large number of open client connections (limited only by the number of free ports), and clients should be able to connect and disconnect as they please. The server also has to verify that client usernames are unique and handle collisions gracefully.

  The server is responsible for managing the state of both clients and *conversations.*
- **Conversations.** A conversation is an interactive text-exchange session between some number of clients, and is the ultimate purpose of the IM system. The exact nature of a conversation is not specified (although the hints section details a couple of possibilities), except to say that it allows clients to send text messages to each other. Messaging in a conversation should be instantaneous, in the sense that incoming messages should be displayed immediately, not held until the recipient requests them. You should visually separate messages of different conversations (e.g., into distinct windows, tabs, panes, etc).
- **Client/server interaction.** A client and server interact by exchanging messages in a protocol of your devising — the protocol is not specified. Using this protocol, the user interface presented by the client should:
  - Provide a facility for seeing which users are currently logged in;
  - Provide a facility for creating, joining and leaving conversations;
  - Allow the user to participate in multiple conversations simultaneously;
  - Provide a history of all the messages within a conversation for as long as the client is in that conversation;
- **No authentication.** In a production system, logging in as a client would require some form of password authentication. For simplicity, this IM system will not use authentication, meaning that anyone can log in as a client and claim any username they choose.

Tasks

1. **Team preparation.** Meet with your team and write a team contract.
2. **Conversation design.** Define a precise notion of *conversation* in your IM system. See the hints on how to do this. Specifically, name the Java classes you will create to implementing conversations, give specs of their public methods, and give a brief description of how they will interact. Include a snapshot diagram of a conversation in action.
3. **Client/server protocol.** Design a set of commands the clients and server will use to communicate, allowing clients to perform the actions stipulated by the specification. Create a specification of the client/server protocol as a grammar. Also think about the state of the server, and the state of the client (if it stores any).
4. **Usability design.** Sketch your user interface and its various screens and dialogs. Use these sketches to explore alternatives quickly and to plan the structure and flow of your interface. *Sketching on paper* is recommended. Turn in the sketches you decided to go with for Milestone 2, along with commentary as needed to explain non-obvious parts.
5. **Concurrency strategy.** You should argue that your design is free or race conditions and deadlocks. Be specific about which data structures or design patterns you will use to ensure thread safe behavior.
6. **Testing strategy.** Devise a strategy for testing your IM system. Describe what automated tests you will use, and what manual tests you will perform. Since UI front-end testing is often most easily done by hand, documentation of your strategy is especially important. As you think about how to test your program, you are likely to find that you want to revisit your code design (for example, to make a cleaner API to permit unit testing independently of the GUI).
7. **Implementation.** As always, your code should be clear, well-organized, and usefully documented. See the hints for further suggestions.
8. **Testing.** Execute your testing strategy, using JUnit and by performing manual tests of the GUI. In your report, document the results of your manual tests.
9. **Reflection.** Each team member is to write a brief commentary describing what you learned from this experience, with one paragraph each about:
    - **Product**. What was easy? What was hard? What was unexpected? What would you do differently in designing the chat system if you were to do it again?
    - **Team**. How did you feel the group did? How did your team work? How was the coding? How did you split the work?
    - **Individual**. How do you think you did, personally? What did you do in the project? How do you feel about it?

Infrastructure

No initial code is provided for this project. However, two *runner* classes are provided with `main` methods you should fill in:

- Running `main.Client.main()` with no command-line arguments must start an instance of your GUI chat client.
- Running `main.Server.main()` with no command-line arguments must start an instance of your chat server.

You should consider using packages other than `main` to organize your code.

Deliverables and Grading

There are *four* deadlines for this project.

For the first deadline (**midnight, November 29**), you will have a meeting with your TA, and your deliverables are:

- the team contract;
- the conversation design;
- the client/server protocol;

This design deliverable should be submitted by committing one PDF to the `root` of your project repository.

During lecture on **November 30th** you will meet with your project TA discuss your design and client/server protocol.

For the second deadline (**midnight, December 6**), you will have another meeting with your TA, and your deliverables are:

- concurrency strategy;
- UI sketches (paper sketches);
- the testing strategy;
- and a demo of *some* working portion of the project that demonstrates significant effort towards understanding a critical or high-risk area of the design.

The code designs and testing strategy must be submitted by midnight on December 6 as one PDF to the `root` of your repository. The demo will take place at the meeting with your TA.

Your demo might show, for example, a basic server that sends and receives messages but without a GUI client. Or you might have a working basic GUI with no server backend but a simple API for connecting to one. Talk to your TA beforehand if you are unsure about what is sufficient.

You will meet with your project TA sometime **Dec. 7-9**. Be prepared to show UI sketches, present your demo, and discuss your design.

On **December 7th**, the staff may or may not release an amendment to this project. This will mean an additional requirement or feature to implement before the final deadline. When designing your instant messaging system, watch out for designs that will make extensions difficult.

For the third deadline (**midnight, December 14**), your deliverables are:

- the implementation;
- the tests;
- and the testing report.

The fourth and final deadline (**midnight, December 15th**) is the individual reflection.

The grading breakdown is as follows:

- 25% for the design, protocol, and usability design, and concurrency strategy
- 50% for initial demo and implementation
- 15% for testing strategy and testing
- 10% for team contract and reflections

Awards

The course staff will judge and award prizes to teams whose instant messaging systems embody exemplary design and implementation.

You may optionally **submit your project for prize consideration on Tuesday December 13.** There will be some time slots during the day for your team to present your system, which you can sign up for in advance. Your team will give a 5-minute presentation to the course staff in which you demonstrate your system and describe its design. You must commit your work (up to that point) to Subversion by 10 am on

December 13th. You are **not** required to give this presentation (but then you won't win anything, either). Everyone can continue to work on the project until the final deadline, but only the work demonstrated in this presentation will be considered for prizes.

Serious award contenders should consider going above and beyond the required specification to implement their own extensions.

You might add standard instant messaging features like away messages, auto-replies, offline messaging, password-protected accounts, user icons, graphical emoticons... or you might integrate voice chat, a shared whiteboard, encrypted conversations with perfect forward secrecy, or something as yet unheard of!

Hints

Defining a conversation. Part of your job is to determine what a conversation means. For example, does a conversation have a name, and can other users join the conversation by specifying the name? Is it like a chat room, that people can enter and exit? In that case, can a conversation be empty (a chatroom can), waiting for users?

Or is a conversation more like a phone call, where a person "dials" another person? In that case, can the receiving party deny the conversation?

However you define a conversation, remember to *keep it simple for your first iteration.* You can always extend your program with interesting ideas if you have time left.

Designing a protocol. You must also devise a client/server protocol for this project. You should strongly consider using a text-based protocol, which may be easier for testing and debugging.

Services that use plaintext protocols — e.g. HTTP or SMTP — can talk to a human just as well as another machine by using a client program that sends and receives characters. Think back to the protocol used in `telnet`. You can run `telnet` by opening a command prompt and typing `telnet hostname port`. The protocol is simple enough for humans to use and for machines to pass messages to each other.

Handling multiple clients. Since instant messaging is useless without at least two people, your server must be able to handle multiple clients connected at the same time. One reasonable design approach is using one thread for reading input from each client but adds a central state machine representing the state of the server (using one more thread, to which each of the client threads pass messages through a shared queue).

Design for safe concurrency. In general, making an argument that an implementation is free of concurrency bugs (like race conditions and deadlocks) is very difficult and error-prone. The best strategy therefore is to design your program to allow a very simple argument, by limiting your use of concurrency and especially avoiding shared state wherever possible. For example, one approach is to use concurrency only for reading sockets, and to make the rest of the design single-threaded.

And note that, even though user interfaces are concurrent by nature, **Swing is not thread safe**. Understand what code will run in the main thread, threads you explicitly spin, or the Swing event dispatching thread. Recommended reading: Threads and Swing.

Design for testability. To make it possible to write unit tests without having to open socket connections and parse streams of responses, you should design your state machine(s) in such a way that they can be driven directly by a unit test -- either by calling methods, or by putting messages into a queue read by the state machine's thread.

Testing GUIs is particularly challenging. Follow good design practice and separate as much functionality as possible into modules you can test using automated mechanisms. You should maximize the amount of your system you can test with complete independence from any GUI.

Another useful testing technique is the idea of a *stub* (method stubs, mock objects). To test one component of your system in isolation, you can create trivial implementations of the other components with which it is coupled. This might allow you to test your server without opening network connections, or to test your client backend with automated rather than GUI tests.

Implementation. Develop in iterations. Focus on important modules first, and defer making cosmetic improvements to your user interface until after all the code is well-organized and thoroughly tested. Make use of assertions.

6.005 Project 2

11/23

1st Meeting

Eric wants to do returnins

Do no work over break

Milestone 1

Arianna - Protocol

L XML - ready for change?

Plaz - Contract

Eric - Conversation Designs

Drafts 5PM Mon

Large Pieces

Server - Plaz

~~Backend~~ Backend Client - Eric

GUI client - Arienna

② Contract

Goals: Start w/ get good grade
Contest: talk Cator

Personal Goals: to learn things
have fun
good grade

Obstacles: time
tecnical issues
Part integration

2 say A - 1 says B
└ will get A if do all things well
always have fully functioning project

A: we'll see how much parts
ideal: world even split

(3) Meetings

Meetings during class time good - lectures

l else nessary

Doodle ?

Class time - for meetings when needed

eat during meetings

Minutes - I will do it

- SVN repo ?
- G Docs

Work

8 ? hrs / week

Unclear

mutually agree

Write down

set internal deadlines to review mtg

Case by case basis

Decision

Majority

Consensus better

Each person works on their part

If involves other people's parts - need their buy in

IM client

GM Skype

theplaz

Aa

Everyone use Windows

Brainstorm Conversation design now

⑤

Prof: Arbitary # users

Performance + Scaling irrelevant

Server:

- server just rotes packet

Presence - add, list

Conversation

∟ can you have multiple?

have to

Plaz	610	513	0390
Erick	646	285	3105
Airanna	978	505	8818

6.005 Project

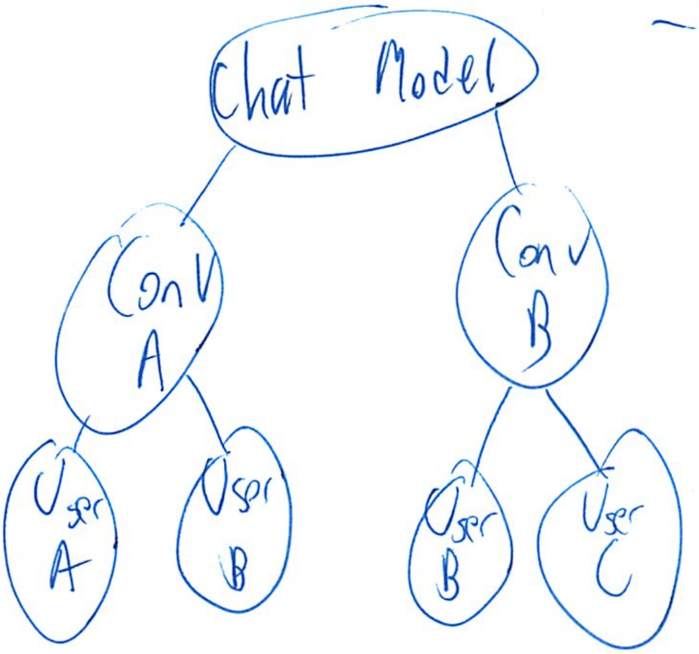
11/28

Do contract tonight - internally late

Eric : Conversation design

Need to be ready for change

- will release a change



Arianna

Same thing

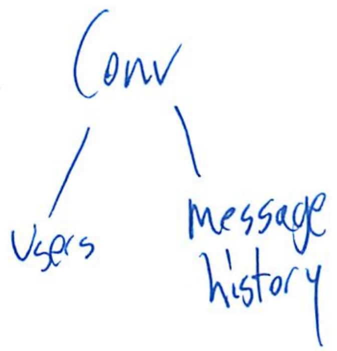
Server is just a passthrough for messages

still presence

Client or Server keep track who is in conversation

Message has multiple people in To field

So client



Can change things later

Message delivery notification?
└ not needed for now

Offline support - none?

Notify when someone comes online?

Push vs pull

don't care scability

If someone disconnects from sever send message

1 user per client

Use TCP - eaiser

When close window what happens

Send conversation list of users to all users on that conversation

~~Send~~

To ask people to join - one of us sends message to have someone join

Crappy networks not supported

Add conv + leave conv messages

To create conv - gen 40 char random nonce

L send Join conv to other party

Also have user visable ~~field~~ title 1st user sends

XML based messages

How to write?

Due tomorrow night

Conversation Design

- name classes
- Specs of public methods

- also for Server
  - presence list

Everyone online is a friend

When sever gets hello - pass along hello
Sends everyone list who is online

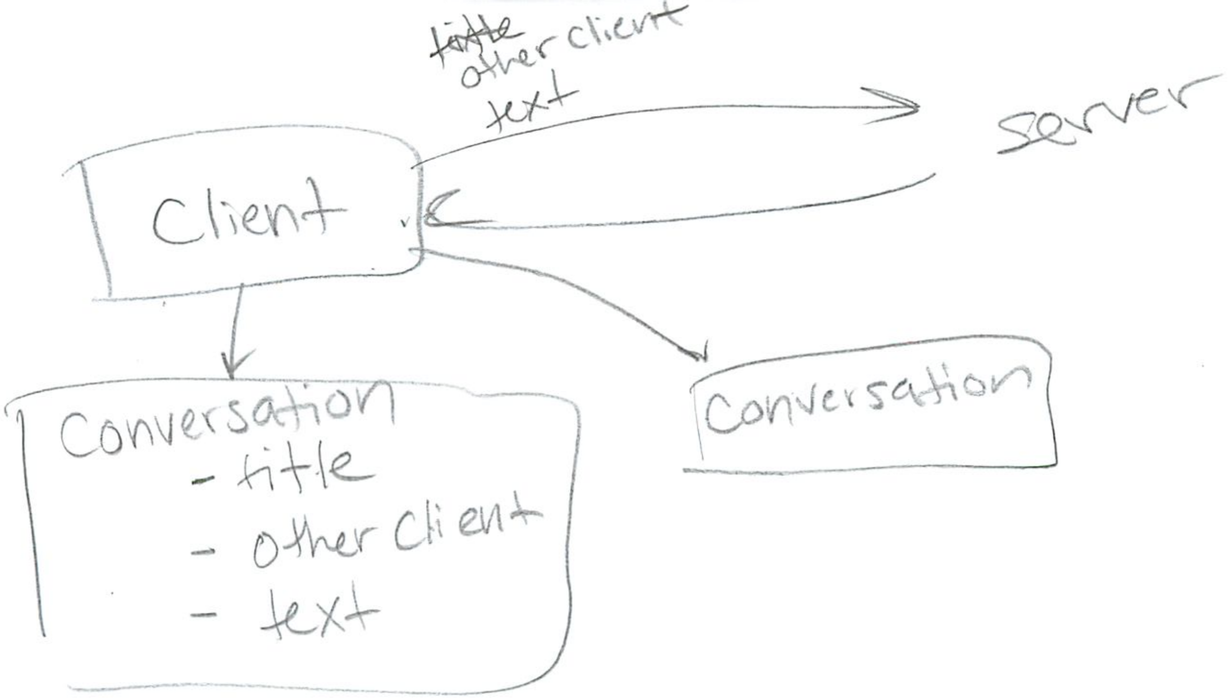
Enter username when hello
Server will error if username taken

Erick will do client + server

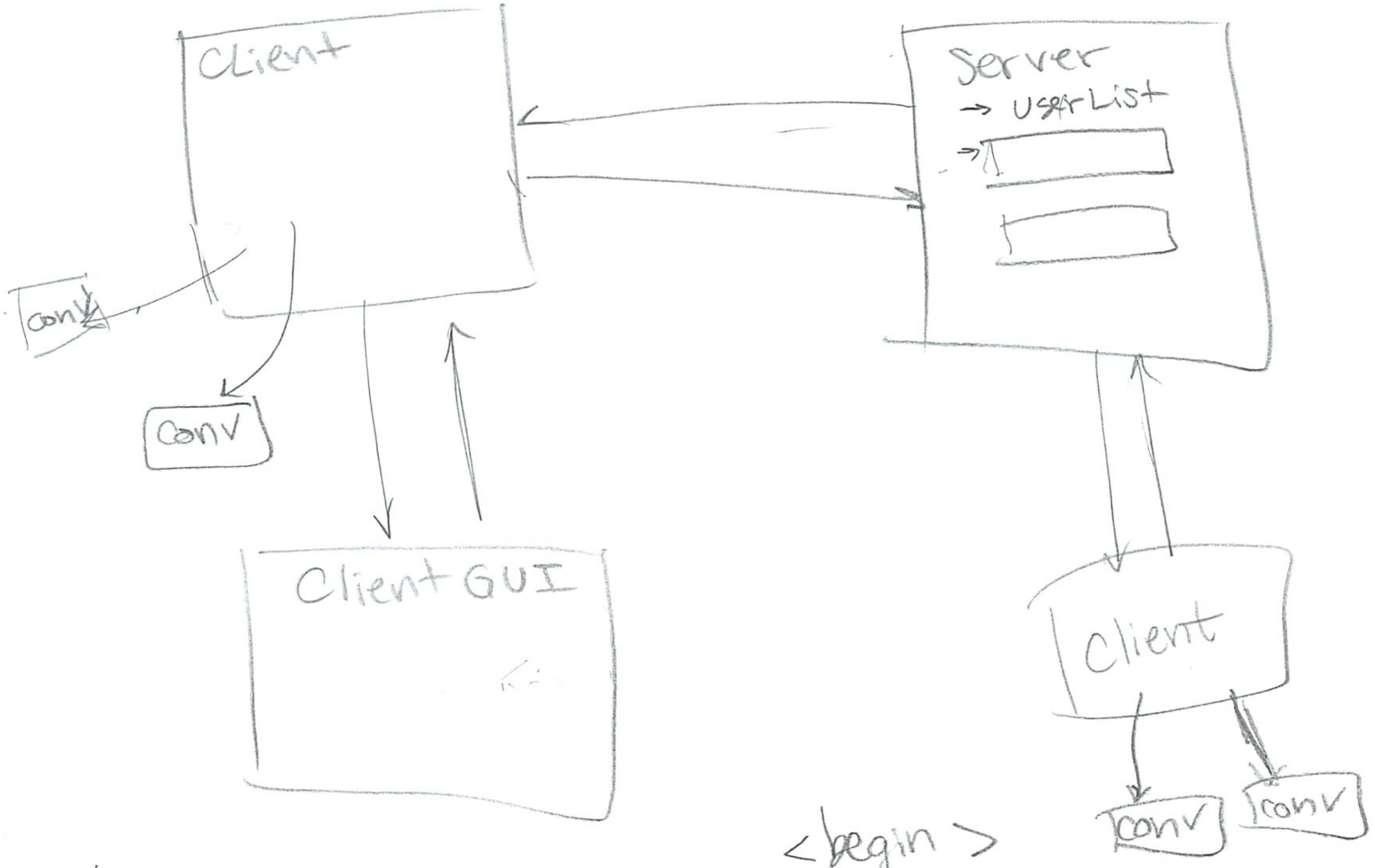
Also need snapshot diagram
- how objects ~~mae~~ move together
~~Add~~ - focus on how server is in middle

Snapshot diagram
- system
- client
- server

11/28


~~title~~
other client
text
server
Client
Conversation
- title
- other client
- text
Conversation

11/28


Client
Server
→ userList
conv
conv
Client GUI
Client
conv
conv

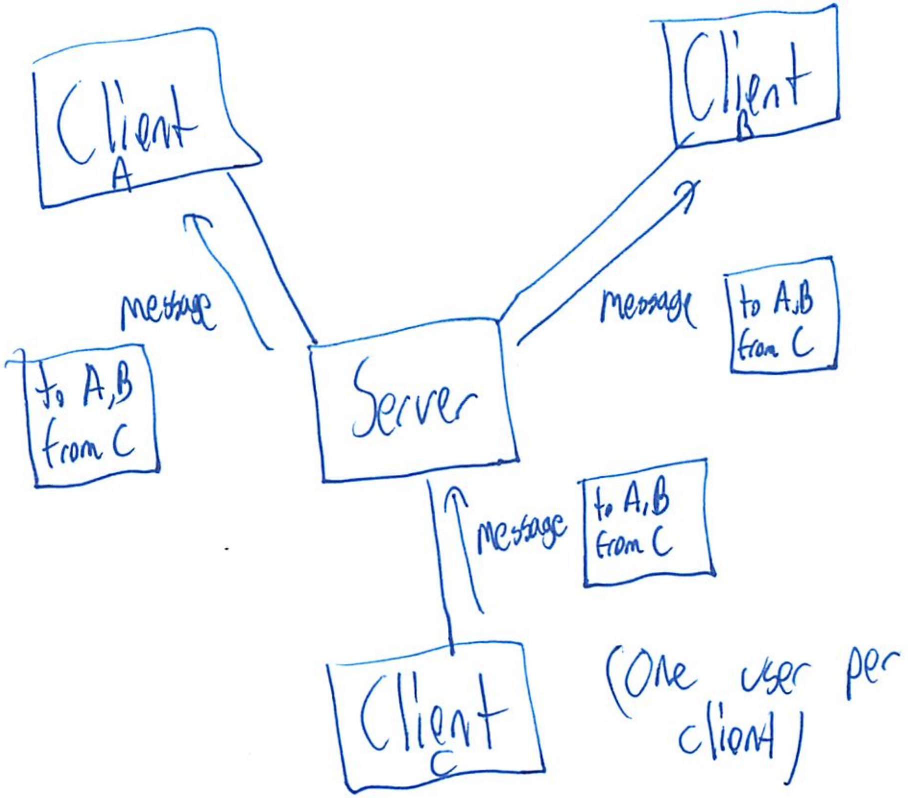
<message>

<begin>

<from> ~~~ </from>
<to> ~~~ </to>
<add user> ~ </addUser>
<title> ~~~ </title>

</begin>

11/29


Client A
Client B
Server
message
to A,B from C
message
to A,B from C
message
to A,B from C
Client C
(One user per client)

11/29

Server

Presence

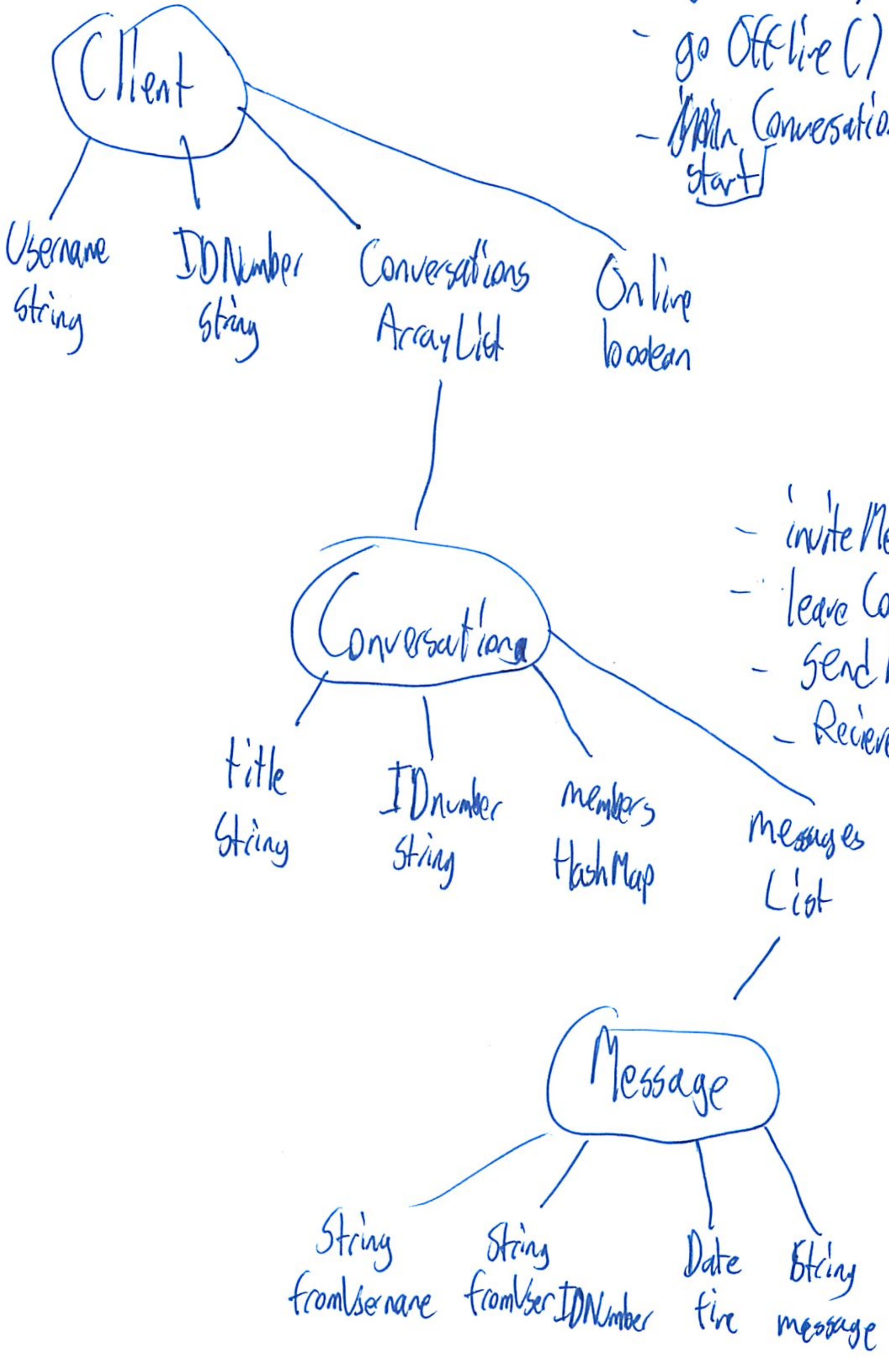
members
HashMap <String, String>

- addUser()
- removeUser()
- getUsers()

passes messeges through withat storing or converting to object

11/29

Client



- goOnline()
- goOffline()
- ~~Join~~ start Conversation()

- inviteMember()
- leave Conversation()
- Send Message()
- Recieve Message()

getters ...

PS4 Return in

11/20

Changing internal rep is so silly
Their internal rep is silly!

My SAT solver is wack
Why?
Did this so long ago!

It looks like lost pts since formla solver wrong

But why am I failing simple tests
The basic tests work
But not making null

Urg SAT solver so annoying!
Did it work before?
I feel like I need to fix this anyway

So $a \wedge b \wedge !a$
put fake a should be what?
(make it disappear?

JWang said junit much longer since some assertions
running in bg

Should do check comments on Caeser

Very annoying to have to do

Fixed most

Empty broke now I think...

6.005 Team Project 2 – Deliverable 1

ezuk-moshary-theplaz
11/29/2011

**NOTE: We all agree on the team contract, we just couldn't quite figure out the Adobe signatures…

1 Goals

1. Get a good grade: fully functional project highest priority
2. Do the project to the best of our ability without hindering our performance in other classes
3. If you really want an A, put in the effort
4. If you see something you don't like, talk to the person. If you can't work something out: fix it yourself.
5. We will talk about the contest later

2 Threats

1. Unclear instructions
2. Lack of time
3. Unforseen technical issues
4. Integration of components

3 Meetings

1. Meetings will preferably be held during class time
2. Anyone can ask that the group should meet
3. All team members should attend the meeting
4. If you miss a meeting, you get an angry email from the other team members
5. If the problem continues, the team members will talk to the course staff
6. Try to keep Skype open to chat during the other times

4 Work

1. Do not submit code that breaks compililation to the SVN repositiory
2. Ask for help if you need it
3. Work as many hours per week as our needed
4. Mutually agree on work distribution at later point
5. Write down what each person should work on
6. Set internal deadlines to review work before deadline

5 Decision Making

1. If you think a decision will make someone angry, ask them, and have a discussion
2. Majority rules, but a consensus is better
3. Each person works on their own part
4. If the decision involves someone else's part: ask them

ezuk ________________ *moshary* ________________ *theplaz* Michael Plasmeier

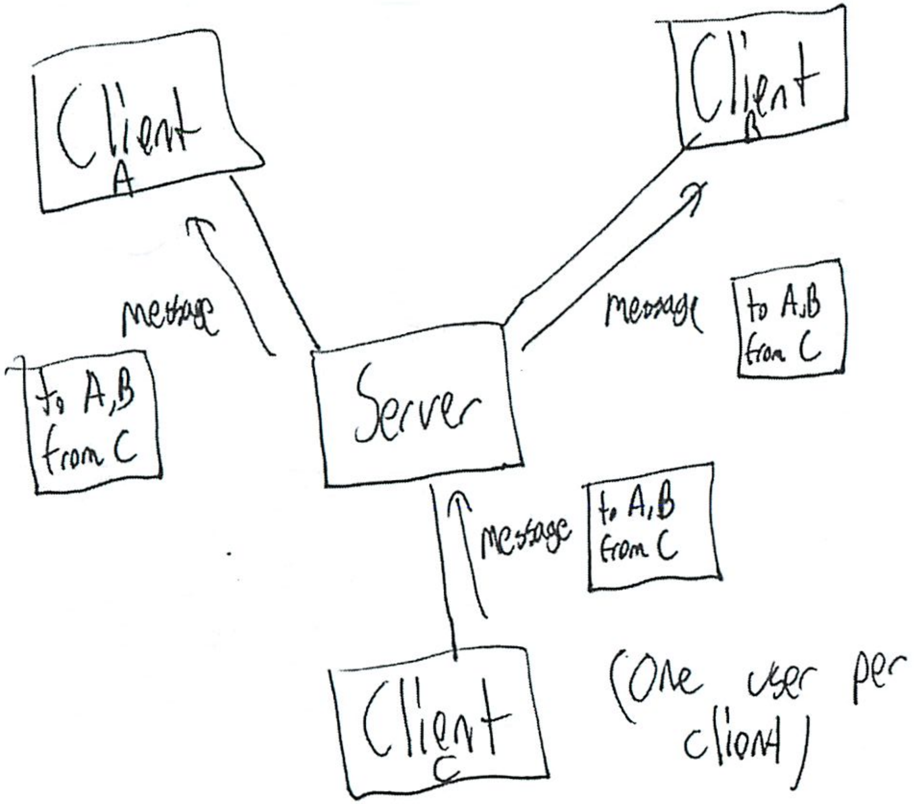
Digitally signed by Michael E Plasmeier
DN: c=US, st=Massachusetts,
o=Massachusetts Institute of Technology,
ou=Client CA v1, cn=Michael E
Plasmeier, email=theplaz@MIT.EDU
Date: 2011.11.29 21:18:33 -05'00'

Brief System Description

As modeled in our snapshot diagrams and java class layout, we will create the IM system through a series of connections between clients:

-All client interaction is relayed through the server, which parses Message objects from one client according to the client-server protocol, and sends the appropriate information to the intended recipients.
-The server will maintain a list of client IDs currently online.
-When a user wants to go online, he creates a new client, and offers a connect message to the server along with a desired username. If connected, he is given a unique client ID.
-A client can create a conversation object, which hosts a specific identification number, a user given name, and a list of members. A client must specify another client to join the conversation.
-If a client accepts an invitation to a conversation, they create a Conversation object on their local machine
-Each client in a conversation will have separate but identical ID Conversation objects hosted on their local machine. These conversation objects will be represented by separate windows in the GUI.
-A message sent from one client in a conversation will be sent to the server, where it will be distributed to the associated clients in the conversation, and to their appropriate Conversation objects.
-Disconnections from conversations and disconnections from Server will be represented by Messages sent to all Clients currently conversing with the disconnecting Client.

11/29



(One user per client)

11/25

Server

Presence

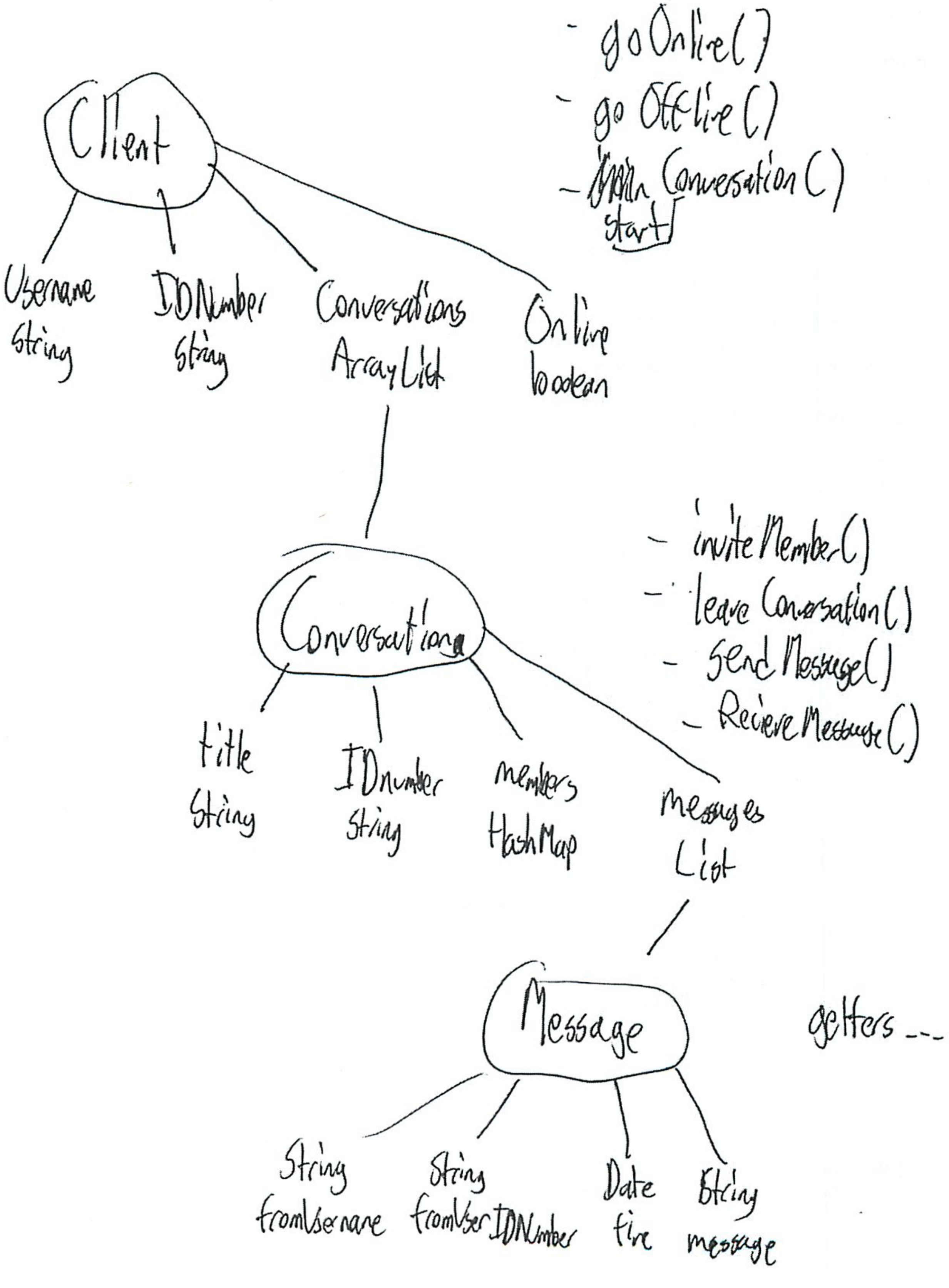
- add User()
- remove User()
- get Users()

members
HashMap <String, String>

passes messeges through without storing or converting to object

11/2c

Client



Client
Username
String
IDNumber
String
Conversations
ArrayList
Online
boolean
- goOnline()
- go Offline()
- ~~Open~~ Conversation()
start
Conversations
title
String
IDnumber
String
members
HashMap
messages
List
- inviteMember()
- leave Conversation()
- Send Message()
- Recieve Message()
Message
getters ---
String
fromUsername
String
fromUserIDNumber
Date
time
String
message

Client.java

```
package client;

import java.util.ArrayList;

/**
 * GUI chat client runner.
 */
public class Client {


    // user specified username
    private String username;
    // random generated user ID number
    private String IDNumber;
    // list of conversations that client belongs to
    private ArrayList<Conversation> conversations;
    // client online or offline
    private boolean online;

    /**
     * Constructor for Client object
     * @param username
     * @param IDNumber
     * @param conversations
     */
    public Client(String username, String IDNumber, ArrayList<Conversation>
conversations){
        this.username = username;
        this.IDNumber = IDNumber;
        this.conversations = conversations;
        this.online = false;
    }

    /**
     * Constructor for Client object
     * @param username
     * @param IDNumber
     */
    public Client(String username, String IDNumber){
        this.username = username;
        this.IDNumber = IDNumber;
        this.conversations = new ArrayList<Conversation>();
        this.online = false;
    }

    /**
     * Constructor for Client object
     * @param username
     */
    public Client(String username){
        this.username = username;
        //TODO: random generate ID number
        this.IDNumber = "";
        this.conversations = new ArrayList<Conversation>();
        this.online = false;
    }

    /**
     * Constructor for Client object
     */
    public Client(){
        //TODO: random generate ID number
        this.IDNumber = "";
        this.conversations = new ArrayList<Conversation>();
        this.online = false;

    }


    /**
     * Connect to chat with desired username
     * @param desiredUsername: desired user name for client
     * @modifies online: sets 'true' if specified user name not taken
     * @returns true if online;
     *          false if request denied
     */
    //TODO: implement
    public boolean goOnline(String desiredUsername){
        boolean returnValue = false;
        return returnValue;
    }



    /**
     * Disconnect from server
     * @modifies online: sets 'false' if disconnect successful
     *
     */
    //TODO: implement
    public void goOffline(){

    }



    /**
     * Start a new conversation
     * @param title: name of conversation
     * @param desiredUser: specified user to join conversation
     * @modifies existing conversations
     */
    //TODO: implement
    public void startConversation(String title, String desiredUser){

    }
    /**
     * Start a GUI chat client.
     */
    public static void main(String[] args) {
        // YOUR CODE HERE
        // It is not required (or recommended) to implement the client in
        // this runner class.
    }
}
```

Conversation.java

```
package client;


import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;


public class Conversation{
    // user-specified conversation name
    private String title;
    // random generated unique conversation ID
    private String IDNumber;
    // map of conversation members; Key->IDNumber, Value->Username
    private HashMap<String,String> members;
    // local text history of conversation
    private List<Message> messages;

    /**
     * Conversation object constructor
     * @param title
     * @param IDNumber
     * @param members
     */
    public Conversation(String title, String IDNumber, HashMap<String,String>
members){
        this.title = title;
        this.IDNumber = IDNumber;
        this.members = members;
        this.messages = new ArrayList<Message>();

    }

    /**
     * Conversation object constructor
     * @param title
     * @param IDNumber
     */
    public Conversation(String title, String IDNumber){
        this.title = title;
        this.IDNumber = IDNumber;
        this.members = new HashMap<String,String>();
        this.messages = new ArrayList<Message>();

    }

    /**
     * Conversation object constructor
     * @param title
     */
    public Conversation(String title){
        this.title = title;
        // TODO: must change to generate a random conversation ID hash-code
        this.IDNumber = "";
        this.members = new HashMap<String,String>();
        this.messages = new ArrayList<Message>();

    }



    /**
     * Invite a user to the conversation
     * @param clientID: client ID to be added to the conversation
     * @modifies this.members: adds clientID to list of members
     */
    //TODO: implement
    public void inviteMember(String clientID){

    }

    /**
     * Leave this conversation
     * Send a message to everyone
     * ?and close this object
     */
    //TODO: implement
    public void leaveConversation(){

    }

    /**
     * Remove a user from a conversation
     * (When we receive a message that the user has gone offline)
     * @param clientID: client ID to be removed from the conversation
     * @modifies this.members: removes clientID from list of members
     */
    //TODO: implement
    private void removeMember(String clientID){

    }

    /**
     * Send message within a conversation
     * @param conversationID
     * @param users
     * @param text
     * @modifies this.messages: adds message to list of messages
     */
    //TODO: implement
    public void sendMessage(String conversationID, ArrayList<String> users,
String text){

    }

    /**
     * Receive message to a specified conversation
     * @param conversationID
     * @param text
     * @modifies this.messages: adds message to list of messages
     */
    //TODO: implement
    public void receiveMessage(String conversationID, String text){

    }
}
```

Message.java

```
package client;

import java.util.Date;

/**
 * A single message
 */
public class Message {
    //from
    private String fromUsername;
    private String fromUserIDNumber;
    //time
    private Date time;
    //message
    private String message;

    /** Constructor
     *
     */
    public void Message(String fromUsername, String fromUserIDNumber, Date
time, String message) {
        this.fromUsername = fromUsername;
        this.fromUserIDNumber = fromUserIDNumber;
        this.time = time;
        this.message = message;
    }

    /**
     * Gets the user's username
     * @return String Username of the user that sent the message
     */
    //TODO: implement
    public String getFromUsername() {

    }

    /**
     * Gets the user's IDNumber
     * @return String UserIDNumber of the user that sent the message
     */
    //TODO: implement
    public String getFromUserIDNumber() {

    }

    /**
     * Gets the message's sent time
     * @return Date Timestamp of message
     */
    //TODO: implement
    public Date getTimestamp() {
    }

    /**
     * Gets the message's message
     * @return String text of message
     */
    //TODO: implement
    public String getText() {

    }

}
```

Server.java

```
package server;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Chat server runner.
 */
public class Server {
    // map of conversation members; Key->IDNumber, Value->Username
    private HashMap<String,String> onlineMembers;


    /**
     * Constructor for server object
     */
    public Server(){
        this.onlineMembers= new HashMap<String,String>();

    }

    /**
     * Adds a member to the list of online members
     * @param username: username to be added
     * @param userID: userID of username to be added
     * @modifies onlineMembers: adds the user to the map of online members
     *
     * Checks for duplicate usernames before adding
     *
     */
    //TODO: implement
    public void addUser(String username, String userID){

    }

    /**
     * Removes a user from the list of online members
     * @param username: username to be added
     * @param userID: userID of username to be added
     * @modifies onlineMembers: adds the user to the map of online members
     *
     */
    //TODO: implement
    public void removeUser(String username, String userID){

    }

    /**
     * Retrieves a copy of the list of online members
     */
    //TODO: implement
    public HashMap<String,String> getUsers(){

    }


    /**
     * Process a message from a client, and send it to the appropriate
clients
     * @param conversationID
     * @param users
     * @param text
     */
    public void processMessage(String conversationID, ArrayList<String>
users, String text){

    }




    /**
     * Start a chat server.
     */
    public static void main(String[] args) {
        // YOUR CODE HERE
        // It is not required (or recommended) to implement the server in
        // this runner class.
    }
}
```

Client to Server Protocol

```
Message ::= Begin
        (Connect|MessageBody|Disconnect|UserListReq|AddUser|LeaveConversation) End

Connect::= From ConnectMessage

Disconnect::= From DisconnectMessage

UserListReq::= From UserListReqMessage

MessageBody ::= From To+ Title Content

AddUser::= From To+ <addUser> User </addUser> Title

LeaveConversation ::= From To+ <leaveConversation> User </leaveConversation> Title

To::= <to> Text </to>

From::= <from> Text </from>

Title::= <title> Text </title>

Content::= <content> (Text| Newline | Tab)* </content>

Begin ::= <message version=1.0 type=
        (connect|messageBody|disconnect|userListReq|addUser|leaveConversation) >

End ::=</message>

Text:: = [^\n\t]*

UserListReqMessage ::= requesting user list

ConnectMessage::= connecting

DisconnectMessage::= disconnecting

Newline::=\n

Tab::=\t
```

Server to Client Protocol

Message ::= Begin (MessageBody | UserList | Welcome | FailedConnect | Goodbye | UserArrival
| UserDeparture) End

UserList::= FromServer <userlist> User* </userlist>

MessageBody ::= From To+ Title Content

Welcome::= FromServer welcome <userlist>User*</userlist>

FailedConnect::= FromServer <nameTaken>User</nameTaken>

Goodbye::= FromServer goodbye

UserArrival::= FromServer <userArrival>User</userArrival> Title

UserDeparture::= FromServer <userDeparture>User</userDeparture> Title

ChatArrival::= From To+ <arrival> User </arrival> Title

ChatDeparture::= From To+ <departure> User </departure> Title

User::=<user> Text </user>

To::= <to> Text </to>

From::= <from> Text </from>

Title::= <title> Text </title>

Content::= <content> (Text| Newline | Tab)* </content>

FromServer::= <from> server </from>

Begin ::= Begin ::= <message version=1.0 type= (messageBody | userList | welcome |
failedConnect | goodbye | userArrival | userDeparture) >

End ::= </message>

Text:: = [^\n\t]*

Newline::= \n

Tab::= \t

An Explanation of the Grammars

1) Client to Server: The client will send the following types of messages to the server
    a. Connect: A connect message request to the server. The From field will be read as a requested username. If it is taken, the Server will reply with a FailedConnection message and terminate the connection.
    b. MessageBody: An actual message to be sent to other users. There can be more than one designated recipient. The title will be a unique conversation identifier.
    c. Disconnect: A message to notify the server that this user will be leaving the chat program.
    d. UserListReq: A request to have the server send a current list of users.
    e. AddUser: Gets the server to send a notification that a new user is being added to a previously existing conversation.
    f. LeaveConversation: Gets the server to send a notification that a user leaving a previously existing conversation.

2) Server to Client: The Server will send the following types of messages to the Client
    a. MessageBody: A message from another user to this client. This has basically just been passed along by the server.
    b. UserList: A list of users currently connected to this server.
    c. Welcome: A confirmed connection to the server. User will receive the user list and can begin im-ing
    d. FailedConnect: The attempted user name is taken. Try to connect again.
    e. Goodbye: A confirmed disconnect from the server.
    f. UserArrival: A notification that a user has arrived on the server.
    g. UserDeparture: A notification that a user has left the server.
    h. ChatArrival: This user has been added to the chat with this title.
    i. ChatDeparture: This user has left the chat with this title.

Notes:

- We are including XML type tags to help with parsing
- Words in blue are terminals (actual strings to be sent)

6.005 Grp Mtg w/ TA

11/30

TA: elena_11

(where in wrong room)

talked out of username and ID

how to make conv ID unique

- suggested server does it - would need to track ids
  - assign
  - don't care if server reboots
- I said could do user + time stamp

local copy of presence list

new conversation is weird from UI view

- have seperate person to person and chat paradimes
- to be creating duplicate sigs

Post

nothing major

when 1 persons join - resend entire list

- could send update

②

Next deliverable

be able to parse

have basic message passing

client + server

just debug print

need UI sketches

Next meeting: Mon in lecture hall in class

Deliverable is next Tue

Stuff pretty much done by Mon meeting

So do message recieved, parse

- does not do something else

Arianna - UI sketches

Plaz - Testing strategy

Eric - Concurrency strategy

- challenge is model/view

6.005
Project 1

12/4

No one else has done anything yet

So we want testing strategy
and plan for server

So did a testing strategy

So build like MS server
(although ~~[illegible]~~ failed testing ...)

- allow args for port -shore !

- switch in Java?
- how to Parse

Wish had some concrete examples
and quotes

Well I could edit grammer

Need to come up w/ some way to test...

Can't switch on strings

Changing grammer

Just doing usernames!

¿Where to return data?

¿Mixing output and functions

L oh well

L try to make code clean?

Or seperate methods?

¿how to output multiple lines?

String[ ]

¿Can only connect from 1 server at once?

and Java will track

So not secure can disconnect anyone!

(Actually less fun w/ so few constraints)

③

Oh darn - need to be able to access other peoples' connections

So need list of collections...

Jnangi
list of outputs

Server class of Users

- Usernames
- Outputs

Collections, Syncronized List

Not very happy w/ fn structure

- but might be best have now

So still need to do conversations and message passing

(Our grammer - even still not that elegant)

(4)

¡But server should not handle conversations - right?¡

Right - just forward

So 1st draft server done¡

(Thorougly untested)

✓ testing starting

Now test like Juang did w/ Minsweeper

└ I send stuff to it

See what responds

Don't try to look inside

✓ Got first add user test

How do you test private methods

└ web¡ can do some setAccessible thing

But error adding second user

└ address already in use!

¡Oh need new client¡ → do tomorrow

6.005 Mtg

12/5

Plaz - Still need to test server
└ : new thread

Eric - Started client
└ Don't know to test
Will work more tonight

For tomorrow: working on client-server interaction

Arianna: look at concurrency
- add concurrency
- write strategy

between components
internal argument

(Eric + Arianna both failed PS6 as well)
└ Tests bad

Can convert user list to a map
└ silly inefficent format

No timestamps now?
- Server does not care
- but client should care

Client Concurrency; UI
- model will be single

  (whatever Harry Potter was)
  - Java UI code does that automaticaly

~~Client does not think about results~~

Server can just send info to client

We will see what threads we end up needing

Think we are close to having stuff
- need to put together

+ eric's client

Meet tomorrow 7:15 PM on Skype

6.005 Grp Meeting

12/6

TA time: 12:40 - 1 Stata 7th Gates Thur

Deliverables

Concurreny - not as detailed as Ariana would want

Can fix as we change

just list of users

GUI - swing handles

Code

Server code - should be implements

2 users can join

Message forwarding untested

Client - Now should be able connect

└ wait till talk to TA

└ using diff threads for C→S and S→C

Actually connect Thur - before mtg?

②

Should test mine more

- not really needed

Work + get done on weekend

Leric + Plaz busy till Fri

Testing Strategy - into the files

black box test Server

Unit test client

PDF put together

next Wed : Next deliverable

Arianna did designs - non Functional

- need title

- Sana message history : table ?
  bg color

Sun 5-6PM Baker 5th

Amendment

The design amendment is to incorporate a typing status for each user in a conversation. This functionality is implemented in Google Talk; the UI alerts you when your conversation partner is currently typing, or has entered text.

Each user should be in one of three states: *no_text*, *is_typing*, *has_typed*. *no_text* is the start state. A user can transition into *is_typing* by starting to type into the UI corresponding to the given conversation. A user transitions into *has_typed* if a certain amount of time has elapsed. This time window is up to you to decide. A user can re-enter the *is_typing* state from *has_typed* by typing again, or by backspacing over text they have already typed. Sending the current message will get you back to *no_text*. It is up to you to decide if deleting all the text can also get you back to *no_text*.

You can display this information as a change in the UI of the conversation (e.g. Alice is typing...), or into the buddy list UI (by changing the color of the user, or making the user italics, or ...). All three states should be easily distinguishable in your UI, regardless of how you choose to display this information to the user. This information has to be pushed to users by the server; it is not acceptable for users to be forced to click a button in order to get their partner's typing status.

Note that if user A is typing to user B in a conversation between only them, it shouldn't appear to user C that user A is in the *is_typing* state. Stated another way, user A should only be seen as typing by user C if user C is involved in a conversation with user A where user A is typing. Thus, you should logically have a state machine corresponding to each (user, conversation) pair.

6.005 TA Mtg

12/8

Status update

L said we just need to put pieces together

Fix Regex

L fix rest of server

(wanted to see ~~th~~ server working)

Show UI of conversation

Make ☒ as Leave Conversation

Start: specify host + port!
GUI

L can have default

Add typing indication

3 states

nothing
istyping
has typing

- participants list

Concurrency

Server: Synced list

Most groups buffer

L system call deadlock

depending on time

Now: select any message to see next

I really don't think it will make a difference

(Hammered hard to TA for that)

~~[illegible]~~

Clients 2 treads

Shared conversation ~~[illegible]~~ item

Update GUI

(leave at 1)

Working6.005

~~[illegible]~~ Project

12/11

So need to lock down testing problems

First the user name ~~[illegible]~~ regex

Oh not the regex - instead duplicate!

Often duplicate server

└ not closing down right?

Server still not shutting down correctly!

Need to do new users on new port

Or replaces other user

? since same port

New sockets better to test anyway

(Committing a lot - but like to do atomic pieces)

For some reason need a new port each time!

Java does not close!

even w/ System.exit!

oh now it looks to work

I think I need to improve testing Threading
to be able to access different Threads

So tried that - can I get it work?

✓ So it works for adding user

Now why not remove user

↳ is something actually wrong?

Thread dies when finishing running?

I think I am getting Thread ordering errors

Much better

Now getting actual errors I think

Oh really stupid!

③

Why does it hang on ~~getUsers~~? userListReq?

Otherwise it works supprisingly well!

~~Ah ha~~

It works on its own!

So weird!!!

An ext should be for each socket

└ then big mistake!

Yeah!

Now at least getting socket closed error

└ from where???

chat grp Meeting 12/10

My
Server errors - hanging

Eric: struggling
- harder then it looks
- Can't test
- Think thread safe

Arianna: putting in listener
- Client code missing
- Needs get + set methods for everything

(discussion) Arianna + Eric
Need to know when new conversation came in

Notify GUI

- Convo starting
- Buddy list updated
- Convo participants update
- (pretty much everything from server)

Back to Work

12/11

So stuck

Asked Jwang - he's too busy

No error message now

Try moving on + sending messages

└ now that line works

So its just the last item??

Still don't really know why - just add items

Now added more - still stuck there....

How did I fix it before?

Try debugging

What am I looking for?

Are we not parsing correctly...

Oh assume to is first...

That might have been it!

Now some add user flow?

Nice that it is ending now!.

2

Try putting ThePlaz in a thread
  L done
But its really eddie thread not starting!
  L it is - by ThePlaz not working - all the others work!

But why does ThePlaz work above
And the compare for user list works!
Oh may be an actual error
  L lines messed up
? Now prints right...
So simply ThePlaz out broke ??
  L but getOut is same as before when it worked!

So happens to ThePlaz again later
Then also happens to others
  L oh time limit runs out ?
    L since mine is first??

③ Sleep

Time makes no <u>difference</u>

They roughly die in order that they were added...

Still fail when comment out objects

Or sending message does not wal

Claims to work...

Adding time does nothing

Michael E Plasmeier

From:	njoliat@gmail.com on behalf of Nicholas Joliat <njoliat@MIT.EDU>
Sent:	Wednesday, December 07, 2011 11:15 PM
To:	Michael E Plasmeier
Subject:	Re: 6.005 announcement: ps6 grades
Follow Up Flag:	Follow up
Flag Status:	Flagged

Hi Michael,
The thing which I think overrides the ps4/ps6 issue is the need to be consistent with other decisions Rob has made w.r.t. grading of this particular pset and in general. I've made my decision based on that criteria, in spite of the fact that I sympathize with you for having lost a large number of points. If you would like to debate this further, please start a discussion with Rob instead. Good luck on the returnin and let me know if I can help with that; my office hours are 3-4pm tomorrow in 24-322.
Best,
Nick

On Wed, Dec 7, 2011 at 7:54 PM, Michael E Plasmeier <theplaz@mit.edu> wrote:
> Hi Nick,
>
> Thanks for looking over my code.
>
> As for consistency, I had a similar issue on PS4 (I had messed up some of the framing code so all automated tests failed). I talked to Eric Wong <eytw11@MIT.EDU> and he scheduled a time for me to come in and discuss my code. He then reviewed my code and issued me a regrade where he manually took off points for the flaws in my framing code and the actual flaws in my implementation. The regrade was higher than my original grade and it replaced my original grade. In addition, I received the tests I had actually failed so that I could fix my implementation and submit a returnin. The returnin was conducted with the published returnin policy.
>
> Could I ask what is causing the lack of consistency between PS4 and PS6?
>
> Thank you -Michael Plasmeier
>
> -----Original Message-----
> From: njoliat@gmail.com [mailto:njoliat@gmail.com] On Behalf Of
> Nicholas Joliat
> Sent: Wednesday, December 07, 2011 6:03 PM
> To: Michael E Plasmeier
> Subject: Re: 6.005 announcement: ps6 grades
>
> Hi Michael,
> I've looked at your code and there are a number of protocol-related problems.
> - You're not parsing the command line args correctly. As the pset indicates, there are always 1 or 3 args. Currently your code breaks if there's only one arg (you always look at args[1]) and also you treat the optional 2nd and 3rd arg as if they were one arg.
> - Your BOARD message that the server sends back is comma-separated; it should be space-separated; please review the protocol.
> I'm aware that this is a lot of points for these errors, but for the sake of consistency I can't do a special regrade of your code. I imagine the errors might be reasonably quick to fix, though, so consider submitting a returnin.
> Best,

> Nick
>
> On Mon, Dec 5, 2011 at 11:32 PM, Nicholas Joliat <njoliat@mit.edu> wrote:
>> Hi Michael,
>> I'll take a look at your code tonight or tomorrow and get back to you.
>>
>> On Mon, Dec 5, 2011 at 8:11 PM, Michael E Plasmeier <theplaz@mit.edu> wrote:
>>> When can I meet with you to discuss this?
>>>
>>>
>>>
>>> I don't know if you have a fixed time; otherwise my schedule is
>>> available
>>> here: http://doodle.com/theplaz
>>>
>>>
>>>
>>> Thanks
>>>
>>> -Michael
>>>
>>>
>>>
>>> From: Michael E Plasmeier
>>> Sent: Sunday, December 04, 2011 6:55 PM
>>> To: Nicholas Joliat
>>> Cc: Samuel Siyue Wang
>>> Subject: FW: 6.005 announcement: ps6 grades
>>>
>>>
>>>
>>> Hi,
>>>
>>>
>>>
>>> How did I get a 35 on the project? I must have done some little
>>> thing wrong. Can I get a manual regrade, like I got on ps4.
>>>
>>>
>>>
>>> -Michael
>>>
>>>
>>>
>>> From: Samuel Wang [mailto:samuelsw@MIT.EDU]
>>>
>>> Sent: Sunday, December 04, 2011 6:28 PM
>>> To: Samuel Siyue Wang
>>> Subject: 6.005 announcement: ps6 grades
>>>
>>>
>>>

>>> Note: This mail was sent to all students in the stellar class
>>> Software Construction
>>>
>>> ps6 grades
>>>
>>> ____________________________________
>>>
>>> I'm pushing ps6 grades in the next 5-10 minutes. Email Nick
>>> (njoliat@mit.edu) for questions.
>>>
>>> ____________________________________
>>>
>>> This announcement was made in Stellar on 2011 December 04 by Samuel
>>> Wang
>>>
>>> The announcement is also posted on the class website:
>>> https://stellar.mit.edu/S/course/6/fa11/6.005/

6.005 PS6

12/1

Return in

Oh can only be 1 arg

And it ~~doesnt~~ actually comes as 3 args

Space seperated is so stupid!

 L harder to read!

(I hate return in - so annoying)

Oh right blank and comma different!

Fixed tests - now they work

But do args work?

Why does server terminate now?

 L Ah open now

I should retest telnet - but don't want to now

 ? need athena or what did I do?

Ah Putty

(2) Not working w/ new lines — Dav:

└ but did not change anything

— test on Athena

Test on Athena

Seemed to work

One touble on 0,0

But could not reproduce

Sent it in to TA for unoffical 2nd look

└ akward to word

But worth a try

More 6.005 Chat Testing

12/4

Jwang suggests

```
while( BufferedReader.ready()){
    ; read()

}
```

Read B.R. docs...

But line should block if not ready

- └ oh that is why it hung...

But why is it returning null...

Am I reading lines early?

ThePlaz says ready...

- └ its always Ready ... w/ null! ☹

Or ThePlaz out gets messed up?

- └ try sending 2x

So it must be matched...

Or are we printing 2nd line when not needed?

└ like always print 2 lines?

So blocks when unused

Now it prints right ?!

└ or at least prints ...

Its on 4 + 5 for some reason ?!

Others null ...

So changing it not to send nulls

That seems to work better ☺

✓ Woot stuff passes

except 1

Now stuff is false ...

(I am amazed I always manage to figure this out..)

Why did commenting out not ~~[illegible]~~break it before?

It still hangs at the end/last one!

Never deal w/ servers going offline

└ will ignore?

So I think server works

except for

Is it blocking?

Or not responding?

Even when no respone it stops

And when remove item before it still ends

them

Ask tomorrow

6.005 TA
Help Mtg

12/12

(5 min late)

Ariana

Getting help on UI updating

Testing client

- model getting updated
- proper messages sent

(I am having trouble getting Stata online)

Now GUI can ref client

- Get it done
- Don't worry about fancy indirection

ServerTest hanging

- Never seen before
- Test says 21/21
- (could better test shutdown - all log off
  - and close conversation
  - shut outs PrintWriters + threads off

Online: Tests seem to run right
but then hang

If get rid of teardown it passes!

✓ Woot!

TA: lots of MagicText
- L other class - could be enumeration
- could still return text

Put stuff as constants

Done Typing

Could seperate out to individual classes
- L with a getMessage() method

Put Prefixes etc together at top of File
- or own enumeration

For Eric's testing use our server
(leaving now)

6.005 Returnin

PS6

12/13

TA got back to me

Two small errors

(1) – no quotes in help message

(where I exactly followed the specs!)

(2) "if a square is flagged a dig operation still suceeds. Digging a flagged square does not change the board"

I have no clue what that means

Review instructions

And what I do now

↳ don't seem to be doing anything...

(I liked this p-set)

i return F first on to String

run tests

↳ pass the same!

⓪

Ok emailed in for clarification

(This is so stupid - not learning anything - just fixing bugs!)

Replied

1st string looks fine

└ visable means dug

So revert that

Instead check if square flagged when dig

if flagged do nothing

So I think he meant have no effect

not just no visible effect

Rerun tests

└ pass ✓

add one to dig flagged state

(I actually liked this project more - so much simpler!)

6.005 Work

12/13

Ok now this project

See what changed (a lot)

Some changes w/ my part

Fix code ^ then

prettyness

- just w/ grammer
- remove printing

Well check tests first

So blocks on remove user

- need to make it match

Why do the other tests not fail ??

Oh project never refreshed in Eclipse

L even though I restarted it!

✓ Fixed tests

Still need to add some for disconnect

L and fix the other points

L no did leave conversation

Do some out of order - see if can handle

✓ Did w/ disconnect

Now rewrite test strategy

Or check to-dos

✓ Rewrite testing strategy for server

Group Meeting

Arianna: Not sure how to test client

Test system together

Current tests

- Server w/ fake client
- Client w/ fake server

New

- The two interacting w/o GUI
- GUI write up w/ screenshot

③

My to dos

1. Remae To-Dos

2. Beutify server

Arianna

Client should never close w/o message

↳unless force quit

¿ out of scope

A: Server does send a error + continues

I will spend 10-20 min on it

¿Can you send up an observe socket on same port

If A→B B→A then A ⊖ B?

¿Dummy GUI

What happens when Server → Client

It works!

Woot

¿Come back on w/ same user

(4)

Conv window only shows up once a message is sent

Bug lots of people on participants list duplicate

L will add a guard in the ~~[illegible]~~ client

~~Testing buddy list~~

~~Cls~~

Closing buddy list - does not close windows

L it needs to from a server perspective

(Testing stuff)

Username taken even when invalid

✓ Fixed

New line - cant w/ Shift + Enter
- too bad

/N just prints

Jumps only on my computers

Ariana spent 12 hrs on this yesterday

So duplicated conversations show in diff windows
when created

. 1 window for all others

Stat conv is gone

Are they globally unique?

Should not do

Or check if same people

Hmmm??

What happens when server goes ↓

18.111.105.225

Ok back to work for me

Finished Testing Strategy

Have 40 todos (TODO in code)

↳ They most have been done

⑥

(Very PM of me thinking about it...)

Now Thread Saftey Argument

↳ Thought I did this...

Only thing is online members;

; and server socket;

Try wrapping in Sync block

↳ Nope not there! - only 1 user can use it then!

Ok fixed it

Closing correctly

; notify people;

Arianna; try it

Will add it quickly + ughly

⊗ Can't find a way for close to be called

Arianna + Eric don't know

Can it

7

Now if client disconnects w/o saying anything

Piazza post

Oh can do out.equals?

So tried that

✓ Tests still pass

Sometimes when a person signs on they never hear back

Request one every so often

~~18.111.105.255~~

18.111.105.225

Take out printing

Force quit didn't work

└ take it out

Test w/ in test?

or done?

8

Bigest deal to do: Tests + Beutification

↑ I Finished on my section

↑ I really need to do

What did the TA suggest again?

- Get rid of magic text
- Could use classes
- Return text
- Put stuff as constants
- Individual classes
  - L w/ a get Message method
- Put prefixes togeter as enums

↗ do First

~~They get rid of an~~

Arianna: thy can be lower case

this is BS - worse design

Plaz: But TA said we have to

Can't switch on enum

This is so stupid!

✓ Woot Fixed + tests pass

✓ Same for remove user

Now next thing

get Message Header?

?seperate strings class?

Oh prob should do whole message
not just parts ...

✓ Done for addUser - do for rest

↳ boring

✓ Done for remove user

Do a parse from method

✓ Do for a bunch more

↳ ✓ Tests still pass

10

I think its a lot better

I fixed all the issues ...

(Actually I kinda agree its better)

Now just see if functions are in good order ...

Oh parse prefix

(Actually this is kinda fun

✓ Cleaned up more

✓ Still passes tests

email group

6.005 Lecture 12/14

Conclusion

□ Wrapup

□ Project awards

□ Quiz game

□ Course eval

Proj 2 due tonight

Proj 2 reflections due tomorrow night

Returnins due tonight

[illegible]

What was 6.005 about?

- building good software
- ready for change
  - abstract data types
  - design patterns
    - visitor - easy to add functions
    - interpretor - easy to add variants
    - listeners
    - monitor - locking
    - MVC
    - client - server
  - grammars
  - [illegible]
- safe from bugs
  - testing
  - static checking
  - [illegible] locking
- easy to understand
  - specs
    - preconditions
    - post conditions
  - statemachines

2

More patterns

Composite

lexer/parser / Abstract Syntax Tree

map/ filter/ reduce

Tech literecy

- JAVA
- IDE → Eclipse
- SVN
- Unit testing → Junit
- EclEmma
  - └ tells you if your tests cover code (never heard of...) (sounds interesting)
- Regex
- Map
- Lists
- Sets

3

- Streams
- threads
- locks

Other Classes

6.172 fall - Performance

6.813 spring - Usability

6.170 spring - dependability
+high level design
- Uses web programming

6.814 spring - databases

6.035 spring - compiler

IAP

6.370 - Battle code

6.470 - Web Programming Competition
- learn too

Summer

MEET - Teach Java to Isreli + Palestineans

MISTI - Many teaching experiences

Project Awards

Best theme - Dinochat

- makes dino sounds
- but not when you send certain words ;)
- tabbed interface

Technical Awesomeness - Glass Chat

(all techy team)

proven forward secrecy

diffy helmen key exchange

(large UI as well)

└ w/ command line arguments not

Best Features - Oh hai!

Oh UI looks very nice

└ repainted UI

Using blue gradient like Windows like theme

5 Can change user pic

~~Video chat~~

5 languages

Can change message font

Video chat

Line grap video chat

Best SW Design - Saiz Chat

Implements visitor patten

Bots that respond

- Weather
- Troll

Lots of packages

encode/decode patterns

Java Quiz

Course Evals

6.005 Project Review

12/14

On my own

Have we met all the requirements?

Still 35 TODOs Eric!

Looked over code
(Glanced)
- looks pretty good
- Eric has some magic strings
- but better than my originally

! Specs say server is required to maintain state of conversations

Lrutro!
- too late to refactor whole design
- argue is managing - by forwarding
  - TA kinda raised red flags
  - but not explictly
  - Perhaps thats why it seemed like a weird design choice...

Can't really join a conversation
- must be invited

②

Would that be Ok?

- Some groups did that
- normal IM systems do that

Need to update all the docs

- I did mine

My server is concurrent
tested

*Make sure it runs correctly

We did not do explicit thread buffering

I think black box testing is enough

I did Waterfall model for the server

6.005 Acceptance Meeting

12/04

Plaz: finished cleanup

Marco

Eric: lots of testing of client
  incoming typing in GUI - not model
  Threads have sleep
    └ machine dependent
  Todos almost all gone
    └ Failed connects + disconnects
      System just ~~[illegible]~~ closes
    Con ✓ tests - part of
      Client testing

Arianna:
  Wrote testing file - up
  Did everything could think of

  Add lines about unit testing?
  Updating design docs
    - design description
    - look + review

Each component should be right

②

Eric will change to be more like

TA told us stuff was Ok

Or just maintain list of convo in server for useless

└ Do that

and endConv?

just take startConv - don't pass on

✓ Fixed

Now

Design docs updating

Made some changes

✓ Done

Make main/runner class

Some last min bugs

└ added too much sync stuff ;

TA said our stuff was fine before

└ Acianna: don't take the extra stuff out

✓ Done Main

Eric typing thread saftey

then some specs

✓ Done; whole thing

6.005 Software Construction

Grade Report

Grade Report for Michael E. Plasmeier

Assignment/Exam Name	Graph	Due Date	Points	Max Pts
Problem Set 0	📊	09.08.2011	97.00	100.00
Problem Set 1	📊	09.15.2011	84.00	100.00
Problem Set 2	📊	09.22.2011	83.00	100.00
Problem Set 3	📊	09.29.2011	85.00	100.00
Problem Set 4	📊	10.11.2011	71.00 R	100.00
Quiz 1	📊	10.14.2011	79.00	100.00
Project 1	📊	10.27.2011	92.00	100.00
Problem Set 5	📊	11.03.2011	91.00	100.00
Problem Set 6	📊	11.10.2011	35.00 R	100.00
Problem Set 7	📊	11.17.2011	100.00	100.00
Quiz 2	📊	11.21.2011	91.00	100.00
Project 2	📊	12.14.2011	78.00	100.00

Instructor's Comments

Returning never listed for some reason

Final grade: A