

Archetype Games

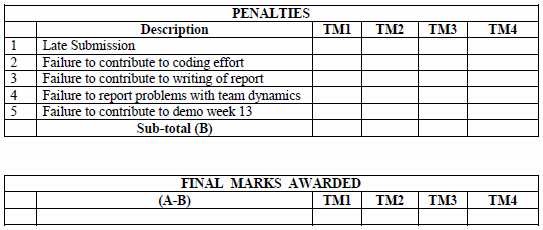
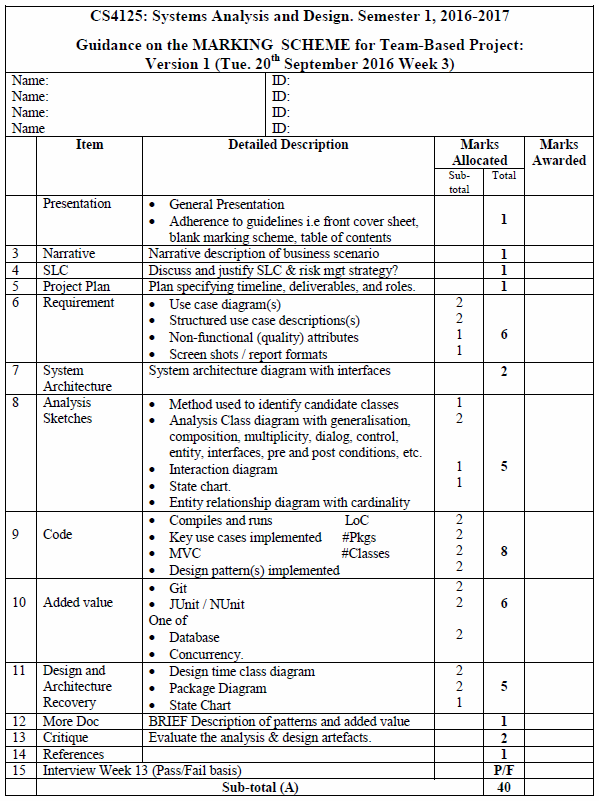
CS4125: System Analysis and Design

Dean Maloney (14140306)

Fergal Bowers (14151324)

Philip Waldron (14173026)

Pawel Pencherzewski (14199041)



Contents

[Narrative Description 8](#_Toc466886447)

[Overview 8](#_Toc466886448)

[What is Archetype Games? 9](#_Toc466886449)

[Software Lifecycle 10](#_Toc466886450)

[Waterfall Model 10](#_Toc466886451)

[V-Model 10](#_Toc466886452)

[Agile Software Development Model 10](#_Toc466886453)

[Project Plan 12](#_Toc466886454)

[Requirements 13](#_Toc466886455)

[Use Case Diagrams 13](#_Toc466886456)

[Use Case Descriptions 17](#_Toc466886457)

[Structured Use Case Descriptions 21](#_Toc466886458)

[Additional Non Function Requirements 29](#_Toc466886459)

[Discussion on Tactics to Support Quality Attributes 30](#_Toc466886460)

[Screen Shots 31](#_Toc466886461)

[System Architecture 35](#_Toc466886462)

[Discussion 35](#_Toc466886463)

[Architecture Decisions Taken 35](#_Toc466886464)

[UML Workbench 35](#_Toc466886465)

[Implementation Language 35](#_Toc466886466)

[Architecture Diagram 36](#_Toc466886467)

[Analysis Sketches 37](#_Toc466886468)

[Identify candidate classes 37](#_Toc466886469)

[User 37](#_Toc466886470)

[Player 37](#_Toc466886471)

[Game 37](#_Toc466886472)

[Leaderboard 37](#_Toc466886473)

[Lobbylist 37](#_Toc466886474)

[Friendlist 38](#_Toc466886475)

[Friend 38](#_Toc466886476)

[Chatroom 38](#_Toc466886477)

[Message 38](#_Toc466886478)

[Invite 38](#_Toc466886479)

[UI 38](#_Toc466886480)

[Log 38](#_Toc466886481)

[Class diagrams 39](#_Toc466886482)

[Interaction diagram 40](#_Toc466886483)

[State chart 41](#_Toc466886484)

[Entity relationship diagrams 42](#_Toc466886485)

[Code 43](#_Toc466886486)

[Overview of Contribution to Code 43](#_Toc466886487)

[Code Implementation 45](#_Toc466886488)

[ChatServer package - ChatRoom 45](#_Toc466886489)

[ChatServer 46](#_Toc466886490)

[FileServer package – FileServer 48](#_Toc466886491)

[FriendSearch 49](#_Toc466886492)

[LeaderboardSearch 52](#_Toc466886494)

[Query 53](#_Toc466886495)

[Search 55](#_Toc466886496)

[UserAdd 56](#_Toc466886497)

[UserSearch 57](#_Toc466886498)

[GameServer Package – GamePlayer 58](#_Toc466886499)

[GameServer 60](#_Toc466886501)

[Match 61](#_Toc466886502)

[MatchMaker 62](#_Toc466886503)

[Result 63](#_Toc466886504)

[ResultWriter 64](#_Toc466886505)

[Player Package – user 65](#_Toc466886506)

[Server Package – ServerFactory 66](#_Toc466886507)

[Chat Package – ChatRoomController 67](#_Toc466886508)

[ChatRoomModel 68](#_Toc466886509)

[Message 70](#_Toc466886510)

[Friend Package – Friend 71](#_Toc466886511)

[FriendInvite 72](#_Toc466886512)

[FriendListController 73](#_Toc466886513)

[FriendListModel 75](#_Toc466886515)

[GameInvite 77](#_Toc466886516)

[Game Package – Draw 78](#_Toc466886517)

[GameController 79](#_Toc466886518)

[GameModel 80](#_Toc466886519)

[GameReport 82](#_Toc466886520)

[Loss 83](#_Toc466886521)

[Win 84](#_Toc466886522)

[Leaderboards Package – LeaderboardController 85](#_Toc466886523)

[LeaderboardModel 86](#_Toc466886524)

[LogIn Package – LogInController 89](#_Toc466886525)

[LogInModel 90](#_Toc466886526)

[LogInMenu Package – MainMenuController 92](#_Toc466886527)

[Player Package – User 93](#_Toc466886528)

[UI Package – ChatRoomUI 94](#_Toc466886529)

[ChatRoomUIFactory 97](#_Toc466886530)

[FriendListUI 98](#_Toc466886531)

[FriendListUIFactory 100](#_Toc466886532)

[GameUI 101](#_Toc466886533)

[GameUIFactory 103](#_Toc466886535)

[LeaderboardUI 104](#_Toc466886536)

[LeaderboardUIFactory 109](#_Toc466886537)

[LogInUI 110](#_Toc466886538)

[LogInUIFactory 112](#_Toc466886539)

[MainMenuUI 113](#_Toc466886540)

[MainMenuFactoryUI 115](#_Toc466886541)

[UIFactory 115](#_Toc466886542)

[CS4125User Package – CS4125User 115](#_Toc466886543)

[Added Value 116](#_Toc466886544)

[Git 116](#_Toc466886545)

[Local Version Control Systems 116](#_Toc466886546)

[Centralised Version Control Systems 117](#_Toc466886547)

[Distributed Version Control Systems 118](#_Toc466886548)

[Concurrency 119](#_Toc466886549)

[Design and Architecture Recovery 120](#_Toc466886550)

[Architectural diagram 120](#_Toc466886551)

[Class Diagram 121](#_Toc466886552)

[State Chart 122](#_Toc466886553)

[Description of Patterns and Added Value 123](#_Toc466886554)

[Discussion 123](#_Toc466886555)

[Critique 124](#_Toc466886556)

[References 125](#_Toc466886557)

# Narrative Description

Overview  
The gaming industry has been undergoing a revolution over the past two decades, as highly-established companies such as Sony and Microsoft contribute quality products using the latest technology to gaming communities across the globe. Gaming on PC has also only increased in popularity over recent years through platforms such as Steam and Origin, and communities are much more plentiful on this platform due to the benefits of having a keyboard to type messages to fellow gamers.   
  
However, it is true to say that an alarmingly high amount of highly-anticipated software is regularly abandoned by the community for a variety of reasons. For example, with the downloadable content (DLC) structure becoming more and more popular, companies have focused on profits ahead of their community. By offering items which increase the player’s chances of being successful while playing the game in exchange for real-life money, dubbed as pay-to-win (P2W), developers alienate their community, leading to falling player bases and frustration amongst those who expected a satisfying multiplayer experience. Payday 2, an immensely popular online game, adapted a controversial P2W DLC schedule without consulting the community, leading to players permanently abandoning the game and outrage on online gaming communities such as Reddit and Steam forums. They were forced to rescind their changes a couple months later, but the damage had already been done. Pokémon Go similarly had a release plagued with server issues, and made the issue worse by not appointing a public relations officer to communicate with the player base. The game’s initial hype died very suddenly due to the resulting anger and confusion amongst the community who wanted answers.  
  
It is important to look back at how communities were successfully built surrounding the release of a multiplayer game and its legacy. Counter-Strike, a game that was originally a modification of the revolutionary Half-Life, was developed with the community in mind. Its player base soared, which prompted Half-Life’s developers, Valve, to develop a Counter-Strike series. To this day, all four of the main-series Counter-Strike games have a large number of regular online players and are regularly updated, with Counter-Strike: Global Offensive (CS:GO), the most recent entry in the series, being the fourth most popular streamed game on Twitch, the current leading game streaming website. While games which have a big impact at conventions such as E3 are often anticipated to become the “next big thing” to dethrone games such as CS:GO, they usually collapse due to repetitive gameplay or due to the aforementioned DLC issue. None of the Counter-Strike games implement a P2W system (CS:GO does include DLC weapon skins, but these do not affect gameplay), instead solely relying on the player’s skill and knowledge of the game. This has caused the game to be a massive hit in the competitive scene of eSports, a gigantic community of its own which American cable company ESPN has shown their interest in.

We at Archetype Games take inspiration from the successes and learn from the mistakes of the disasters throughout gaming’s incredible history. We seriously value public relations and the wishes of the community as we focus on implementing whatever project lies ahead of us. By supplying the player base with incentives such as leader boards, patches, competitions and regular news updates through our client service, we make the greatest effort to ensure the needs, wants and expectations of the loyal player are presented and exceeded. We also offer a platform for the community to communicate amongst each other, befriend each other and to clash in a competitive game. We value the skill of the player, and we ensure that they have every opportunity to display it.

# What is Archetype Games?

Archetype Games offers a client program that offers players numerous ways to engage with others in a competitive or friendly manner. It establishes itself as a hub for gaming and communication purposes, and it allows users to register and join a game, view leader board entries, enter a chat room or maintain a friends list.   
  
The client communicates with the server, which maintains data for each user and handles requests and matchmaking services.

# Software Lifecycle

The team considered numerous software lifecycle models during the planning stage of implementing the project, and we eventually came to the conclusion that the Waterfall Model and V-Model were not the correct approaches.

# Waterfall Model

We wanted to remain as flexible as possible in case we realized during the implementation phase that the resources cannot be dedicated to implementation a planned functionality. We instead sought a lifecycle model which allows revisions of design instead of forcing the implementation into the wrong direction.

-To quote David Parnas, “Many of the [system’s] details only become known to us as we progress in the [system’s] implementation. Some of the things that we learn invalidate our design and we must backtrack.”

With the Waterfall Model, there would have been no adaptability to any requirements that may be re-defined by client. If we wanted to make a change to the requirements once the design stage was completed, we would have no choice but to modify the design to allow the requirements change to occur. Dedicating resources towards this would not be worthwhile, and this flaw was a deciding factor as we decided against the Waterfall Model.

# V-Model

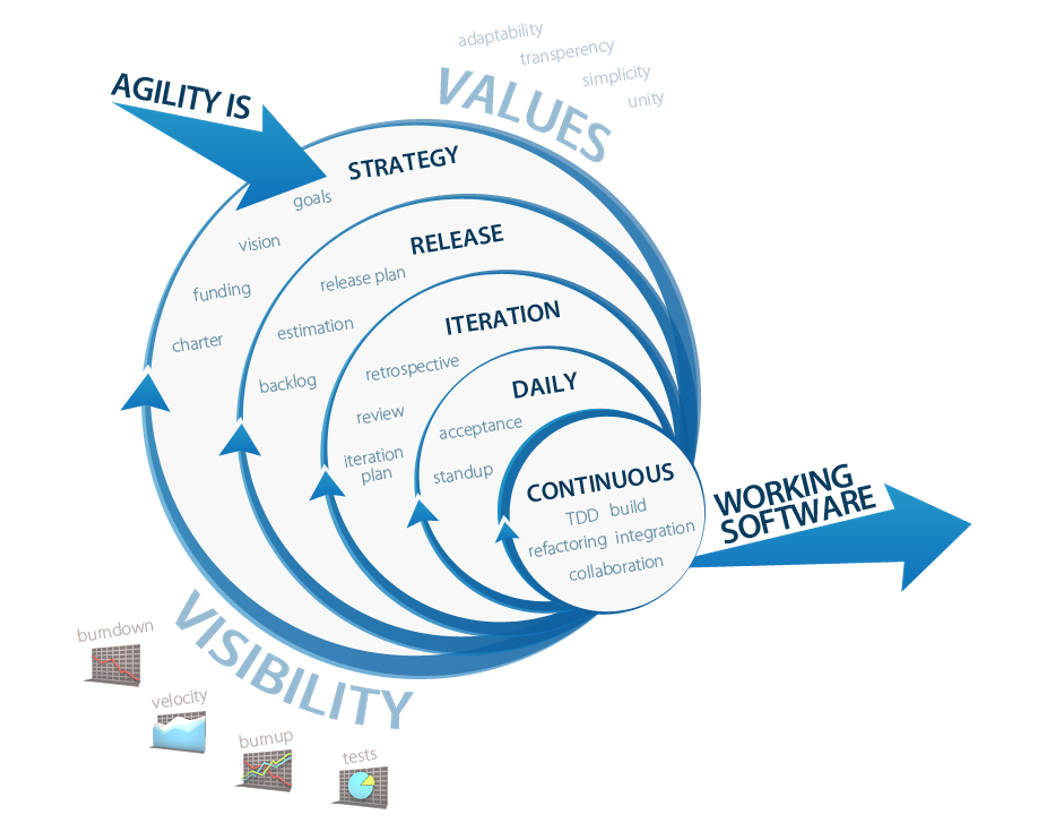
Since we wanted flexibility while implementing the project, we couldn’t ignore the V-Model’s rigid and linear nature. It does not natively allow us to respond to change, which would prove to be crucial during implementation. It also shares many similarities with the Waterfall Model and therefore maintains its drawbacks.

We wanted to be as efficient as possible once we reached the testing stage, and the V-Model does not allow for opportunistic approaches when it comes to finding the most effective way to execute testing our implemented project.

# Agile Software Development Model

Agile remained amongst our final choices for our software lifecycle model, and stood out as the optimal choice. The project’s nature required a software lifecycle model which was as flexible as possible and supported responding to change. The project also has plenty of room for expansion due to patches that will be released for the client throughout the project’s lifespan.  
  
The Agile Software Development Model was exactly what we needed to apply an iterative and incremental approach to development. It allows us to test and implement new features into the project to suit the user’s needs and to respond to public demand. The aim is to maintain a healthy community and player base for a long period of time, and Agile assists us in remaining successful in that regard.

A diagram follows which illustrates how we approached the development of our project under the Agile Software Development Model.

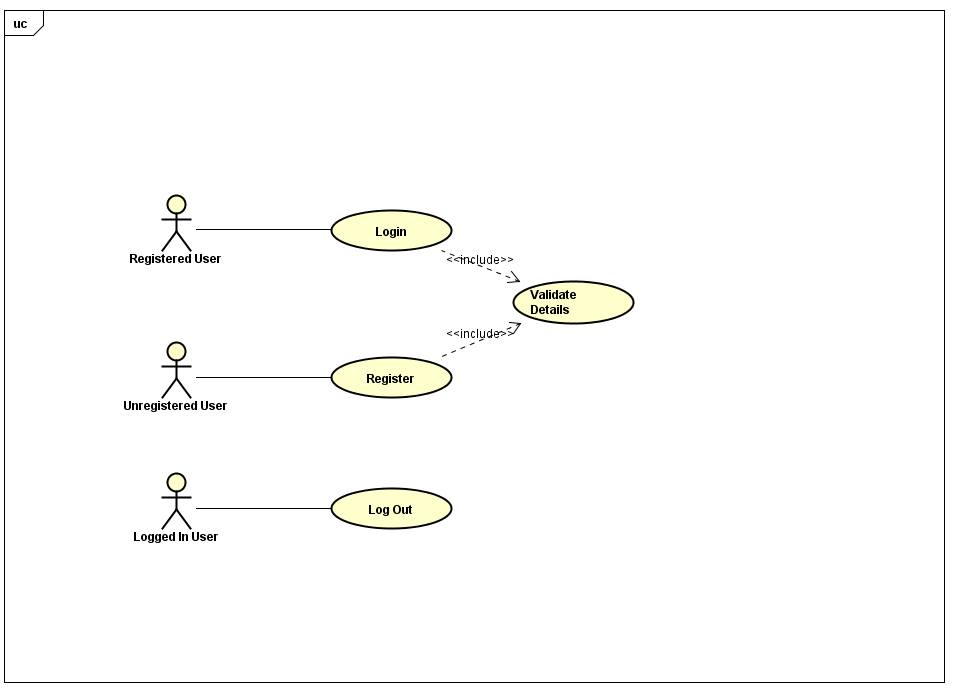


# Project Plan

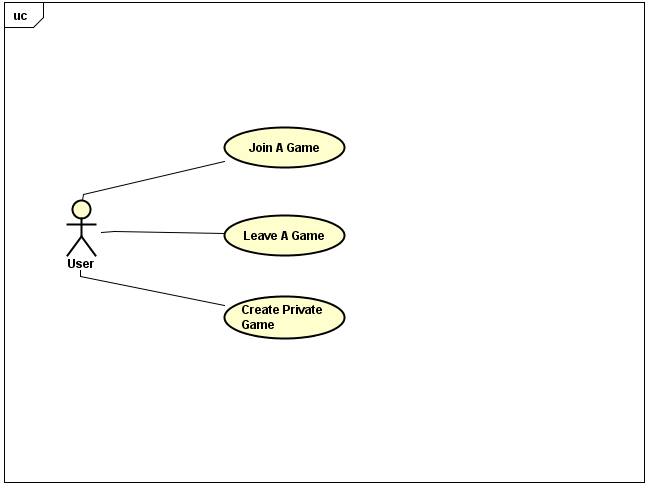
|  |  |  |  |
| --- | --- | --- | --- |
| **Heading** | **Description** | **Allocated to** | **Week** |
| Presentation | Company logo/design cover page | Philip | 7 |
| Narrative Description | Narrative description of business scenarios | Dean | 5 |
| Software Life Cycle | Discussion of software model used | Dean | 5 |
| Project Plan | Specifying jobs and roles | Philip | 4 |
| Requirements | Use case diagrams  Use case descriptions  Structured use case descriptions  Non-functional requirements  Tactics to support quality attributes  Screen shots of GUI | Dean  Fergal  Philip, Pawel  Philip  Philip  Group | 6  6  6  6  6  10 |
| System Architecture | Discussion  Architecture diagram with interfaces | Pawel  Group | 7  7 |
| Analysis Sketches | Identify candidate classes  Class diagrams  Interaction diagram  State chart  Entity relationship diagrams | Philip  Fergal  Pawel  Fergal  Pawel | 7  7  7  7  7 |
| Code | Overview of Contribution to Code  Code implementation | Philip  Group | 10  7 - 10 |
| Added Value | Git  Concurrency | Philip  Dean | 10  10 |
| Design and Architecture Recovery | Architectural diagram  Design time class diagrams  State chart | Group  Group  Group | 10  10  10 |
| Patterns and Added Value | Brief description of patterns and added value | Dean | 10 |
| Critique | Evaluate the analysis and design artefacts | Dean | 10 |
| References | Sources used for learning and information | Group | 10 |

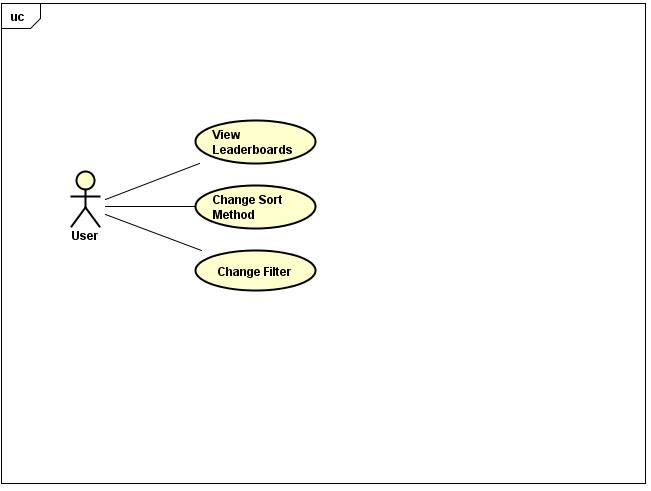
# Requirements

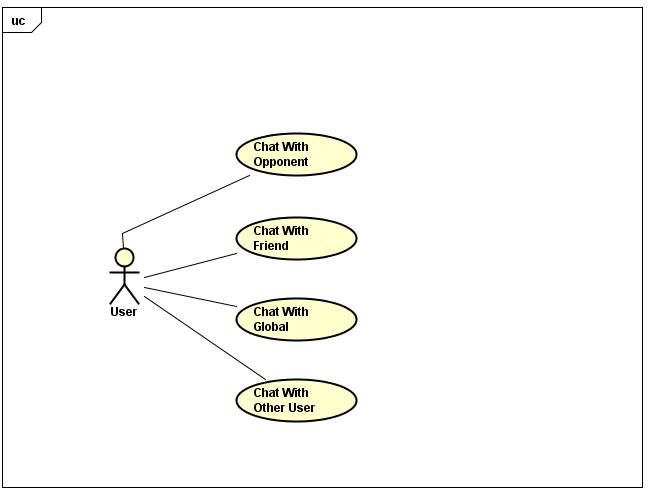
# Use Case Diagrams











# Use Case Descriptions

**Use Case 01:** Chat in Global

**Description:** A user sends a message to the global chat room

**Actors:** User

**Pre-Conditions:**

1. User is logged in.
2. User is connected to the global chat room

**Flow:**

1. User types a message into the text field under global chat.
2. User presses "Send".
3. The Message is send to the chat room.

**Alternate Flow:**

User types a message into the text field under global chat.

1. Request denied: Cannot connect to chat.

**Post-Conditions:**

1. User's message is added to the global chat log.

Non Functional Requirement: Performance  
The server should receive and send the message within 3 seconds.

**Use Case 02:** Chat in Private

**Description:** A user sends a message privately to another player

**Actors:** User, User

**Pre-Conditions:**

1. User is logged in.
2. Other user is online.

**Flow:**

1. User selects other user from playerlist.
2. User selects "Private Message".
3. New chat window is created.
4. User types a message into the text field under private chat.
5. User presses "Send".
6. Message is sent to other user.

**Alternate Flow:**

1. User presses "Send".
2. Request Denied: Other User is offline.
3. New chat window is created.
4. If previous messages were sent between users, populate chat window with past messages

**Post-Conditions:**

1. Users message sent to other user

Non Functional Requirement: Performance  
The server should receive and send the message within 3 seconds.

**Use Case 03:** Chat with Friend

**Description:** A user sends a message to a user on friends list.

**Actors:** User

**Pre-Conditions:**

1. User is logged in.
2. Other User is online.
3. Other user is on user's friends list.

**Flow:**

1. User selects other user from friends list.
2. User selects "Private Message".
3. New chat window is created.
4. User types a message into the text field under private chat.
5. User presses "Send".
6. Message is sent to other user from friends list.

**Alternate Flow:**

1. User presses "Send".
2. Request Denied: Other User is offline.
3. New chat window is created.
4. If previous messages were sent between users, populate chat window with past messages

**Post-Conditions:**

1. User’s message sent to other user from friends list.

Non Functional Requirement: Performance  
The server should receive and send the message within 3 seconds.

**Use Case 04:** Chat in lobby

**Description:** A user sends a message to all users in game lobby

**Actors:** User

**Pre-Conditions:**

1. User is logged in.
2. User is in a game lobby.

**Flow:**

1. User types a message into the text field under lobby chat.
2. User presses "Send".
3. The Message is send to the lobby chat room.

**Alternate Flow:**

User types a message into the text field under lobby chat.

1. Request denied: Cannot connect to chat.

**Post-Conditions:**

1. User's message is added to the lobby chat log.

Non Functional Requirement: Performance  
The server should receive and send the message within 3 seconds.

# Structured Use Case Descriptions

|  |  |  |
| --- | --- | --- |
| **USE CASE 05** | View Leaderboards | |
| **Goal in Context** | The leaderboards are displayed to the user in another window | |
| **Scope & Level** |  | |
| **Preconditions** | None | |
| **Success End Conditions** | The leaderboards are displayed | |
| **Failed End Condition** | The leaderboards are not displayed | |
| **Primary, Secondary, Actors** | User | |
| **Trigger** | ViewLeaderboards request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User selects View Leaderboards button |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 1a | User wants to view leaderboards:  1a. A new window opens, populated with all registered users |

Non Functional Requirement: Usability  
The UI should be easy to use for the intended user.

|  |  |  |
| --- | --- | --- |
| **USE CASE 06** | Change Sort Method | |
| **Goal in Context** | The leaderboards are displayed in a desired sorted order | |
| **Scope & Level** |  | |
| **Preconditions** | Leaderboards are currently displayed | |
| **Success End Conditions** | Sort method is changed | |
| **Failed End Condition** | The sort method is not successfully changed | |
| **Primary, Secondary, Actors** | User | |
| **Trigger** | ChangeSortMethod(sort) request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User selects the method in which to sort the leaderboards |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 1a | Sort method changed:  1a. The leaderboards are repopulated in the relevant order |
| **VARIATIONS** | **Step** | **Branching Action** |
| 1 | User may sort by:  Alphabetical order  Most wins  Highest win/loss ratio |

Non Functional Requirement: Extensibility  
More sort methods should be easy to add in future iterations.

|  |  |  |
| --- | --- | --- |
| **USE CASE 07** | Change Filter | |
| **Goal in Context** | A filter is applied and the leaderboards are displayed taking into account the new filter | |
| **Scope & Level** |  | |
| **Preconditions** | Leaderboards are currently displayed | |
| **Success End Conditions** | Leaderboards are displayed in relation to the filter | |
| **Failed End Condition** | Leaderboards are not filtered correctly | |
| **Primary, Secondary, Actors** | User | |
| **Trigger** | ChangeFilter request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User types in a filter |
| 2 | User selects a button to apply the filter |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 2a | Filter is applied:  1a. The leaderboards are repopulated taking to account the filter |

|  |  |  |
| --- | --- | --- |
| **USE CASE 08** | Add a Friend | |
| **Goal in Context** | Causes a pop up window to appear where the user can enter a name of another user to add as a friend. | |
| **Scope & Level** |  | |
| **Preconditions** | We know: adding user and the user being added | |
| **Success End Conditions** | One user successfully sent a friend invite, the other received it | |
| **Failed End Condition** | The invitation has not been sent/ the end user has not received an invitation | |
| **Primary, Secondary, Actors** | User sending and user receiving a friend invitation | |
| **Trigger** | AddFriend request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User sends a friend invitation by entering a name |
| 2 | The other user receives a friend invitation through “respond to invite” tab |
| 3 | Receiving user can accept or decline the invitation |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 1a | A user does not exist:  1a1. Display error and give another try |

Non Functional Requirement: Usability  
It should be easy to add the intended friend.

|  |  |  |
| --- | --- | --- |
| **USE CASE 09** | Remove a friend | |
| **Goal in Context** | Causes a pop up window to appear where the user can enter a name of a user he/she wants to remove from friends list and him/herself from the other user’s f/l. | |
| **Scope & Level** |  | |
| **Preconditions** | We know: the user removing and the user being removed | |
| **Success End Conditions** | A user successfully removed another user from their friends list and removed himself from their f/l. | |
| **Failed End Condition** | The user entered to be removed from the friends list has not been removed. | |
| **Primary, Secondary, Actors** | User removing a friend and the friend being removed | |
| **Trigger** | RemoveFriend request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User enters a name that he/she wishes to remove from friends list. |
| 2 | Both users’ friends list are updated, removing each other from them. |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 1a | A user does not exist:  1a1. Display error and allow another try |

Non Functional Requirement: Usability  
it should be easy to remove the intended friend.

|  |  |  |
| --- | --- | --- |
| **USE CASE 10** | Invite to a Game | |
| **Goal in Context** | Causes a pop up window to appear where he/she can enter a name of another user to send a game invite to. | |
| **Scope & Level** |  | |
| **Preconditions** | We know: inviting user and the user being invited | |
| **Success End Conditions** | One user successfully sent a game invite, the other received it | |
| **Failed End Condition** | The invitation has not been sent/ the end user has not received an invitation | |
| **Primary, Secondary, Actors** | User sending and user receiving the game invitation | |
| **Trigger** | InviteToGame request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User sends a game invitation by entering a name |
| 2 | The other user receives a game invite through “respond to invite” tab |
| 3 | Receiving user can accept or decline the invitation |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 1a | A user does not exist:  1a1. Display error and allow another try |
| 1b | A user is not online:  1b1. Display error and allow another try |

|  |  |  |
| --- | --- | --- |
| **USE CASE 11** | Respond to Invite | |
| **Goal in Context** | The user is displayed a list of friend and game invitations, which he/she can accept, decline or ignore. | |
| **Scope & Level** |  | |
| **Preconditions** | We know: the user, his/her invitations | |
| **Success End Conditions** | All invitations are displayed correctly and the user is able to respond to them successfully.  Responding to an invite removes it from the list. | |
| **Failed End Condition** | The invitations don’t display or are displayed incorrectly.  The invitations do not respond to actions taken. | |
| **Primary, Secondary, Actors** | The user viewing the invitations | |
| **Trigger** | RespondToInvite request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User presses a Respond to Invite tab/button. |
| 2 | A list of invitations for this user is displayed |
| 3 | Accepting/declining an invitation updates the invitation list. |
| 4 | If user accepted friend invitation the friend lists of both users are updated |
| 5 | If user accepted a game invite he/she is sent to a game screen. |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 5a | User sending invitation is offline:  5a1. Display information that user is offline |

Non Functional Requirement: Performance  
Friend list updates should occur within 3 seconds.  
Game lobby should be joined within 3 seconds.

|  |  |  |
| --- | --- | --- |
| **USE CASE 12** | Nickname Friend | |
| **Goal in Context** | User adds a nickname beside another friend’s username in his/her friend list. | |
| **Scope & Level** |  | |
| **Preconditions** | We know: user adding nickname, the names on his f/l | |
| **Success End Conditions** | User successfully appended a nickname to a friend’s username in his/her friend list. | |
| **Failed End Condition** | No nickname is displayed beside the friend that the nickname was added to. | |
| **Primary, Secondary, Actors** | The user adding the nickname | |
| **Trigger** | AddNickname request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User presses Nickname a friend button |
| 2 | User types in the name of a friend he wants to add a nickname to |
| 3 | User accepts changes made |
| **EXTENSIONS** | **Step** | **Branching Action** |
| 2a | A user does not exist:  2a1. Display error and allow another try |
| 3a | The nickname entered is too long:  3a1. Display error and exit “nicknaming” action |

Non Functional Requirement: Usability  
it should be easy to nickname the intended friend.

|  |  |  |
| --- | --- | --- |
| **USE CASE 13** | View Friend Availability | |
| **Goal in Context** | User is displayed with a list of his/her friends, as well as their online status. | |
| **Scope & Level** |  | |
| **Preconditions** | We know: A user, his/her friends and their online statuses | |
| **Success End Conditions** | The friends are correctly displayed with correct online statuses for every individual user | |
| **Failed End Condition** | The friends are displayed incorrectly or not at all.  The online statuses are incorrect | |
| **Primary, Secondary, Actors** | User viewing his/her friends list. | |
| **Trigger** | ViewFriends request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
| 1 | User presses View friends button/tab. |
| 2 | A list of all friends is displayed to the user with correct online statuses |

# Additional Non Function Requirements

The main menu should be extensible.  
It should be easy to add more functions on future iterations.

The program should support maintainability.  
It should be easy to fix defects, bugs and keep the service operational.

The program should be robust.  
Modules should be able to cope with errors during execution.

The program should be of good quality.  
It should be fit for its intended uses.

# Discussion on Tactics to Support Quality Attributes

We decided on writing the program in the Object Oriented Language Java. This was due to our familiarity with the language and its ease of portability across multiple platforms.

Java will also support our graphical requirements, as the primary objective of the system is to provide an extensible and reusable system that may be applied to other architectures and systems on various platforms. This also rules out the need for languages like C++ as while its speed may offer more power for graphical rendering, our application does not need this.

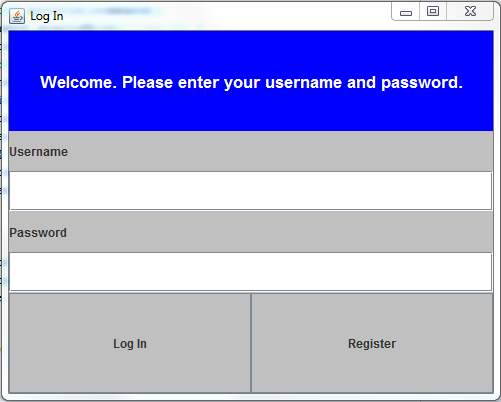
Our user interface should be easy to read and use for those familiar and unfamiliar with our application and those similar, opening up the program to a wider audience. In turn, the low system requirements to run Java proves it to be an excellent candidate.

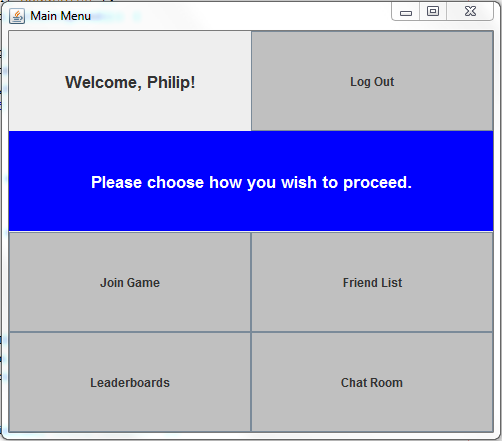
With access to the Java virtual machine on virtually every system, Java helps support our software to be highly portable and mobile.

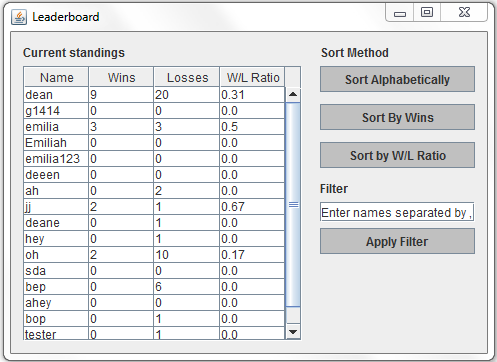
While ideal, we decided not to use an SQL database for our system as its database requirements are minimal and text file will do, keeping in mind that in a future iteration the program may be modified to use an SQL database as data accessing and storage requirement grow.

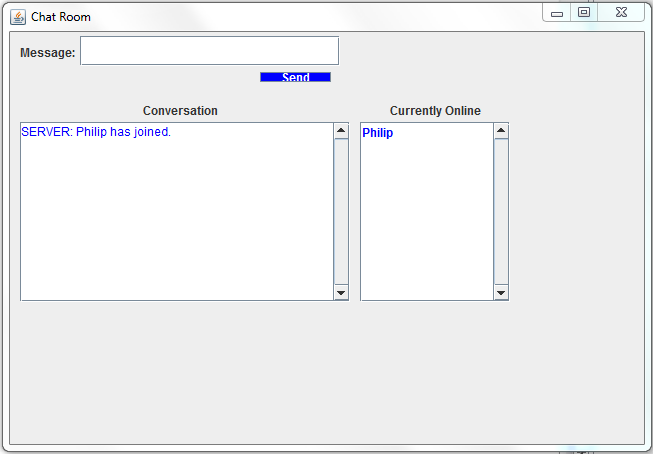
Our server will also use Java as our group has recently become familiar with Java socket programming. With keeping to Java for both system, it could be easier to open up private server hosting for clients in the future.

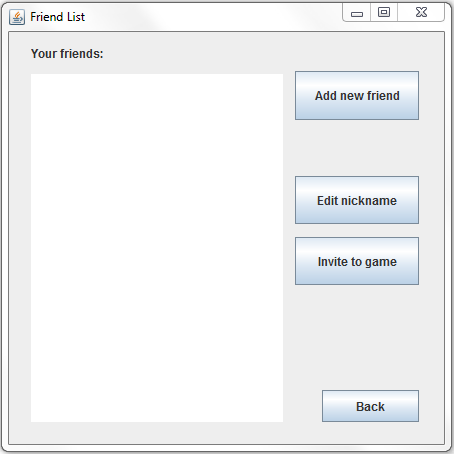
# Screen Shots

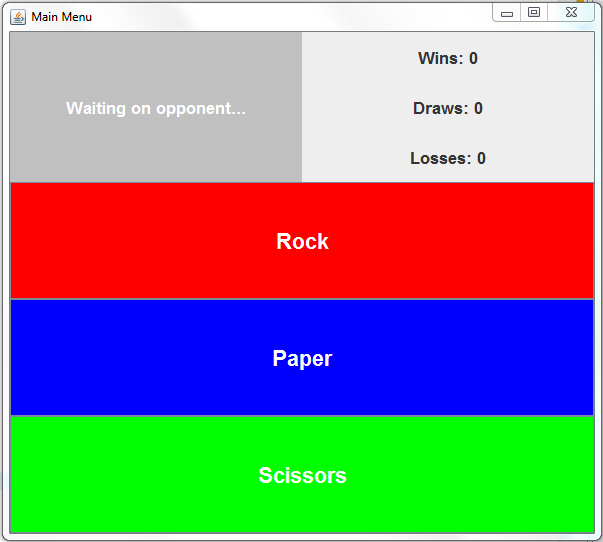












# System Architecture

# Discussion

# Architecture Decisions Taken

We will have two sub-systems: the client side and the server side, each running independently. The client will send messages to the server which in turn will respond with requested information or data.

Most of the business logic will be dealt with by the server with access to the text files, which store data. The server will perform the main functions of the application as it manages and processes requests from multiple users. It will consist of multiple packages including: ChatServer, FileServer and GameServer. Each “sub-server” will utilize threads, which with the use of text files (used as databases collecting data about each individual player) and messages, maintain contact with the clients and provide any information requested from them. The ChatServer for example utilizes threads and exchanges messages to add users into chatrooms. The FileServer mainly uses and manipulates text files with user information such as friends and scores to display friend list or leaderboard.

Factory methods are used for the user interfaces in the user session. MVC will be used to design the UI structure as well.

# UML Workbench

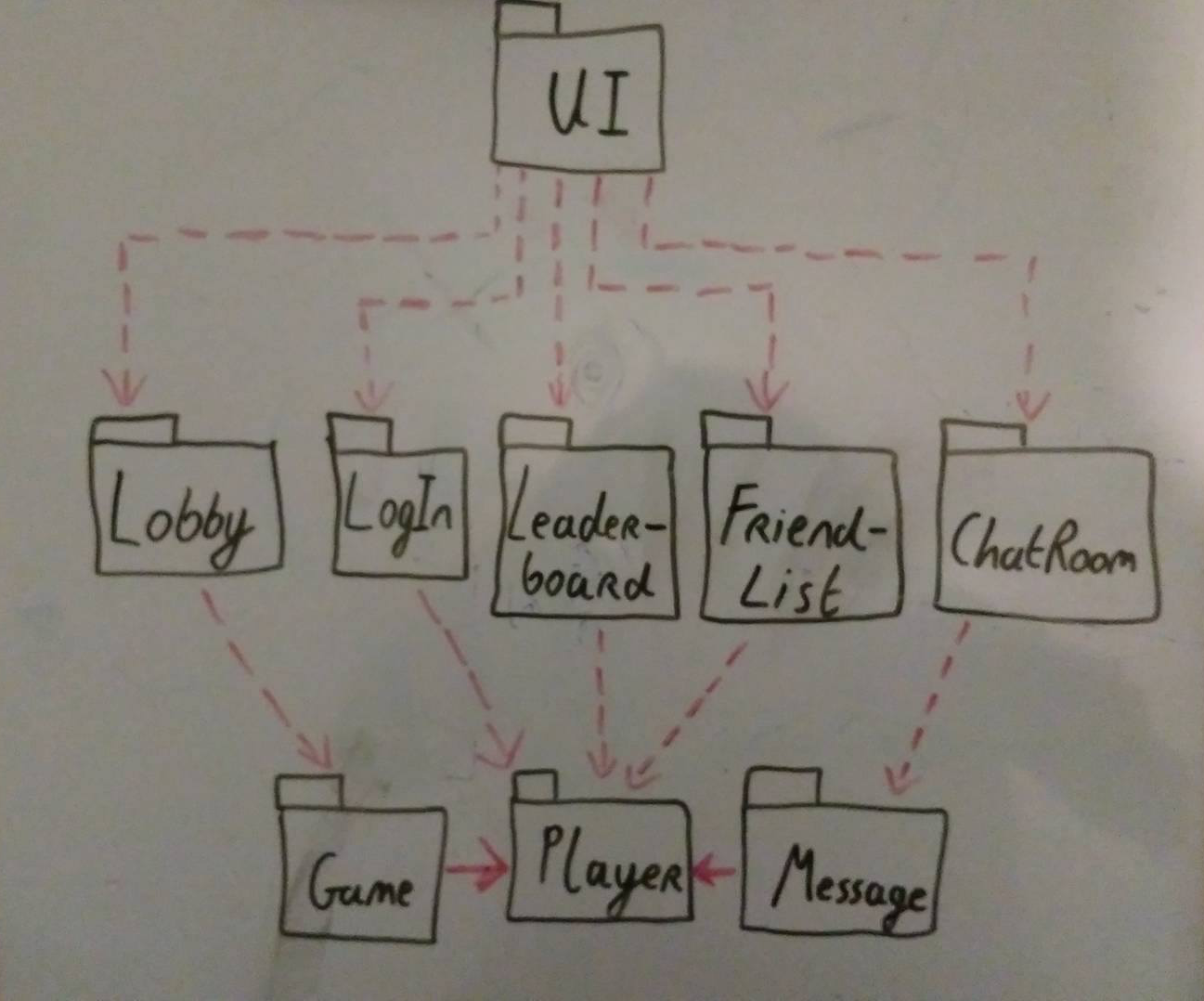
Our UML workbench consisted of Astah and Visual Paradigm. These basic workbenches were easy to use and offered ideal tools for software development and system design. Astah is ideal for sketch like diagrams as it has a very simple interface, while Visual Paradigm is suitable for more complex diagrams as it interfaces with NetBeans.

We used Astah for creating the sketches of class diagrams and use cases. It is simple in use and created clear and easily understandable diagrams. The Visual Paradigm workbench can be used for the architectural and design blueprints; the design time class diagram, architectural diagram and state chart.

# Implementation Language

Java was our choice of the implementation language as an object oriented language, because of how familiar each of our team members are with it. Java is platform independent, meaning that - when compiled it can run on any platform supporting Java without the need for another recompilation. The JVM also has a garbage collector which automatically manages the memory. It is useful with Java applications as it prevents memory leakage.

# Architecture Diagram



# Analysis Sketches

# Identify candidate classes

Using a data driven design method, we came up with a list of potential candidate classes by using the noun identification technique.

Initial list:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Username | Password | Registration | Logging | Game | System |
| Play | Invite | Message | Profile | User | Notification |
| Request | Send | Accept | Decline | User Interface | Display |
| Friend | Friendlist | Dialog | Input | Leaderboard | Lobbylist |
| Chatroom | Player | User | Text | Connection |  |

Heuristics:

1. Too vague or to specific – RED
2. An operation – BLUE
3. An attribute - Orange
4. Out of Scope - GREEN
5. Other similar objects – Purple

Filtered list:

|  |  |  |  |
| --- | --- | --- | --- |
| User | Player | Game | Leaderboard |
| Lobbylist | Friendlist | Friend | Chatroom |
| Message | Invite | User Interface | Log |

## User

A user encompasses all the data related to the logged in user who uses the application. Their data is updated after each game they complete.

## Player

A user who is matched into a game becomes a player, accepting their inputs to play the game.

## Game

The game is the logic for the game to be played.

## Leaderboard

The leaderboard encompasses all the data to be displayed on the leaderboard, letting data displayed be sorted by various sorting methods.

## Lobbylist

Lobbylist keeps track of all open connections, filtering out unavailable connections (already in game, etc.).

## Friendlist

Friendlist keeps track of all the added friends of the current user.

## Friend

Friend encompasses the data of a friend of the user, allowing certain elements to be edited such as a nickname.

## Chatroom

Chatroom handles messages that the user receives from other users, friends, the server etc. and allows the user to input messages to be sent to their desired destination.

## Message

A message encompasses the data relevant to a sent message; its contents, sender and destination.

## Invite

There are two types of invites:

1. Game invite: Allows a user to challenge a friend directly to play a game, and allows the recipient to choose to accept.
2. Friend invite: Allows a user to send a friend request to another user, and allows the recipient to choose to accept.

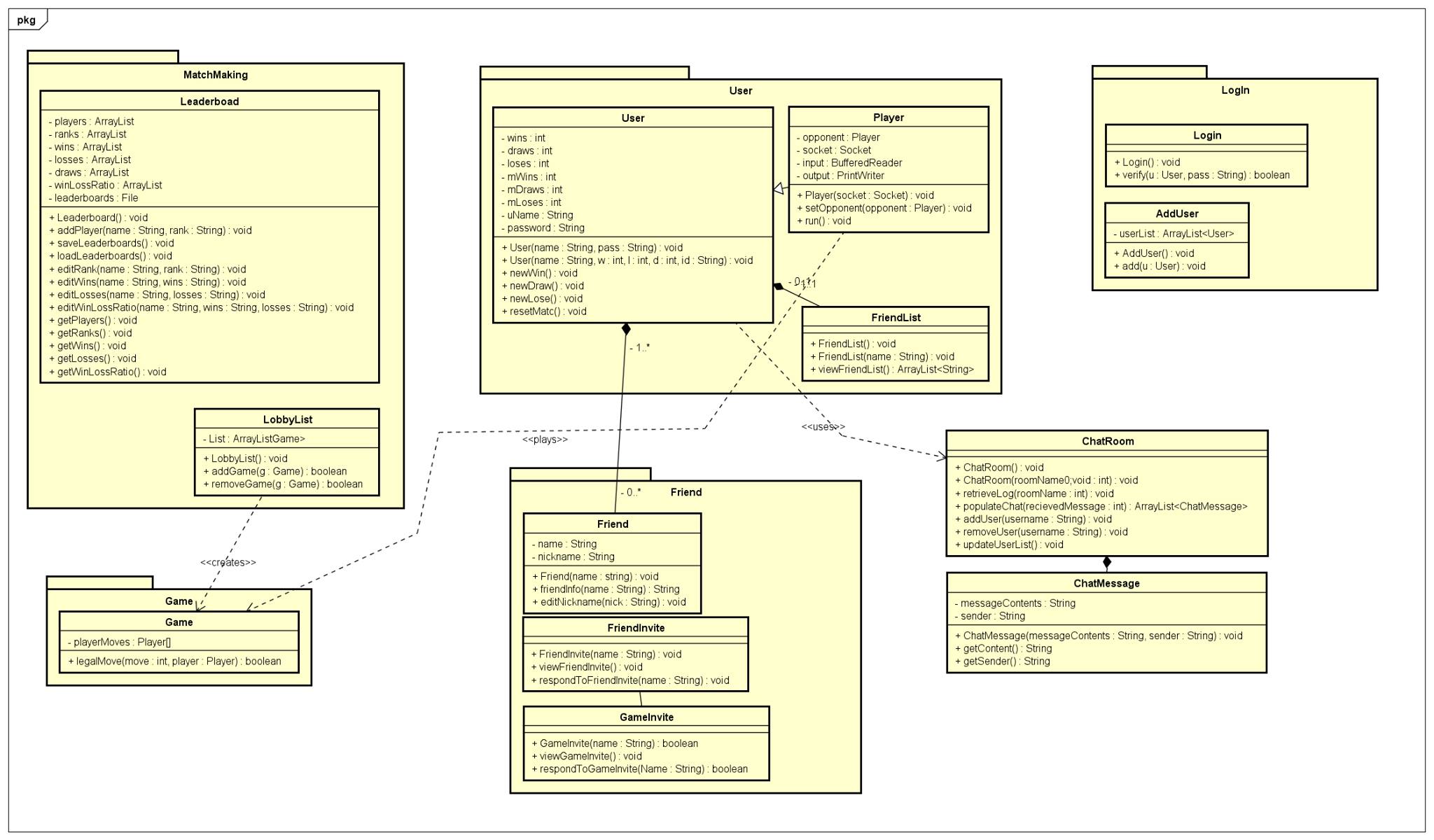
## UI

Handles all graphical elements that the user may interact with once they launch the application.

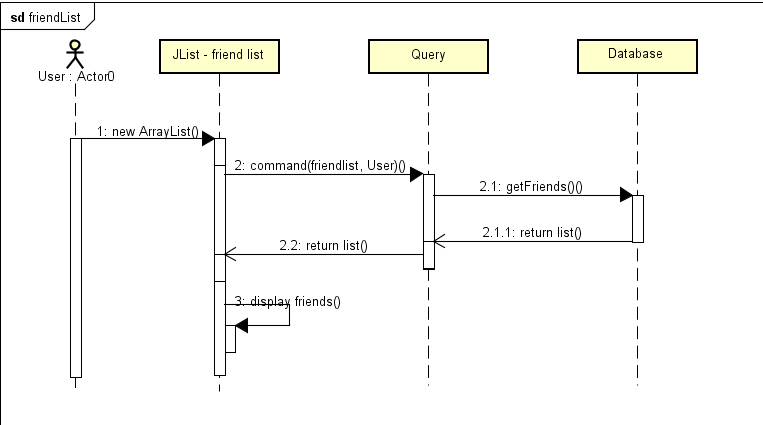
## Log

Log is for taking input information from the user. It encompasses users logging in, out and registering.

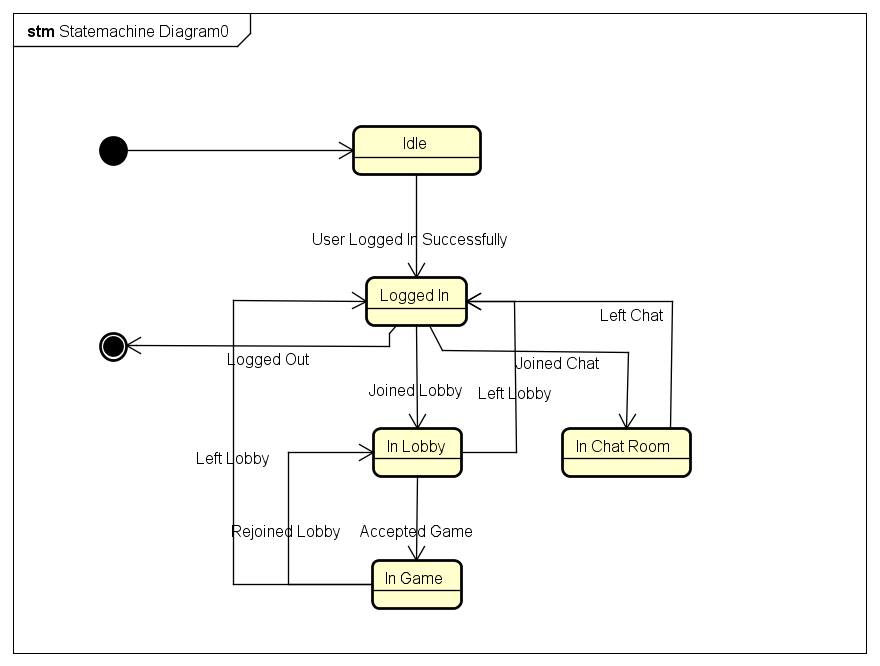
# Class diagrams



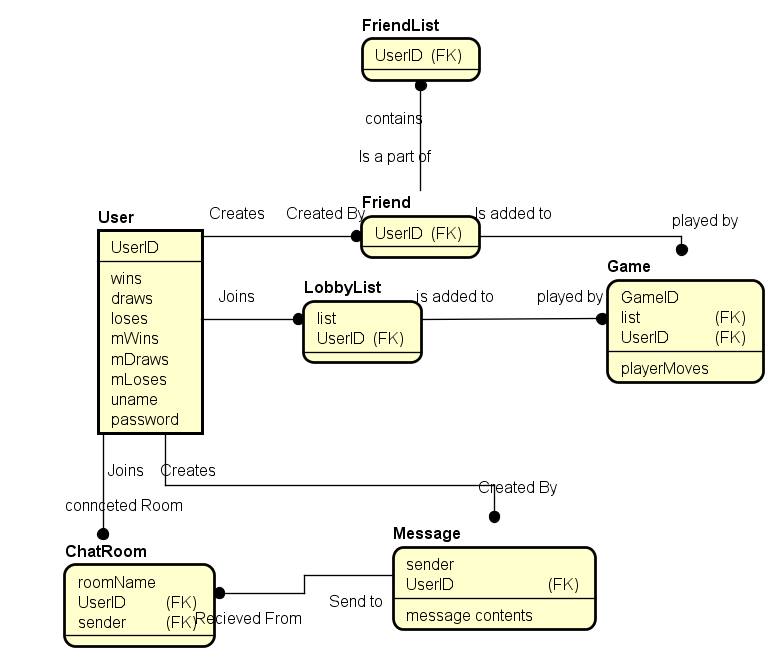
# Interaction diagram



# State chart



# Entity relationship diagrams



# Code

# Overview of Contribution to Code

|  |  |  |  |
| --- | --- | --- | --- |
| **Package** | **Class** | **LOC** | **Author** |
| **CS4125Server** |  |  | Dean |
| ChatServer | ChatRoom | 53 | Fergal |
| ChatServer | ChatServer | 83 | Fergal |
| FileServer | FileServer | 25 | Dean |
| FileServer | FriendSearch | 118 | Pawel |
| FileServer | LeaderboardSearch | 30 | Philip |
| FileServer | Query | 108 | Dean (33%) Philip (33%) Pawel (33%) |
| FileServer | Search | 10 | Philip |
| FileServer | UserAdd | 28 | Dean |
| FileServer | UserSearch | 60 | Dean (50%) Philip (50%) |
| GameServer | GamePlayer | 104 | Dean |
| GameServer | GameServer | 25 | Dean |
| GameServer | Match | 31 | Dean |
| GameServer | MatchMaker | 49 | Dean |
| GameServer | Result | 31 | Dean |
| GameServer | ResultWriter | 47 | Dean |
| Player | User | 69 | Philip |
| Server | ServerFactory | 13 | Dean |
| **CS4125User** |  |  | Dean |
| Chat | ChatRoomController | 34 | Fergal |
| Chat | ChatRoomModel | 89 | Fergal |
| Chat | Message | 18 | Fergal |
| Friend | Friend | 20 | Pawel |
| Friend | FriendInvite | 26 | Pawel |
| Friend | FriendListController | 60 | Pawel |
| Friend | FriendListModel | 83 | Pawel |
| Friend | GameInvite | 25 | Pawel |
| Game | Draw | 20 | Dean |
| Game | GameController | 52 | Dean |
| Game | GameModel | 101 | Dean |
| Game | GameReport | 8 | Dean |
| Game | Loss | 20 | Dean |
| Game | Win | 20 | Dean |
| Leaderboards | LeaderboardController | 55 | Philip |
| Leaderboards | LeaderboardModel | 134 | Philip |
| LogIn | LoginController | 46 | Dean |
| LogIn | LoginModel | 77 | Dean |
| MainMenu | MainMenuController | 53 | Dean |
| Player | User | 64 | Philip |
| UI | ChatRoomUI | 129 | Fergal |
| UI | ChatRoomUIFactory | 19 | Dean |
| UI | FriendListUI | 89 | Pawel |
| UI | FriendListUIFactory | 18 | Dean |
| UI | GameUI | 117 | Dean |
| UI | GameUIFactory | 19 | Dean |
| UI | LeaderboardUI | 177 | Philip |
| UI | LeaderboardUIFactory | 18 | Dean |
| UI | LogInUI | 94 | Dean |
| UI | LogInUIFactory | 11 | Dean |
| UI | MainMenuUI | 102 | Dean |
| UI | MainMenuUIFactory | 16 | Dean |
| UI | UIFactory | 9 | Dean |
| cs4125User | CS4125User | 16 | Dean |
|  | | **Total LOC** | **Author** |
| 1255 | Dean |
| 406 | Fergal |
| 605 | Philip |
| 457 | Pawel |
| 2723 | **Total** |

# Code Implementation

# ChatServer package - ChatRoom

package ChatServer**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

public class ChatRoom **extends** Thread **{**

BufferedReader input**;**

PrintWriter output**;**

Socket socket**;**

ChatServer cs**;**

String username **=** ""**;**

public ChatRoom**(**ChatServer s**,** Socket so**)** **throws** Exception **{**

socket **=** so**;**

cs **=** s**;**

input **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

output **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

**}**

public void run**()** **{**

output**.**println**(**"CONNECTED"**);**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

String command **=** input**.**readLine**();**

**if** **(**command **!=** **null)** **{**

System**.**out**.**println**(**command**);**

**if** **(**command**.**startsWith**(**"USER"**))** **{**

System**.**out**.**println**(**"Username received."**);**

username **=** command**.**substring**(**6**);**

cs**.**newUser**(**username**);**

**}** **else** **if** **(**command**.**startsWith**(**"QUIT"**))** **{**

cs**.**deleteUser**(**username**);**

interrupt**();**

**}** **else** **{**

System**.**out**.**println**(**"Message received."**);**

cs**.**sendMessage**(**username**,** command**);**

**}**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"User died: " **+** e**);**

interrupt**();**

**}**

**}**

System**.**out**.**println**(**"Query closed."**);**

**return;**

**}**

**}**

# ChatServer

package ChatServer**;**

**import** java**.**io**.\*;**

**import** java**.**net**.\*;**

**import** java**.**util**.\*;**

public class ChatServer **extends** Thread **{**

public ArrayList**<**Socket**>** ConnectedSockets **=** **new** ArrayList**<>();**

public ArrayList**<**String**>** ConnectedUsers **=** **new** ArrayList**<>();**

private ServerSocket server**;**

public ChatServer**()** **throws** IOException **{**

server **=** **new** ServerSocket**(**1234**);**

start**();**

**}**

@Override

public void run**()** **{**

System**.**out**.**println**(**"Chat Server Is Running"**);**

**while** **(true)** **{**

**try** **{**

Socket sock **=** server**.**accept**();**

ConnectedSockets**.**add**(**sock**);**

System**.**out**.**println**(**"New Chat User: " **+** sock**.**getLocalAddress**().**getHostName**());**

ChatRoom chat **=** **new** ChatRoom**(this,** sock**);**

chat**.**start**();**

**}** **catch** **(**Exception x**)** **{**

System**.**out**.**println**(**"Player died: " **+** x**);**

**}**

**}**

**}**

public void newUser**(**String username**)** **throws** IOException **{**

System**.**out**.**println**(**"Adding new user"**);**

ConnectedUsers**.**add**(**username**);**

String output **=** ""**;**

**for** **(**int c **=** 0**;** c **<** ConnectedUsers**.**size**();** c**++)** **{**

output **+=** ConnectedUsers**.**get**(**c**)** **+** " "**;**

**}**

**for** **(**int i **=** 0**;** i **<** ConnectedSockets**.**size**();** i**++)** **{**

Socket tempSock **=** **(**Socket**)** ConnectedSockets**.**get**(**i**);**

PrintWriter userListOut **=** **new** PrintWriter**(**tempSock**.**getOutputStream**());**

System**.**out**.**println**(**"Message to " **+** tempSock **+** " sent."**);**

userListOut**.**println**(**"USERS: " **+** output**);**

userListOut**.**println**(**"MESSAGE: SERVER~~" **+** username **+** " has joined."**);**

userListOut**.**flush**();**

**}**

**}**

public void deleteUser**(**String username**)** **throws** IOException **{**

System**.**out**.**println**(**"Adding new user"**);**

ConnectedUsers**.**add**(**username**);**

String output **=** ""**;**

**for** **(**int c **=** 0**;** c **<** ConnectedUsers**.**size**();** c**++)** **{**

**if** **(**ConnectedUsers**.**get**(**c**).**equals**(**username**))** **{**

ConnectedUsers**.**remove**(**c**);**

**}** **else** **{**

output **+=** ConnectedUsers**.**get**(**c**)** **+** " "**;**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** ConnectedSockets**.**size**();** i**++)** **{**

Socket tempSock **=** **(**Socket**)** ConnectedSockets**.**get**(**i**);**

PrintWriter userListOut **=** **new** PrintWriter**(**tempSock**.**getOutputStream**());**

userListOut**.**println**(**"USERS: " **+** output**);**

userListOut**.**println**(**"MESSAGE: SERVER~~" **+** username **+** " has left."**);**

userListOut**.**flush**();**

**}**

**}**

**}**

# FileServer package – FileServer

package FileServer**;**

**import** java**.**net**.**ServerSocket**;**

public class FileServer **extends** Thread **{**

ServerSocket listener**;**

public FileServer**()** **throws** Exception **{**

listener **=** **new** ServerSocket**(**5555**);**

start**();**

**}**

public void run**()** **{**

System**.**out**.**println**(**"File Server is Running"**);**

**while** **(true)** **{**

**try** **{**

Query q **=** **new** Query**(**listener**.**accept**());**

q**.**start**();**

**}** **catch** **(**Exception e**)** **{**

System**.**out**.**println**(**"Player died: " **+** e**);**

**}**

**}**

**}**

**}**

# FriendSearch

package FileServer**;**

**import** java**.**io**.**BufferedWriter**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**FileWriter**;**

**import** java**.**util**.**Scanner**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**ArrayList**;**

public class FriendSearch **extends** Search **{**

public FriendSearch**(**String user**)** **{**

username **=** user**;**

**}**

public String getFriends**()** **throws** IOException **{**

boolean found **=** **false;**

String u **=** "You have no friends :("**;;**

Scanner sc **=** **new** Scanner**(new** FileReader**(**"friends.txt"**));**

**while** **(**sc**.**hasNext**()** **&&** found **==** **false)** **{**

String temp **=** sc**.**nextLine**();**

**if** **(**temp**.**startsWith**(**username**))** **{**

u **=** temp**;**

found **=** **true;**

**}**

**}**

sc**.**close**();**

**return** u**;**

**}**

public boolean addFriend**(**String friend**)** **throws** IOException **{**

ArrayList**<**String**>** users **=** **new** ArrayList**<**String**>();**

Scanner sc **=** **new** Scanner**(new** FileReader**(**"friends.txt"**));**

boolean found1 **=** **false,** found2 **=** **false;**

int index1 **=** 0**,** index2 **=** 0**;**

String line1 **=** ""**,** line2 **=** ""**;**

**while** **(**sc**.**hasNext**())** **{**

String temp **=** sc**.**nextLine**();**

users**.**add**(**temp**);**

**}**

System**.**out**.**println**(**users**.**size**());**

**for** **(**int c **=** 0**;** **(**c **<** users**.**size**());** c**++)** **{**

String lineTemp **=** users**.**get**(**c**);**

String**[]** temp **=** lineTemp**.**split**(**", "**);**

System**.**out**.**println**(**friend **+** "\n" **+** temp**[**0**]);**

**if** **(**username**.**toLowerCase**().**equals**(**temp**[**0**].**toLowerCase**()))** **{**

line1 **=** lineTemp **+** ", " **+** friend **+** ", ()"**;**

index1 **=** c**;**

found1 **=** **true;**

System**.**out**.**println**(**"found1 = true"**);**

# 

**}** **else** **if** **(**friend**.**toLowerCase**().**equals**(**temp**[**0**].**toLowerCase**()))** **{**

line2 **=** lineTemp **+** ", " **+** username **+** ", ()"**;**

index2 **=** c**;**

found2 **=** **true;**

System**.**out**.**println**(**"found2 = true"**);**

**}**

**}**

**if** **(**found1 **==** **true** **&&** found2 **==** **true)** **{**

users**.**set**(**index1**,** line1**);**

users**.**set**(**index2**,** line2**);**

//Rewrite the data back to the file

BufferedWriter bw **=** **new** BufferedWriter**(new** FileWriter**(**"friends.txt"**));**

int counter **=** 0**;**

**while** **(**counter **<** users**.**size**())** **{**

bw**.**write**(**users**.**get**(**counter**));**

bw**.**newLine**();**

counter**++;**

**}**

bw**.**close**();**

**return** **true;**

**}** **else** **{**

**return** **false;**

**}**

**}**

public void editNick**(**String selected**,** String nick**)** **throws** IOException **{**

ArrayList**<**String**>** users **=** **new** ArrayList**<**String**>();**

Scanner sc **=** **new** Scanner**(new** FileReader**(**"friends.txt"**));**

boolean found **=** **false;**

**while** **(**sc**.**hasNext**())** **{**

String temp **=** sc**.**nextLine**();**

users**.**add**(**temp**);**

**}**

//Edit the user's records

**for** **(**int c **=** 0**;** c **<** users**.**size**()** **&&** found **==** **false;** c**++)** **{**

String**[]** temp **=** **(**users**.**get**(**c**)).**split**(**", "**);**

**if** **(**username**.**toLowerCase**().**equals**(**temp**[**0**].**toLowerCase**()))** **{**

boolean found2 **=** **false;**

**for** **(**int co **=** 1**;** co **<** temp**.**length **&&** found2 **==** **false;** co **+=** 2**)** **{**

**if** **(**selected**.**equals**(**temp**[**co**]))** **{**

found2 **=** **true;**

String temp2 **=** username **+** ", "**;**

temp**[**co **+** 1**]** **=** nick**;**

**for** **(**int c2 **=** 1**;** c2 **<** temp**.**length**;** c2**++)** **{**

temp2 **+=** temp**[**c2**]** **+** ", "**;**

**}**

users**.**set**(**c**,** temp2**);**

**}**

**}**

found **=** **true;**

**}**

**}**

//Rewrite the data back to the file

BufferedWriter bw **=** **new** BufferedWriter**(new** FileWriter**(**"friends.txt"**));**

**for** **(**int counter **=** 0**;** counter **<** users**.**size**();** counter**++)** **{**

bw**.**write**(**users**.**get**(**counter**));**

bw**.**newLine**();**

**}**

bw**.**close**();**

**}**

**}**

# LeaderboardSearch

package FileServer**;**

**import** Player**.**User**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Scanner**;**

public class LeaderboardSearch **extends** Search**{**

public LeaderboardSearch**(){**

**}**

public ArrayList**<**User**>** getLeaderboardItems**()** **throws** IOException **{**

ArrayList**<**User**>** users **=** **new** ArrayList**<>();**

**try** **(**Scanner sc **=** **new** Scanner**(new** FileReader**(**"Usernames.txt"**)))** **{**

User uFromFile**;**

**while** **(**sc**.**hasNext**())** **{**

String uArray**[]** **=** sc**.**nextLine**().**split**(**","**);**

username **=** uArray**[**0**];**

wins **=** Integer**.**parseInt**(**uArray**[**2**]);**

losses **=** Integer**.**parseInt**(**uArray**[**4**]);**

uFromFile **=** **new** User**(**username**,** wins**,** losses**);**

users**.**add**(**uFromFile**);**

**}**

**}**

**return** users**;**

**}**

**}**

# Query

package FileServer**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** Player**.\*;**

**import** java**.**util**.**ArrayList**;**

public class Query **extends** Thread **{**

BufferedReader input**;**

PrintWriter output**;**

Socket socket**;**

public Query**(**Socket so**)** **throws** Exception **{**

socket **=** so**;**

input **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

output **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

**}**

public void run**()** **{**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

String command **=** input**.**readLine**();**

**if** **(**command **!=** **null)** **{**

**if** **(**command**.**startsWith**(**"LOGIN"**))** **{**

String**[]** cA **=** command**.**substring**(**6**).**split**(**","**);**

UserSearch us **=** **new** UserSearch**(**cA**[**0**],** cA**[**1**]);**

User tempUser **=** us**.**search**();**

**if** **(**tempUser **==** **null)** **{**

output**.**println**(**"FAILIURE The username and password do not match. Please try again"**);**

**}** **else** **{**

output**.**println**(**"SUCCESS " **+** tempUser**.**getW**()** **+** "," **+** tempUser**.**getD**()** **+** "," **+** tempUser**.**getL**());**

socket**.**close**();**

interrupt**();**

**}**

**}** **else** **if** **(**command**.**startsWith**(**"REGISTER"**))** **{**

String**[]** cA **=** command**.**substring**(**9**).**split**(**","**);**

UserSearch us **=** **new** UserSearch**(**cA**[**0**],** cA**[**1**]);**

boolean check **=** us**.**exists**();**

**if** **(**check **==** **false)** **{**

UserAdd ua **=** **new** UserAdd**(new** User**(**cA**[**0**],** cA**[**1**]));**

ua**.**add**();**

output**.**println**(**"REGISTERED"**);**

socket**.**close**();**

interrupt**();**

**}** **else** **{**

output**.**println**(**"FAILIURE The username already exists. Try again with a different username."**);**

**}**

**}** **else** **if** **(**command**.**startsWith**(**"LEADERBOARDDATA"**))** **{**

LeaderboardSearch board **=** **new** LeaderboardSearch**();**

ArrayList**<**User**>** temp **=** board**.**getLeaderboardItems**();**

String data **=** ""**;**

**for** **(**int i **=** 0**;** i **<** temp**.**size**();** i**++)** **{**

data **+=** temp**.**get**(**i**).**getUsername**()** **+** " " **+** temp**.**get**(**i**).**getW**()** **+** " " **+** temp**.**get**(**i**).**getL**()** **+** ","**;**

**}**

output**.**println**(**data**);**

socket**.**close**();**

interrupt**();**

**}** **else** **if** **(**command**.**startsWith**(**"friendList"**))** **{**

String**[]** cA **=** command**.**split**(**", "**);**

FriendSearch s **=** **new** FriendSearch**(**cA**[**1**]);**

String list **=** s**.**getFriends**();**

output**.**println**(**list**);**

**}** **else** **if** **(**command**.**startsWith**(**"editNickname"**))** **{**

String**[]** cA **=** command**.**split**(**", "**);** //editNickname, currentPlayer.getUsername(), selected, nick

FriendSearch s **=** **new** FriendSearch**(**cA**[**1**]);**

s**.**editNick**(**cA**[**2**],** cA**[**3**]);**

String list **=** s**.**getFriends**();**

output**.**println**(**list**);**

**}** **else** **if** **(**command**.**startsWith**(**"inviteToGame"**))** **{**

String**[]** cA **=** command**.**split**(**", "**);** //inviteToGame, currentPlayer.getUsername(), selected

**}** **else** **if** **(**command**.**startsWith**(**"addFriend"**))** **{**

String**[]** cA **=** command**.**split**(**", "**);** // addFriend, currentPlayer.getUsername(), addedFriend

System**.**out**.**println**(**"Adding friend"**);**

FriendSearch s **=** **new** FriendSearch**(**cA**[**1**]);**

boolean added **=** s**.**addFriend**(**cA**[**2**]);**

**if** **(**added **==** **true)** **{**

output**.**println**(**"SUCCESS"**);**

String list **=** s**.**getFriends**();**

output**.**println**(**list**);**

**}** **else** **{**

output**.**println**(**"FAILIURE"**);**

**}**

**}** **else** **if** **(**command**.**startsWith**(**"CLOSE"**))** **{**

socket**.**close**();**

interrupt**();**

**}**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"User died: " **+** e**);**

**try** **{**

socket**.**close**();**

**}** **catch** **(**IOException ex**)** **{**

**}**

interrupt**();**

**}**

**}**

System**.**out**.**println**(**"Query closed."**);**

**}**

**}**

# Search

package FileServer**;**

public class Search **{**

protected String username **=** ""**;**

protected String password **=** ""**;**

protected int wins **=**0**;**

protected int draws **=** 0**;**

protected int losses **=** 0**;**

**}**

# UserAdd

package FileServer**;**

**import** java**.**io**.**BufferedWriter**;**

**import** java**.**io**.**FileWriter**;**

**import** java**.**io**.**IOException**;**

**import** Player**.\*;**

public class UserAdd **{**

User user**;**

public UserAdd**(**User u**)**

**{**

user **=** u**;**

**}**

public void add**()** **throws** IOException

**{**

BufferedWriter x **=** **new** BufferedWriter**(new** FileWriter**(**"Usernames.txt"**,** **true));**

x**.**write**(**user**.**getUsername**()** **+** "," **+** user**.**getPassword**()** **+** ",0,0,0"**);**

x**.**newLine**();**

x**.**close**();**

BufferedWriter y **=** **new** BufferedWriter**(new** FileWriter**(**"friends.txt"**,** **true));**

y**.**write**(**user**.**getUsername**());**

y**.**newLine**();**

y**.**close**();**

**}**

**}**

# UserSearch

package FileServer**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**util**.**Scanner**;**

**import** Player**.\*;**

**import** java**.**io**.**IOException**;**

public class UserSearch **extends** Search**{**

public UserSearch**(**String username**,** String password**)** **{**

**this.**username **=** username**;**

**this.**password **=** password**;**

**}**

public User search**()** **throws** IOException

**{**

boolean found **=** **false;**

Scanner sc **=** **new** Scanner**(new** FileReader**(**"Usernames.txt"**));**

**while** **(**sc**.**hasNext**()** **&&** found **==** **false)** **{**

String u **=** sc**.**nextLine**();**

String uA**[]** **=** u**.**split**(**","**);**

String uTemp **=** uA**[**0**].**toLowerCase**();**

String pTemp **=** uA**[**1**].**toLowerCase**();**

**if** **(**username**.**toLowerCase**().**equals**(**uTemp**)&&(**password**.**toLowerCase**().**equals**(**pTemp**)))**

**{**

wins **=** Integer**.**parseInt**(**uA**[**2**]);**

draws **=** Integer**.**parseInt**(**uA**[**3**]);**

losses **=** Integer**.**parseInt**(**uA**[**4**]);**

found **=** **true;**

**}**

**}**

sc**.**close**();**

**if** **(**found **==** **false)**

**return** **null;**

**else**

**{**

User uFromFile **=** **new** User**(**username**,** password**,** wins**,** draws**,** losses**);**

**return** uFromFile**;**

**}**

**}**

public boolean exists**()** **throws** IOException

**{**

boolean found **=** **false;**

Scanner sc **=** **new** Scanner**(new** FileReader**(**"Usernames.txt"**));**

**while** **(**sc**.**hasNext**()** **&&** found **==** **false)** **{**

String u **=** sc**.**nextLine**();**

String uA**[]** **=** u**.**split**(**","**);**

String uTemp **=** uA**[**0**].**toLowerCase**();**

**if** **(**username**.**toLowerCase**().**equals**(**uTemp**))**

found **=** **true;**

**}**

sc**.**close**();**

**if** **(**found **==** **false)**

**return** **false;**

**else**

**return** **true;**

**}**

**}**

# GameServer Package – GamePlayer

package GameServer**;**

**import** Player**.**User**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

public class GamePlayer **extends** Thread **{**

BufferedReader input**;**

PrintWriter output**;**

Socket socket**;**

String username**;**

int score**;**

int wins**,** draws**,** losses**;**

String move **=** ""**;**

int status **=** 0**;**

public GamePlayer**(**Socket socket**)** **throws** Exception **{**

**this.**socket **=** socket**;**

**try** **{**

input **=** **new** BufferedReader**(**

**new** InputStreamReader**(**socket**.**getInputStream**()));**

output **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"Player died: " **+** e**);**

**}**

**}**

public int getScore**()** **{**

**return** score**;**

**}**

public String getUsername**()** **{**

**return** username**;**

**}**

public String getMove**()** **{**

**return** move**;**

**}**

public void sendResult**(**Result result**)** **{**

output**.**println**(**"RESULT " **+** result**.**getResults**());**

move **=** ""**;**

**if** **(**result**.**getWinner**().**equals**(**username**))** **{**

wins**++;**

**}** **else** **if** **(**result**.**getWinner**().**equals**(**"draw"**))** **{**

draws**++;**

**}** **else** **{**

losses**++;**

**}**

**}**

# 

public int getStatus**()** **{**

**return** status**;**

**}**

public void inGame**(**String oppUsername**)** **{**

status **=** 1**;**

output**.**println**(**"MAKEMOVE " **+** oppUsername**);**

**}**

public void run**()** **{**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

String command **=** input**.**readLine**();**

**if** **(**command**.**startsWith**(**"DETAILS"**))** **{**

String**[]** details **=** command**.**substring**(**8**).**split**(**","**);**

username **=** details**[**0**];**

wins **=** Integer**.**parseInt**(**details**[**1**]);**

draws **=** Integer**.**parseInt**(**details**[**2**]);**

losses **=** Integer**.**parseInt**(**details**[**3**]);**

score **=** 1000 **+** **(**25 **\*** wins**)** **-** **(**25 **\*** losses**);**

output**.**println**(**"MESSAGE Finding another user..."**);**

**}** **else** **if** **(**command**.**startsWith**(**"QUIT"**))** **{**

ResultWriter rw **=** **new** ResultWriter**(new** User**(**username**,** wins**,** draws**,** losses**));**

rw**.**write**();**

socket**.**close**();**

status **=** **-**1**;**

interrupt**();**

**}** **else** **if** **(**command**.**startsWith**(**"MOVE"**))** **{**

move **=** command**.**substring**(**5**);**

System**.**out**.**println**(**command**.**substring**(**5**));**

**}** **else** **if** **(**command**.**startsWith**(**"FORFEIT"**))** **{**

losses**++;**

ResultWriter rw **=** **new** ResultWriter**(new** User**(**username**,** wins**,** draws**,** losses**));**

rw**.**write**();**

socket**.**close**();**

status **=** **-**1**;**

interrupt**();**

**}** **else** **if** **(**command**.**startsWith**(**"NEWGAME"**))** **{**

status **=** 0**;**

System**.**out**.**println**(**username **+** " is back in matchmaking."**);**

**}**

**}** **catch** **(**IOException e**)** **{**

**}**

**}**

**return;**

**}**

**}**

# GameServer

package GameServer**;**

**import** java**.**net**.**ServerSocket**;**

public class GameServer **extends** Thread **{**

ServerSocket listener**;**

public GameServer**()** **throws** Exception **{**

listener **=** **new** ServerSocket**(**4444**);**

start**();**

**}**

public void run**()**

**{**

System**.**out**.**println**(**"Rock Paper Scissors Server is Running"**);**

MatchMaker lobby **=** **new** MatchMaker**();**

**while** **(true)** **try** **{**

GamePlayer lplayer **=** **new** GamePlayer**(**listener**.**accept**());**

lplayer**.**start**();**

System**.**out**.**println**(**"Player joined queue."**);**

lobby**.**addPlayer**(**lplayer**);**

**}**

**catch(**Exception e**)**

**{**

System**.**out**.**println**(**"Player died: " **+** e**);**

**}**

**}**

**}**

# Match

package GameServer**;**

public class Match **extends** Thread**{**

GamePlayer player1**,** player2**;**

public Match**(**GamePlayer gp1**,** GamePlayer gp2**)**

**{**

System**.**out**.**println**(**"Match started"**);**

player1 **=** gp1**;**

player2 **=** gp2**;**

player1**.**inGame**(**player2**.**getUsername**());**

player2**.**inGame**(**player1**.**getUsername**());**

System**.**out**.**println**(**"Game begin."**);**

start**();**

**}**

public void run**()** **{**

**while** **(!**Thread**.**interrupted**())**

**{**

System**.**out**.**print**(**""**);**

**if(!(**player1**.**getMove**().**equals**(**""**))&&!(**player2**.**getMove**().**equals**(**""**)))**

**{**

Result result **=** **new** Result**(**player1**.**getUsername**(),** player1**.**getMove**(),**player2**.**getUsername**(),** player2**.**getMove**());**

player1**.**sendResult**(**result**);**

player2**.**sendResult**(**result**);**

interrupt**();**

System**.**out**.**println**(**"Game over."**);**

**}**

**}**

**return;**

**}**

**}**

# MatchMaker

package GameServer**;**

**import** java**.**util**.**ArrayList**;**

public class MatchMaker **extends** Thread **{**

ArrayList**<**GamePlayer**>** lobbyA**;**

public MatchMaker**()** **{**

lobbyA **=** **new** ArrayList**<**GamePlayer**>();**

start**();**

**}**

public void addPlayer**(**GamePlayer gp**)** **{**

lobbyA**.**add**(**gp**);**

**}**

public void run**()** **{**

**while** **(true)** **{**

System**.**out**.**print**(**""**);**

**for** **(**int i **=** 0**;** i **<** lobbyA**.**size**()** **-** 1**;** i**++)** **{**

**if** **(**lobbyA**.**get**(**i**).**getStatus**()** **==** 0**)** **{**

int lowest **=** 10000**,** lowestJ **=** **-**1**;**

**for** **(**int j **=** i **+** 1**;** j **<** lobbyA**.**size**();** j**++)** **{**

**if** **(**lobbyA**.**get**(**j**).**getStatus**()** **==** **-**1**)** **{**

lobbyA**.**remove**(**j**);**

**}** **else** **if** **(**lobbyA**.**get**(**j**).**getStatus**()** **==** 0**)** **{**

int temp **=** Math**.**abs**(**lobbyA**.**get**(**i**).**getScore**()** **-** lobbyA**.**get**(**j**).**getScore**());**

System**.**out**.**println**(**temp**);**

**if** **(**temp **<** lowest**)** **{**

lowestJ **=** j**;**

lowest **=** temp**;**

**}**

**}**

**}**

**if** **(**lowestJ **!=** **-**1**)** **{**

System**.**out**.**println**(**"Match made"**);**

Match m **=** **new** Match**(**lobbyA**.**get**(**i**),** lobbyA**.**get**(**lowestJ**));**

**}**

**}** **else** **if** **(**lobbyA**.**get**(**i**).**getStatus**()** **==** **-**1**)** **{**

lobbyA**.**remove**(**i**);**

**}**

**}**

**}**

**}**

**}**

# Result

package GameServer**;**

public class Result **{**

String player1**,** player2**,** move1**,** move2**,** winner**;**

public Result**(**String p1**,** String m1**,** String p2**,** String m2**)**

**{**

player1 **=** p1**;**

player2 **=** p2**;**

move1 **=** m1**;**

move2 **=** m2**;**

**if(**move1**.**equals**(**move2**))**

winner **=** "draw"**;**

**else** **if((**move2**.**equals**(**"rock"**)&&**move1**.**equals**(**"paper"**))**

**||(**move2**.**equals**(**"paper"**)&&**move1**.**equals**(**"scissors"**))**

**||(**move2**.**equals**(**"scissors"**)&&**move1**.**equals**(**"rock"**)))**

winner **=** player1**;**

**else** **if((**move1**.**equals**(**"rock"**)&&**move2**.**equals**(**"paper"**))**

**||(**move1**.**equals**(**"paper"**)&&**move2**.**equals**(**"scissors"**))**

**||(**move1**.**equals**(**"scissors"**)&&**move2**.**equals**(**"rock"**)))**

winner **=** player2**;**

**}**

public String getResults**()**

**{**

**return** player1 **+** "," **+** move1 **+** "," **+** player2 **+** "," **+** move2 **+** "," **+** winner**;**

**}**

public String getWinner**()**

**{**

**return** winner**;**

**}**

**}**

# ResultWriter

package GameServer**;**

**import** java**.**io**.**BufferedWriter**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**FileWriter**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Scanner**;**

**import** Player**.\*;**

**import** java**.**io**.**IOException**;**

public class ResultWriter **{**

User user**;**

public ResultWriter**(**User u**)** **{**

user **=** u**;**

**}**

public void write**()** **throws** IOException **{**

ArrayList**<**String**>** users **=** **new** ArrayList**<**String**>();**

Scanner sc **=** **new** Scanner**(new** FileReader**(**"Usernames.txt"**));**

boolean found **=** **false;**

**while** **(**sc**.**hasNext**())** **{**

String temp **=** sc**.**nextLine**();**

users**.**add**(**temp**);**

**}**

//Edit the user's records

**for** **(**int c **=** 0**;** c **<** users**.**size**()** **&&** found **==** **false;** c**++)** **{**

String**[]** temp **=** **(**users**.**get**(**c**)).**split**(**","**);**

**if** **(**user**.**getUsername**().**toLowerCase**().**equals**(**temp**[**0**].**toLowerCase**()))** **{**

user**.**setPassword**(**temp**[**1**]);**

users**.**set**(**c**,** user**.**getUsername**()** **+** "," **+** user**.**getPassword**()** **+** "," **+** user**.**getW**()** **+** "," **+** user**.**getD**()** **+** "," **+** user**.**getL**());**

found **=** **true;**

**}**

**}**

//Rewrite the data back to the file

BufferedWriter bw **=** **new** BufferedWriter**(new** FileWriter**(**"Usernames.txt"**));**

int counter **=** 0**;**

**while** **(**counter **<** users**.**size**())** **{**

bw**.**write**(**users**.**get**(**counter**));**

bw**.**newLine**();**

counter**++;**

**}**

bw**.**close**();**

**}**

**}**

# Player Package – User

package Player**;**

public class User **{**

int wins **=** 0**,** draws **=** 0**,** losses **=** 0**;**

String username **=** ""**,** password **=** ""**;**

public User**(**String name**,** String pass**)** **{**

username **=** name**;**

password **=** pass**;**

**}**

public User**(**String name**,** String pass**,** int w**,** int d**,** int l**)** **{**

username **=** name**;**

password **=** pass**;**

wins **=** w**;**

draws **=** d**;**

losses **=** l**;**

**}**

public User**(**String name**,** int w**,** int d**,** int l**)** **{**

username **=** name**;**

wins **=** w**;**

draws **=** d**;**

losses **=** l**;**

**}**

public User**(**String name**,** int w**,** int l**)** **{**

username **=** name**;**

wins **=** w**;**

losses **=** l**;**

**}**

public String getUsername**()** **{**

**return** username**;**

**}**

public String getPassword**()** **{**

**return** password**;**

**}**

public void setPassword**(**String pass**)** **{**

password **=** pass**;**

**}**

public int getW**()** **{**

**return** wins**;**

**}**

public int getD**()** **{**

**return** draws**;**

**}**

public int getL**()** **{**

**return** losses**;**

**}**

public void newW**()** **{**

wins**++;**

**}**

public void newD**()** **{**

draws**++;**

**}**

public void newL**()** **{**

losses**++;**

**}**

**}**

# Server Package – ServerFactory

package Server**;**

**import** GameServer**.**GameServer**;**

**import** FileServer**.**FileServer**;**

**import** ChatServer**.**ChatServer**;**

public class ServerFactory **{**

public static void main**(**String**[]** args**)** **throws** Exception **{**

FileServer fs **=** **new** FileServer**();**

GameServer gs **=** **new** GameServer**();**

ChatServer cs **=** **new** ChatServer**();**

**}**

**}**

# Chat Package – ChatRoomController

package Chat**;**

**import** UI**.**ChatRoomUI**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**io**.**IOException**;**

public class ChatRoomController **implements** ActionListener**{**

private ChatRoomUI crui**;**

private ChatRoomModel crm**;**

public ChatRoomController**(**ChatRoomUI ui**)** **throws** IOException**{**

crui **=** ui**;**

crm **=** **new** ChatRoomModel**(**crui**);**

**}**

public void sendMessage**(**String message**){**

**}**

@Override

public void actionPerformed**(**ActionEvent event**)** **{**

Object source **=** event**.**getSource**();**

**if(**source **==** crui**.**jbSend**){**

**if(!**crui**.**getMessage**().**equals**(**""**)){**

String message **=** crui**.**getMessage**();**

crm**.**sendMessage**(**message**);**

crui**.**clearTF**();**

**}**

**}**

**}**

public void quit**()**

**{**

crm**.**quit**();**

**}**

**}**

# ChatRoomModel

package Chat**;**

**import** Player**.**User**;**

**import** UI**.**ChatRoomUI**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

**import** java**.**util**.**ArrayList**;**

**import** javax**.**swing**.**JOptionPane**;**

public class ChatRoomModel **extends** Thread **{**

private final int port **=** 1234**;**

private ChatRoomUI crui**;**

private Socket socket**;**

private PrintWriter out**;**

private BufferedReader in**;**

private User currentUser**;**

private static ArrayList**<**String**>** ConnectedUsers **=** **new** ArrayList**<**String**>();**

public ChatRoomModel**(**ChatRoomUI ui**)** **throws** IOException **{**

**try** **{**

socket **=** **new** Socket**(**"localhost"**,** port**);**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

System**.**out**.**println**(**"Connected to chat server."**);**

crui **=** ui**;**

currentUser **=** crui**.**getCurrentUser**();**

**}** **catch** **(**UnknownHostException e**)** **{**

System**.**out**.**println**(**"Unknown host: localhost.eng"**);**

System**.**exit**(**1**);**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"\nNo I/O\n" **+** e**);**

System**.**exit**(**1**);**

**}**

start**();**

**}**

@Override

public void run**()** **{**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

String recievedMessage **=** in**.**readLine**();**

**if** **(**recievedMessage**.**startsWith**(**"CONNECTED"**))** **{**

System**.**out**.**println**(**"Connected"**);**

out**.**println**(**"USER: " **+** currentUser**.**getUsername**());**

System**.**out**.**println**(**"Username sent"**);**

**}** **else** **if** **(**recievedMessage**.**startsWith**(**"USERS: "**))** **{**

System**.**out**.**println**(**recievedMessage**);**

recievedMessage **=** recievedMessage**.**substring**(**7**);**

recievedMessage **=** recievedMessage**.**replace**(**"["**,** ""**);**

recievedMessage **=** recievedMessage**.**replace**(**"]"**,** ""**);**

System**.**out**.**println**(**recievedMessage**);**

String**[]** usernames **=** recievedMessage**.**split**(**" "**);**

crui**.**updateUsers**(**usernames**);**

**}** **else** **if** **(**recievedMessage**.**startsWith**(**"MESSAGE: "**))** **{**

String temp **=** recievedMessage**.**substring**(**9**);**

String**[]** temp2 **=** temp**.**split**(**"~~"**);**

Message msg **=** **new** Message**(**temp2**[**0**],** temp2**[**1**]);**

crui**.**recieveMessage**(**msg**);**

**}**

**}** **catch** **(**IOException exc**)** **{**

JOptionPane**.**showMessageDialog**(null,** "I/O Error\n" **+** exc**,** "Error"**,** 0**);**

System**.**exit**(**1**);**

**}**

**}**

**return;**

**}**

public void sendMessage**(**String message**)** **{**

out**.**println**(**message**);**

**}**

public void updateUsers**(**String**[]** usernames**)** **{**

crui**.**jliOnline**.**setListData**(**usernames**);**

**}**

public void quit**()**

**{**

out**.**println**(**"QUIT"**);**

interrupt**();**

**}**

**}**

# Message

package Chat**;**

public class Message **{**

private String username**,** message**;**

public Message**(**String u**,** String m**)**

**{**

username **=** u**;**

message **=** m**;**

**}**

public String**[]** get**()**

**{**

String**[]** msg **=** **{**username**,** message**};**

**return** msg**;**

**}**

**}**

# Friend Package – Friend

package Friend**;**

public class Friend **{**

private String name**;**

private String nickname**;**

public Friend**(**String name**){**

**this.**name **=** name**;**

**this.**nickname **=** ""**;**

**}**

public String friendInfo**(**String name**){**

String info **=** **this.**name **+** " (" **+** **this.**nickname **+** ")"**;**

**return** info**;**

**}**

public void editNickname**(**String nick**)** **{**

**this.**nickname **=** "nick"**;**

**}**

**}**

# FriendInvite

package Friend**;**

public class FriendInvite **{**

public boolean FriendInvite**(**String name**){**

// send invite to 'name'

// if accepted add friend to your friendlist

// return true;

// if declined don't add friend

**return** **false;**

**}**

public void viewFriendInvite**(){**

// view window with accept/decline friend invite.

**}**

public boolean respondToFriendInvite**(**String name**){**

// if accepted invitation from 'name', add 'name' to friendlist

// return true;

// if declined invitation from 'name', don't add 'name'

**return** **false;**

**}**

# FriendListController

package Friend**;**

**import** UI**.**FriendListUI**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** javax**.**swing**.**JOptionPane**;**

public class FriendListController **implements** ActionListener **{**

private FriendListUI flui**;**

private FriendListModel flm**;**

public FriendListController**(**FriendListUI ui**)** **{**

flui **=** ui**;**

flm **=** **new** FriendListModel**(**flui**);**

**}**

@Override

public void actionPerformed**(**ActionEvent event**)** **{**

Object source **=** event**.**getSource**();**

**if** **(**source **==** flui**.**btnEditNickname**)** **{**

**if** **(**flui**.**list**.**getSelectedValue**()** **!=** **null)** **{**

String**[]** temp **=** flui**.**list**.**getSelectedValue**().**split**(**" - "**);**

String selected **=** temp**[**0**];**

String nick **=** JOptionPane**.**showInputDialog**(null,** "Please enter a nickname for " **+** selected **+** "."**);**

flm**.**sendMessage**(**"editNickname, " **+** flui**.**currentPlayer**.**getUsername**()** **+** ", " **+** selected **+** ", " **+** nick**);**

**}**

**else** **{**

JOptionPane**.**showMessageDialog**(null,**

"Please select a friend first to edit their nickname!"**,**

"No Friend Selected"**,**

JOptionPane**.**WARNING\_MESSAGE**);**

**}**

**}** **else** **if** **(**source **==** flui**.**btnInviteToGame**)** **{**

String selected **=** flui**.**list**.**getSelectedValue**();**

flm**.**sendMessage**(**"inviteToGame, " **+** flui**.**currentPlayer**.**getUsername**()** **+** ", " **+** selected**);**

**}** **else** **if** **(**source **==** flui**.**btnAddFriend**)** **{**

String addedFriend **=** JOptionPane**.**showInputDialog**(null,** "Who do you want to add?"**);**

boolean canCont **=** **true;**

**for** **(**int i **=** 0**;** i **<** flui**.**usernames**.**size**()** **&&** canCont **==** **true;** i**++)** **{**

String**[]** temp **=** flui**.**usernames**.**get**(**i**).**split**(**" - "**);**

String selected **=** temp**[**0**];**

**if** **(**addedFriend**.**equals**(**selected**))** **{**

canCont **=** **false;**

**}**

**}**

**if** **(**canCont **==** **false)** **{**

JOptionPane**.**showMessageDialog**(null,** "The user is already in your friends list."**,** "Invalid Friend"**,** 0**);**

**}** **else** **{**

flm**.**sendMessage**(**"addFriend, " **+** flui**.**currentPlayer**.**getUsername**()** **+** ", " **+** addedFriend**);**

**}**

**}**

# 

**else** **if** **(**source **==** flui**.**btnBack**)** **{**

flui**.**frame**.**setVisible**(false);**

flui**.**mmui**.**setVisible**(true);**

flm**.**sendMessage**(**"CLOSE"**);**

flm**.**interrupt**();**

**}**

**}**

**}**

# FriendListModel

package Friend**;**

**import** UI**.**FriendListUI**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

**import** javax**.**swing**.**JOptionPane**;**

public class FriendListModel **extends** Thread **{**

private Socket socket**;**

private PrintWriter out**;**

private BufferedReader in**;**

private FriendListUI flui**;**

private String username**;**

public FriendListModel**(**FriendListUI ui**)** **{**

**try** **{**

flui **=** ui**;**

// socket connecting to server and input/output streams

username **=** flui**.**currentPlayer**.**getUsername**();**

socket **=** **new** Socket**(**"localhost"**,** 5555**);**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

out**.**flush**();**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

out**.**println**(**"friendList, " **+** username**);**

out**.**flush**();**

**}** **catch** **(**UnknownHostException e**)** **{**

System**.**out**.**println**(**"Unknown host: localhost.eng"**);**

System**.**exit**(**1**);**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"No I/O"**);**

System**.**exit**(**1**);**

**}**

start**();**

**}**

public void sendMessage**(**String msg**)** **{**

out**.**println**(**msg**);**

**}**

@Override

public void run**()** **{**

String received**;**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

received **=** in**.**readLine**();**

**if** **(**received **!=** **null)** **{**

**if** **(**received**.**startsWith**(**username**))** **{**

flui**.**clear**();**

/\*flui.listOfFriends\*/ String**[]** temp **=** received**.**split**(**", "**);**

**if** **(**temp**.**length **>** 1**)** **{**

**for** **(**int i **=** 1**;** i **<** temp**.**length**;** i **+=** 2**)** **{**

String elem **=** temp**[**i**]** **+** " - " **+** temp**[**i **+** 1**];**

flui**.**addElement**(**elem**);**

**}**

**}**

**}** **else** **if** **(**received**.**startsWith**(**"FAILIURE"**))** **{**

JOptionPane**.**showMessageDialog**(null,** "The user does not exist in the friends list."**,** "Invalid User"**,** 0**);**

**}** **else** **if** **(**received**.**startsWith**(**"inviteToGame"**))** **{**

out**.**println**(**received**);**

out**.**flush**();**

socket**.**close**();**

**}** **else** **if** **(**received**.**startsWith**(**"addFriend"**))** **{**

out**.**println**(**received**);**

out**.**flush**();**

socket**.**close**();**

**}** **else** **if** **(**received**.**startsWith**(**"CLOSE"**))** **{**

interrupt**();**

socket**.**close**();**

**}**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"I/O Error\n" **+** e**);**

**}**

**}**

**}**

**}**

# GameInvite

package Friend**;**

public class GameInvite **{**

public boolean GameInvite**(**String name**){**

// send invite to 'name'

// if 'name' accepted, start game

// return true;

// if 'name' declined, don't do anything

**return** **false;**

**}**

public void viewGameInvite**(){**

// view window with accept/decline friend invite.

**}**

public boolean respondToGameInvite**(**String name**){**

// if accepted invitation from 'name', start game

// return true;

// if declined invitation from 'name', don't do anything

**return** **false;**

**}**

**}**

# Game Package – Draw

package Game**;**

**import** javax**.**swing**.**JOptionPane**;**

public class Draw **extends** GameReport **{**

public Draw**(**String**[]** r**)** **{**

results **=** r**;**

**}**

public boolean report**()** **{**

int check **=** JOptionPane**.**showConfirmDialog**(null,** results**[**0**]** **+** " chose " **+** results**[**1**]** **+** ".\n"

**+** results**[**2**]** **+** " chose " **+** results**[**3**]** **+** ".\n\nIt was a draw!\n\nWould you like to find a new game?"**,** "Game Over"**,** JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**check **==** JOptionPane**.**YES\_OPTION**)** **{**

**return** **true;**

**}** **else** **{**

**return** **false;**

**}**

**}**

**}**

# GameController

package Game**;**

**import** UI**.**GameUI**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**io**.**IOException**;**

**import** javax**.**swing**.**JOptionPane**;**

public class GameController **implements** ActionListener **{**

private GameUI gui**;**

private GameModel gm**;**

public boolean canMove **=** **false;**

public GameController**(**GameUI ui**)** **throws** IOException**{**

gui **=** ui**;**

gm **=** **new** GameModel**(**gui**);**

**}**

public void closing**()** **throws** IOException

**{**

int check **=** JOptionPane**.**showConfirmDialog**(null,** "Are you sure you want to leave the match?\n\nThe round will be forfeited."**,** "Confirm Exit"**,** JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**check **==** JOptionPane**.**YES\_OPTION**)** **{**

gm**.**finish**();**

gui**.**close**();**

**}**

**}**

public void move**()**

**{**

canMove **=** **true;**

**}**

@Override

public void actionPerformed**(**ActionEvent event**)** **{**

Object source **=** event**.**getSource**();**

**if** **(**canMove **==** **true)** **{**

**if** **(**source **==** gui**.**rock**)** **{**

gui**.**setText**(**"Please wait..."**);**

gm**.**sendMessage**(**"MOVE rock"**);**

canMove **=** **false;**

**}** **else** **if** **(**source **==** gui**.**paper**)** **{**

gui**.**setText**(**"Please wait..."**);**

gm**.**sendMessage**(**"MOVE paper"**);**

canMove **=** **false;**

**}** **else** **if** **(**source **==** gui**.**scissors**)** **{**

gui**.**setText**(**"Please wait..."**);**

gm**.**sendMessage**(**"MOVE scissors"**);**

canMove **=** **false;**

**}**

**}**

**}**

**}**

# GameModel

package Game**;**

**import** Player**.**User**;**

**import** UI**.**GameUI**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

**import** javax**.**swing**.**JOptionPane**;**

public class GameModel **extends** Thread **{**

private GameUI gui**;**

private Socket socket**;**

private PrintWriter out**;**

private BufferedReader in**;**

private User currentUser**;**

private GameReport gr**;**

public GameModel**(**GameUI ui**)** **throws** IOException **{**

gui **=** ui**;**

**try** **{**

socket **=** **new** Socket**(**"localhost"**,** 4444**);**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

System**.**out**.**println**(**"Connected to server."**);**

currentUser **=** gui**.**getCurrentUser**();**

out**.**println**(**"DETAILS " **+** currentUser**.**getUsername**()** **+** "," **+** currentUser**.**getW**()** **+** "," **+** currentUser**.**getD**()** **+** "," **+** currentUser**.**getL**());**

**}** **catch** **(**UnknownHostException e**)** **{**

System**.**out**.**println**(**"Unknown host: localhost.eng"**);**

System**.**exit**(**1**);**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"\nNo I/O\n" **+** e**);**

System**.**exit**(**1**);**

**}**

start**();**

**}**

public void finish**()**

**{**

out**.**println**(**"FORFEIT"**);**

interrupt**();**

**}**

public void sendMessage**(**String msg**)** **{**

out**.**println**(**msg**);**

**}**

public void run**()** **{**

String response**;**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

response **=** in**.**readLine**();**

**if** **(**response **!=** **null)** **{**

**if** **(**response**.**startsWith**(**"MAKEMOVE"**))** **{**

JOptionPane**.**showMessageDialog**(null,** "You are playing against " **+** response**.**substring**(**9**)**

**+** ".\n\nBest of luck!"**,** "Match Briefing"**,** 1**);**

gui**.**setText**(**"Make a move!"**);**

gui**.**move**();**

**}** **else** **if** **(**response**.**startsWith**(**"RESULT"**))** **{**

String**[]** results **=** response**.**substring**(**7**).**split**(**","**);**

**if** **(**results**[**4**].**equals**(**currentUser**.**getUsername**()))** **{**

currentUser**.**newW**();**

gr **=** **new** Win**(**results**);**

gui**.**winsP**.**setText**(**"Wins: " **+** currentUser**.**getW**());**

**}** **else** **if** **(**results**[**4**].**equals**(**"draw"**))** **{**

gr **=** **new** Draw**(**results**);**

currentUser**.**newD**();**

gui**.**drawsP**.**setText**(**"Draws: " **+** currentUser**.**getD**());**

**}** **else** **{**

gr **=** **new** Loss**(**results**);**

currentUser**.**newL**();**

gui**.**lossesP**.**setText**(**"Losses: " **+** currentUser**.**getL**());**

**}**

boolean cont **=** gr**.**report**();**

**if** **(**cont **==** **true)** **{**

out**.**println**(**"NEWGAME"**);**

**}** **else** **{**

System**.**out**.**println**(**"Disconnecting from server."**);**

out**.**println**(**"QUIT"**);**

gui**.**close**();**

socket**.**close**();**

interrupt**();**

**}**

**}**

**}**

**}** **catch** **(**IOException e**)** **{**

JOptionPane**.**showMessageDialog**(null,** "I/O Error\n" **+** e**,** "Error"**,** 0**);**

System**.**exit**(**1**);**

**}**

**}**

**return;**

**}**

public void close**()** **throws** IOException **{**

socket**.**close**();**

interrupt**();**

**}**

**}**

# GameReport

package Game**;**

public abstract class GameReport **{**

public String**[]** results**;**

public abstract boolean report**();**

**}**

# Loss

package Game**;**

**import** javax**.**swing**.**JOptionPane**;**

public class Loss **extends** GameReport **{**

public Loss**(**String**[]** r**)** **{**

results **=** r**;**

**}**

public boolean report**()** **{**

int check **=** JOptionPane**.**showConfirmDialog**(null,** results**[**0**]** **+** " chose " **+** results**[**1**]** **+** ".\n"

**+** results**[**2**]** **+** " chose " **+** results**[**3**]** **+** ".\n\nYou lost! Hard luck!\n\nWould you like to find a new game?"**,** "Game Over"**,** JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**check **==** JOptionPane**.**YES\_OPTION**)** **{**

**return** **true;**

**}** **else** **{**

**return** **false;**

**}**

**}**

**}**

# Win

package Game**;**

**import** javax**.**swing**.**JOptionPane**;**

public class Win **extends** GameReport **{**

public Win**(**String**[]** r**)** **{**

results **=** r**;**

**}**

public boolean report**()** **{**

int check **=** JOptionPane**.**showConfirmDialog**(null,** results**[**0**]** **+** " chose " **+** results**[**1**]** **+** ".\n"

**+** results**[**2**]** **+** " chose " **+** results**[**3**]** **+** ".\n\nYou won! Congratulations!\n\nWould you like to find a new game?"**,** "Game Over"**,** JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**check **==** JOptionPane**.**YES\_OPTION**)** **{**

**return** **true;**

**}** **else** **{**

**return** **false;**

**}**

**}**

**}**

# Leaderboards Package – LeaderboardController

package Leaderboards**;**

**import** Player**.**User**;**

**import** UI**.**LeaderboardUI**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

public class LeaderboardController **{**

private LeaderboardUI component**;**

private LeaderboardModel model**;**

private User currentUser**;**

public LeaderboardController**(**LeaderboardUI lui**,** User currentUserTemp**)** **{**

currentUser **=** currentUserTemp**;**

component **=** lui**;**

model **=** **new** LeaderboardModel**();**

component**.**setVisible**(true);**

component**.**addAlphabeticButtonActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent evt**)** **{**

populateTable**(**"Alphabetic"**);**

**}**

**});**

component**.**addWinButtonActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent evt**)** **{**

populateTable**(**"Wins"**);**

**}**

**});**

component**.**addWLRatioButtonActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent evt**)** **{**

populateTable**(**"WLRatio"**);**

**}**

**});**

component**.**addFilterButtonActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent evt**)** **{**

String text **=** component**.**filterTextArea**.**getText**();**

populateTable**(**text**);**

**}**

**});**

populateTable**(**"raw"**);**

**}**

private void populateTable**(**String sort**)** **{**

component**.**populateTable**(**model**.**getEntries**(**sort**));**

**}**

**}**

# LeaderboardModel

package Leaderboards**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Comparator**;**

public class LeaderboardModel **extends** Thread **{**

private static ArrayList**<**String**[]>** entries**;**

private boolean alphaSort **=** **false;**

private Socket socket**;**

private PrintWriter out**;**

private BufferedReader in**;**

public LeaderboardModel**()** **{**

entries **=** **new** ArrayList**<>();**

readFile**();**

**}**

public ArrayList**<**String**[]>** getEntries**(**String sort**)** **{**

**if** **(**"raw"**.**equals**(**sort**))** **{**

**return** entries**;**

**}**

**else** **if** **(**"Alphabetic"**.**equals**(**sort**))** **{**

**if** **(**alphaSort **==** **false)** **{**

java**.**util**.**Collections**.**sort**(**entries**,** **new** Comparator**<**String**[]>()** **{**

@Override

public int compare**(**String**[]** array1**,** String**[]** array2**)** **{**

**return** array1**[**0**].**compareTo**(**array2**[**0**]);**

**}**

**});**

alphaSort **=** **true;**

**}**

**else** **{**

java**.**util**.**Collections**.**sort**(**entries**,** **new** Comparator**<**String**[]>()** **{**

@Override

public int compare**(**String**[]** array1**,** String**[]** array2**)** **{**

**return** array2**[**0**].**compareTo**(**array1**[**0**]);**

**}**

**});**

alphaSort **=** **false;**

**}**

**}**

**else** **if** **(**"Wins"**.**equals**(**sort**))** **{**

java**.**util**.**Collections**.**sort**(**entries**,** **new** Comparator**<**String**[]>()** **{**

@Override

public int compare**(**String**[]** array1**,** String**[]** array2**)** **{**

**return** Integer**.**parseInt**(**array2**[**1**])** **-** Integer**.**parseInt**(**array1**[**1**]);**

**}**

**});**

alphaSort **=** **false;**

**}**

**else** **if** **(**"WLRatio"**.**equals**(**sort**))** **{**

java**.**util**.**Collections**.**sort**(**entries**,** **new** Comparator**<**String**[]>()** **{**

@Override

public int compare**(**String**[]** array1**,** String**[]** array2**)** **{**

**return** Double**.**compare**(**Double**.**parseDouble**(**array2**[**3**]),** Double**.**parseDouble**(**array1**[**3**]));**

**}**

**});**

alphaSort **=** **false;**

**}**

**else** **{**

**if** **(**sort**.**length**()** **<** 1**)** **{**

**return** entries**;**

**}**

**else** **{**

String**[]** filter **=** sort**.**split**(**","**);**

String**[]** temp**;**

ArrayList**<**String**[]>** entriesTemp **=** **new** ArrayList**<>();**

**for(**int i **=** 0**;**i **<** entries**.**size**();**i**++)** **{**

temp **=** entries**.**get**(**i**);**

**if** **(**java**.**util**.**Arrays**.**asList**(**filter**).**contains**(**temp**[**0**]))** **{**

entriesTemp**.**add**(**temp**);**

**}**

**}**

alphaSort **=** **false;**

**return** entriesTemp**;**

**}**

**}**

**return** entries**;**

**}**

private double computeWLRatio**(**String**[]** temp**)** **{**

double number **=** **((**Double**.**parseDouble**(**temp**[**1**])** **/** **(**Double**.**parseDouble**(**temp**[**1**])** **+** Double**.**parseDouble**(**temp**[**2**]))));**

number **=** Math**.**round**(**number **\*** 100**);**

number **=** number**/**100**;**

**return** number**;**

**}**

private void readFile**()** **{**

**try** **{**

socket **=** **new** Socket**(**"localhost"**,** 5555**);**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

System**.**out**.**println**(**"Connected to server."**);**

**}** **catch** **(**UnknownHostException e**)** **{**

System**.**out**.**println**(**"Unknown host: localhost.eng"**);**

System**.**exit**(**1**);**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"\nNo I/O\n" **+** e**);**

System**.**exit**(**1**);**

**}**

out**.**println**(**"LEADERBOARDDATA"**);**

entries**.**clear**();**

start**();**

**}**

@Override

public void run**()** **{**

String response**;**

String wl**;**

**try** **{**

response **=** in**.**readLine**();**

String**[]** temp **=** response**.**split**(**","**);**

**for** **(**int i **=** 0**;** i **<** temp**.**length**;** i**++)** **{**

String**[]** line **=** temp**[**i**].**split**(**" "**);**

wl **=** Double**.**toString**(**computeWLRatio**(**line**));**

String**[]** entryLine **=** **{**line**[**0**],** line**[**1**],** line**[**2**],** wl**};**

entries**.**add**(**entryLine**);**

**}**

socket**.**close**();**

interrupt**();**

**}**

**catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"\nNo I/O\n" **+** e**);**

**}**

**}**

**}**

# LogIn Package – LogInController

package LogIn**;**

**import** UI**.**LogInUI**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

public class LogInController **implements** ActionListener **{**

private LogInUI liui**;**

private LogInModel lim**;**

public LogInController**(**LogInUI ui**)**

**{**

liui **=** ui**;**

lim **=** **new** LogInModel**(**liui**);**

**}**

public void logIn**(**String**[]** credentials**)** **{**

**if** **(!(**credentials**[**0**].**equals**(**""**))** **&&** **!(**credentials**[**1**].**equals**(**""**)))** **{**

lim**.**sendMessage**(**"LOGIN " **+** credentials**[**0**]** **+** "," **+** credentials**[**1**]);**

lim**.**setUsername**(**credentials**[**0**]);**

**}** **else** **{**

liui**.**empty**();**

**}**

**}**

public void register**(**String**[]** credentials**)**

**{**

**if** **(!(**credentials**[**0**].**equals**(**""**))** **&&** **!(**credentials**[**1**].**equals**(**""**)))** **{**

lim**.**sendMessage**(**"REGISTER " **+** credentials**[**0**]** **+** "," **+** credentials**[**1**]);**

lim**.**setUsername**(**credentials**[**0**]);**

**}** **else** **{**

liui**.**empty**();**

**}**

**}**

public void actionPerformed**(**ActionEvent event**)** **{**

Object source **=** event**.**getSource**();**

**if** **(**source **==** liui**.**logIn**)** **{**

String**[]** credentials **=** **{**liui**.**uNameI**.**getText**(),** String**.**valueOf**(**liui**.**passI**.**getPassword**())};**

logIn**(**credentials**);**

**}** **else** **if** **(**source **==** liui**.**newUser**)** **{**

String**[]** credentials **=** **{**liui**.**uNameI**.**getText**(),** String**.**valueOf**(**liui**.**passI**.**getPassword**())};**

register**(**credentials**);**

**}**

**}**

**}**

# LogInModel

package LogIn**;**

**import** Player**.**User**;**

**import** UI**.**LogInUI**;**

**import** UI**.**MainMenuUIFactory**;**

**import** UI**.**UIFactory**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** java**.**net**.**Socket**;**

public class LogInModel **extends** Thread **{**

private Socket socket**;**

private PrintWriter out**;**

private BufferedReader in**;**

private String username**;**

private LogInUI liui**;**

public LogInModel**(**LogInUI ui**)** **{**

liui **=** ui**;**

**try** **{**

socket **=** **new** Socket**(**"localhost"**,** 5555**);**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

System**.**out**.**println**(**"Connected to server."**);**

**}** **catch** **(**Exception e**)** **{**

JOptionPane**.**showMessageDialog**(null,** "The server is currently offline. Please try connecting later."**,** "Server Offline"**,** 1**);**

System**.**exit**(**1**);**

**}**

start**();**

**}**

public void sendMessage**(**String msg**)** **{**

out**.**println**(**msg**);**

**}**

public void setUsername**(**String user**)**

**{**

username **=** user**;**

**}**

public void run**()** **{**

String response**;**

**while** **(!**Thread**.**interrupted**())** **{**

**try** **{**

response **=** in**.**readLine**();**

**if** **(**response**.**startsWith**(**"SUCCESS"**))** **{**

String**[]** uA **=** response**.**substring**(**8**).**split**(**","**);**

User currentUser **=** **new** User**(**username**,** Integer**.**parseInt**(**uA**[**0**]),** Integer**.**parseInt**(**uA**[**1**]),** Integer**.**parseInt**(**uA**[**2**]));**

JOptionPane**.**showMessageDialog**(null,** "Welcome, " **+** username **+** "!\n\nYou have " **+** currentUser**.**getW**()** **+** " wins, " **+** currentUser**.**getD**()** **+** " draws and " **+** currentUser**.**getL**()** **+** " losses."**,** "Login Successful"**,** 1**);**

liui**.**setVisible**(false);**

UIFactory mmui **=** **new** MainMenuUIFactory**(**currentUser**);**

mmui**.**create**();**

interrupt**();**

socket**.**close**();**

**}** **else** **if** **(**response**.**startsWith**(**"FAILIURE"**))** **{**

liui**.**setText**(**response**.**substring**(**9**));**

**}** **else** **if** **(**response**.**startsWith**(**"REGISTERED"**))** **{**

User currentUser **=** **new** User**(**username**,** 0**,** 0**,** 0**);**

JOptionPane**.**showMessageDialog**(null,** "Welcome, " **+** username **+** "!\n\nYou have " **+** currentUser**.**getW**()** **+** " wins, " **+** currentUser**.**getD**()** **+** " draws and " **+** currentUser**.**getL**()** **+** " losses."**,** "Login Successful"**,** 1**);**

liui**.**setVisible**(false);**

UIFactory mmui **=** **new** MainMenuUIFactory**(**currentUser**);**

mmui**.**create**();**

interrupt**();**

socket**.**close**();**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"\nNo I/O\n" **+** e**);**

**}**

**}**

System**.**out**.**println**(**"Thread stopped."**);**

**return;**

**}**

**}**

# LogInMenu Package – MainMenuController

package MainMenu**;**

**import** UI**.**ChatRoomUIFactory**;**

**import** UI**.**FriendListUIFactory**;**

**import** UI**.**GameUIFactory**;**

**import** UI**.**LeaderboardUIFactory**;**

**import** UI**.**LogInUIFactory**;**

**import** UI**.**MainMenuUI**;**

**import** UI**.**UIFactory**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**io**.**IOException**;**

public class MainMenuController **implements** ActionListener **{**

private MainMenuUI mmui**;**

public boolean inChat **=** **false;**

UIFactory uif**;**

public MainMenuController**(**MainMenuUI ui**)** **{**

mmui **=** ui**;**

**}**

@Override

public void actionPerformed**(**ActionEvent event**)** **{**

**try** **{**

Object source **=** event**.**getSource**();**

**if** **(**source **==** mmui**.**joinGame**)** **{**

uif **=** **new** GameUIFactory**(**mmui**,** mmui**.**currentUser**);**

uif**.**create**();**

mmui**.**setVisible**(false);**

**}** **else** **if** **(**source **==** mmui**.**friendList**)** **{**

uif **=** **new** FriendListUIFactory**(**mmui**.**currentUser**,** mmui**);**

uif**.**create**();**

//mmui.setVisible(false);

**}** **else** **if** **(**source **==** mmui**.**leaderboards**)** **{**

uif **=** **new** LeaderboardUIFactory**(**mmui**.**currentUser**);**

uif**.**create**();**

**}** **else** **if** **(**source **==** mmui**.**chatRoom **&&** inChat **==** **false)** **{**

uif **=** **new** ChatRoomUIFactory**(**mmui**.**currentUser**,** mmui**);**

uif**.**create**();**

inChat **=** **true;**

**}** **else** **if** **(**source **==** mmui**.**logOut**)** **{**

uif **=** **new** LogInUIFactory**();**

uif**.**create**();**

mmui**.**setVisible**(false);**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"I/O Error\n" **+** e**);**

**}**

**}**

**}**

# Player Package – User

package Player**;**

public class User **{**

private int wins **=** 0**,** draws **=** 0**,** losses **=** 0**;**

private String username **=** ""**,** password **=** ""**;**

public User**(**String name**,** String pass**)** **{**

username **=** name**;**

password **=** pass**;**

**}**

public User**(**String name**,** String pass**,** int w**,** int d**,** int l**)** **{**

username **=** name**;**

password **=** pass**;**

wins **=** w**;**

draws **=** d**;**

losses **=** l**;**

**}**

public User**(**String name**,** int w**,** int d**,** int l**)** **{**

username **=** name**;**

wins **=** w**;**

draws **=** d**;**

losses **=** l**;**

**}**

public String getUsername**()** **{**

**return** username**;**

**}**

public String getPassword**()** **{**

**return** password**;**

**}**

public void setPassword**(**String pass**)** **{**

password **=** pass**;**

**}**

public int getW**()** **{**

**return** wins**;**

**}**

public int getD**()** **{**

**return** draws**;**

**}**

public int getL**()** **{**

**return** losses**;**

**}**

public void newW**()** **{**

wins**++;**

**}**

public void newD**()** **{**

draws**++;**

**}**

public void newL**()** **{**

losses**++;**

**}**

**}**

# UI Package – ChatRoomUI

package UI**;**

**import** Player**.\*;**

**import** Chat**.\*;**

**import** java**.**awt**.**Color**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JTextField**;**

**import** java**.**io**.**IOException**;**

**import** javax**.**swing**.\*;**

public class ChatRoomUI **{**

public JFrame chatWindow**;**

public JButton jbSend**;**

private JLabel jlMessage**,** jlConversation**,** jlOnline**;**

public JTextField jtfMessage**;**

public JTextArea jtaConversation**;**

private JScrollPane jspConversation**,** jspOnline**;**

public JList jliOnline**;**

public User currentUser**;**

private ChatRoomController crc**;**

private String**[]** onlineUsers**;**

private MainMenuUI mmui**;**

public ChatRoomUI**(**User u**,** MainMenuUI ui**)** **throws** IOException **{**

currentUser **=** u**;**

mmui **=** ui**;**

jlOnline **=** **new** JLabel**(**"Currently Online"**);**

jlOnline**.**setHorizontalAlignment**(**SwingConstants**.**CENTER**);**

jliOnline **=** **new** JList**();**

jliOnline**.**setForeground**(**Color**.**blue**);**

jtaConversation **=** **new** JTextArea**();**

jtaConversation**.**setColumns**(**20**);**

jtaConversation**.**setForeground**(**Color**.**blue**);**

jtaConversation**.**setLineWrap**(true);**

jtaConversation**.**setRows**(**5**);**

jtaConversation**.**setEditable**(false);**

crc **=** **new** ChatRoomController**(this);**

chatWindow **=** **new** JFrame**(**"Chat Room"**);**

chatWindow**.**setSize**(**650**,** 450**);**

chatWindow**.**setBackground**(**Color**.**lightGray**);**

chatWindow**.**setLayout**(null);**

jbSend **=** **new** JButton**(**"Send"**);**

jbSend**.**addActionListener**(**crc**);**

jbSend**.**setBackground**(**Color**.**blue**);**

jbSend**.**setForeground**(**Color**.**white**);**

jlMessage **=** **new** JLabel**(**"Message: "**);**

jtfMessage **=** **new** JTextField**(**20**);**

jtfMessage**.**requestFocus**();**

jlConversation **=** **new** JLabel**(**"Conversation"**);**

jlConversation**.**setHorizontalAlignment**(**SwingConstants**.**CENTER**);**

jspConversation **=** **new** JScrollPane**();**

jspConversation**.**setHorizontalScrollBarPolicy**(**ScrollPaneConstants**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

jspConversation**.**setVerticalScrollBarPolicy**(**ScrollPaneConstants**.**VERTICAL\_SCROLLBAR\_ALWAYS**);**

jspConversation**.**setViewportView**(**jtaConversation**);**

jspOnline **=** **new** JScrollPane**();**

jspOnline**.**setHorizontalScrollBarPolicy**(**ScrollPaneConstants**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

jspOnline**.**setVerticalScrollBarPolicy**(**ScrollPaneConstants**.**VERTICAL\_SCROLLBAR\_ALWAYS**);**

jspOnline**.**setViewportView**(**jliOnline**);**

chatWindow**.**add**(**jbSend**);**

chatWindow**.**add**(**jlMessage**);**

chatWindow**.**add**(**jtfMessage**);**

chatWindow**.**add**(**jlConversation**);**

chatWindow**.**add**(**jspConversation**);**

chatWindow**.**add**(**jlOnline**);**

chatWindow**.**add**(**jspOnline**);**

jbSend**.**setBounds**(**250**,** 40**,** 71**,** 10**);**

jlMessage**.**setBounds**(**10**,** 10**,** 60**,** 20**);**

jtfMessage**.**setBounds**(**70**,** 4**,** 260**,** 30**);**

jlConversation**.**setBounds**(**100**,** 70**,** 140**,** 16**);**

jspConversation**.**setBounds**(**10**,** 90**,** 330**,** 180**);**

jlOnline**.**setBounds**(**350**,** 70**,** 150**,** 16**);**

jspOnline**.**setBounds**(**350**,** 90**,** 150**,** 180**);**

chatWindow**.**setDefaultCloseOperation**(**WindowConstants**.**DO\_NOTHING\_ON\_CLOSE**);**

chatWindow**.**addWindowListener**(new** java**.**awt**.**event**.**WindowAdapter**()** **{**

@Override

public void windowClosing**(**java**.**awt**.**event**.**WindowEvent windowEvent**)** **{**

mmui**.**chatClosed**();**

chatWindow**.**setVisible**(false);**

crc**.**quit**();**

**}**

**});**

chatWindow**.**setVisible**(true);**

**}**

public User getCurrentUser**()** **{**

**return** currentUser**;**

**}**

public String getMessage**()** **{**

**return** jtfMessage**.**getText**();**

**}**

public void recieveMessage**(**Message message**)** **{**

String**[]** temp **=** message**.**get**();**

jtaConversation**.**append**(**temp**[**0**]** **+** ": " **+** temp**[**1**]** **+** "\n"**);**

**}**

public void clearTF**()** **{**

jtfMessage**.**setText**(**""**);**

jtfMessage**.**requestFocus**();**

**}**

public void updateUsers**(**String**[]** usernames**)** **{**

jliOnline**.**setListData**(**usernames**);**

**}**

**}**

# ChatRoomUIFactory

package UI**;**

**import** Player**.**User**;**

**import** java**.**io**.**IOException**;**

public class ChatRoomUIFactory **extends** UIFactory **{**

User currentUser**;**

MainMenuUI mmui**;**

public ChatRoomUIFactory**(**User cu**,** MainMenuUI ui**)** **{**

currentUser **=** cu**;**

mmui **=** ui**;**

**}**

public void create**()** **throws** IOException **{**

ChatRoomUI afui **=** **new** ChatRoomUI**(**currentUser**,** mmui**);**

**}**

**}**

# FriendListUI

package UI**;**

**import** Friend**.\*;**

**import** javax**.**swing**.\*;**

**import** java**.**io**.\*;**

**import** Player**.**User**;**

**import** java**.**util**.**ArrayList**;**

public class FriendListUI **extends** Thread **{**

public JFrame frame**;**

public JButton btnEditNickname**,** btnInviteToGame**,** btnAddFriend**,** btnBack**;**

public JList**<**String**>** list**;**

public User currentPlayer**;**

public String**[]** listOfFriends**;**

public DefaultListModel**<**String**>** listModel**;**

public MainMenuUI mmui**;**

private FriendListController flc**;**

public ArrayList**<**String**>** usernames**;**

/\*\*

\* @throws IOException

\*/

public FriendListUI**(**User cp**,** MainMenuUI ui**)** **{**

currentPlayer **=** cp**;**

mmui **=** ui**;**

listModel **=** **new** DefaultListModel**<**String**>();**

list **=** **new** JList**<**String**>(**listModel**);**

list**.**setBounds**(**22**,** 42**,** 252**,** 348**);**

list**.**setSelectionMode**(**ListSelectionModel**.**SINGLE\_SELECTION**);**

frame **=** **new** JFrame**(**"Friend List"**);**

frame**.**setBounds**(**100**,** 100**,** 451**,** 450**);**

frame**.**setDefaultCloseOperation**(**JFrame**.**EXIT\_ON\_CLOSE**);**

frame**.**getContentPane**().**setLayout**(null);**

JLabel lblYourFriends **=** **new** JLabel**(**"Your friends:"**);**

lblYourFriends**.**setBounds**(**22**,** 13**,** 151**,** 16**);**

frame**.**getContentPane**().**add**(**lblYourFriends**);**

// friends list display

frame**.**getContentPane**().**add**(**list**);**

flc **=** **new** FriendListController**(this);**

btnEditNickname **=** **new** JButton**(**"Edit nickname"**);**

btnEditNickname**.**setBounds**(**286**,** 144**,** 124**,** 48**);**

btnEditNickname**.**addActionListener**(**flc**);**

frame**.**getContentPane**().**add**(**btnEditNickname**);**

btnInviteToGame **=** **new** JButton**(**"Invite to game"**);**

btnInviteToGame**.**setBounds**(**286**,** 205**,** 124**,** 48**);**

btnInviteToGame**.**addActionListener**(**flc**);**

frame**.**getContentPane**().**add**(**btnInviteToGame**);**

btnAddFriend **=** **new** JButton**(**"Add new friend"**);**

btnAddFriend**.**setBounds**(**286**,** 39**,** 124**,** 49**);**

btnAddFriend**.**addActionListener**(**flc**);**

frame**.**getContentPane**().**add**(**btnAddFriend**);**

btnBack **=** **new** JButton**(**"Back"**);**

btnBack**.**setBounds**(**313**,** 358**,** 97**,** 32**);**

btnBack**.**addActionListener**(**flc**);**

frame**.**getContentPane**().**add**(**btnBack**);**

frame**.**setDefaultCloseOperation**(**WindowConstants**.**DO\_NOTHING\_ON\_CLOSE**);**

frame**.**addWindowListener**(new** java**.**awt**.**event**.**WindowAdapter**()** **{**

@Override

public void windowClosing**(**java**.**awt**.**event**.**WindowEvent windowEvent**)** **{**

frame**.**setVisible**(false);**

mmui**.**setVisible**(true);**

**}**

**});**

frame**.**setVisible**(true);**

**}**

public void clear**()**

**{**

listModel**.**clear**();**

usernames **=** **new** ArrayList**<**String**>();**

**}**

public void addElement**(**String s**)**

**{**

listModel**.**addElement**(**s**);**

usernames**.**add**(**s**);**

**}**

**}**

# FriendListUIFactory

package UI**;**

**import** Player**.**User**;**

public class FriendListUIFactory **extends** UIFactory **{**

User currentUser**;**

MainMenuUI mmui**;**

public FriendListUIFactory**(**User cu**,** MainMenuUI ui**)** **{**

mmui **=** ui**;**

currentUser **=** cu**;**

**}**

public void create**()** **{**

FriendListUI flui **=** **new** FriendListUI**(**currentUser**,** mmui**);**

**}**

**}**

# GameUI

package UI**;**

**import** Game**.**GameController**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**GridLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JPasswordField**;**

**import** javax**.**swing**.**JTextField**;**

**import** Player**.\*;**

**import** java**.**io**.**IOException**;**

**import** javax**.**swing**.**SwingConstants**;**

**import** javax**.**swing**.**WindowConstants**;**

public class GameUI **extends** Thread **{**

private JFrame frame**;**

public JButton rock**,** paper**,** scissors**;**

private JTextField uNameI**;**

private JPasswordField passI**;**

public JLabel info**,** winsP**,** drawsP**,** lossesP**;**

private User currentUser**;**

private boolean canMove **=** **false;**

private MainMenuUI mmui**;**

private GameController gc**;**

public GameUI**(**MainMenuUI ui**,** User u**)** **throws** IOException **{**

currentUser **=** u**;**

mmui **=** ui**;**

gc **=** **new** GameController**(this);**

frame **=** **new** JFrame**(**"Main Menu"**);**

frame**.**setSize**(**600**,** 539**);**

JPanel infoPanel **=** **new** JPanel**(new** GridLayout**(**1**,** 2**));**

JPanel buttons **=** **new** JPanel**(new** GridLayout**(**3**,** 1**));**

JPanel stats **=** **new** JPanel**(new** GridLayout**(**3**,** 1**));**

info **=** **new** JLabel**(**"Waiting on opponent..."**,** SwingConstants**.**CENTER**);**

winsP **=** **new** JLabel**(**"Wins: " **+** currentUser**.**getW**(),** SwingConstants**.**CENTER**);**

drawsP **=** **new** JLabel**(**"Draws: " **+** currentUser**.**getD**(),** SwingConstants**.**CENTER**);**

lossesP **=** **new** JLabel**(**"Losses: " **+** currentUser**.**getL**(),** SwingConstants**.**CENTER**);**

infoPanel**.**setBackground**(**Color**.**lightGray**);**

info**.**setForeground**(**Color**.**WHITE**);**

info**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

winsP**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

drawsP**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

lossesP**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

# 

rock **=** **new** JButton**(**"Rock"**);**

rock**.**addActionListener**(**gc**);**

rock**.**setForeground**(**Color**.**WHITE**);**

rock**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 22**));**

paper **=** **new** JButton**(**"Paper"**);**

paper**.**addActionListener**(**gc**);**

paper**.**setForeground**(**Color**.**WHITE**);**

paper**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 22**));**

scissors **=** **new** JButton**(**"Scissors"**);**

scissors**.**addActionListener**(**gc**);**

scissors**.**setForeground**(**Color**.**WHITE**);**

scissors**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 22**));**

infoPanel**.**setPreferredSize**(new** Dimension**(**600**,** 150**));**

buttons**.**setPreferredSize**(new** Dimension**(**600**,** 200**));**

stats**.**setPreferredSize**(new** Dimension**(**600**,** 150**));**

frame**.**add**(**infoPanel**,** BorderLayout**.**NORTH**);**

frame**.**add**(**buttons**,** BorderLayout**.**CENTER**);**

stats**.**add**(**winsP**);**

stats**.**add**(**drawsP**);**

stats**.**add**(**lossesP**);**

infoPanel**.**add**(**info**);**

infoPanel**.**add**(**stats**);**

buttons**.**add**(**rock**);**

buttons**.**add**(**paper**);**

buttons**.**add**(**scissors**);**

rock**.**setBackground**(**Color**.**red**);**

paper**.**setBackground**(**Color**.**blue**);**

scissors**.**setBackground**(**Color**.**green**);**

frame**.**setDefaultCloseOperation**(**WindowConstants**.**DO\_NOTHING\_ON\_CLOSE**);**

frame**.**addWindowListener**(new** java**.**awt**.**event**.**WindowAdapter**()** **{**

@Override

public void windowClosing**(**java**.**awt**.**event**.**WindowEvent windowEvent**)** **{**

**try** **{**

gc**.**closing**();**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"I/O Error\n" **+** e**);**

**}**

**}**

**});**

frame**.**setVisible**(true);**

start**();**

**}**

public void setText**(**String text**)** **{**

info**.**setText**(**text**);**

**}**

public User getCurrentUser**()** **{**

**return** currentUser**;**

**}**

public void move**()** **{**

gc**.**move**();**

**}**

public void close**()** **{**

frame**.**setVisible**(false);**

mmui**.**setVisible**(true);**

**}**

**}**

# GameUIFactory

package UI**;**

**import** Player**.**User**;**

**import** java**.**io**.**IOException**;**

public class GameUIFactory **extends** UIFactory **{**

User currentUser**;**

MainMenuUI mmui**;**

public GameUIFactory**(**MainMenuUI ui**,** User cu**)** **{**

currentUser **=** cu**;**

mmui **=** ui**;**

**}**

public void create**()** **throws** IOException **{**

GameUI gui **=** **new** GameUI**(**mmui**,** currentUser**);**

**}**

**}**

# LeaderboardUI

package UI**;**

**import** Leaderboards**.**LeaderboardController**;**

**import** Player**.**User**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**util**.**ArrayList**;**

public class LeaderboardUI **extends** javax**.**swing**.**JFrame **{**

/\*\*

\* Creates new form LeaderboardUI

\*/

private User currentUser**;**

public LeaderboardUI**(**User currentUser**)** **{**

**this.**currentUser **=** currentUser**;**

initComponents**();**

**}**

private void initComponents**()** **{**

currentStandings **=** **new** javax**.**swing**.**JLabel**();**

sortMethod **=** **new** javax**.**swing**.**JLabel**();**

alphabeticSortButton **=** **new** javax**.**swing**.**JButton**();**

jScrollPane1 **=** **new** javax**.**swing**.**JScrollPane**();**

leaderboardTable **=** **new** javax**.**swing**.**JTable**();**

winSortButton **=** **new** javax**.**swing**.**JButton**();**

wLRatioSortButton **=** **new** javax**.**swing**.**JButton**();**

filter **=** **new** javax**.**swing**.**JLabel**();**

filterTextArea **=** **new** javax**.**swing**.**JTextField**();**

filterButton **=** **new** javax**.**swing**.**JButton**();**

setDefaultCloseOperation**(**javax**.**swing**.**WindowConstants**.**DISPOSE\_ON\_CLOSE**);**

**this.**setTitle**(**"Leaderboard"**);**

leaderboardTable**.**setModel**(new** javax**.**swing**.**table**.**DefaultTableModel**(**

**new** Object **[][]** **{**

**},**

**new** String **[]** **{**

"Name"**,** "Wins"**,** "Losses"**,** "W/L Ratio"

**})** **{**

Class**[]** types **=** **new** Class **[]** **{**

java**.**lang**.**String**.**class**,** java**.**lang**.**String**.**class**,** java**.**lang**.**String**.**class**,** java**.**lang**.**String**.**class

**};**

boolean**[]** canEdit **=** **new** boolean **[]** **{**

**false,** **false,** **false,** **false**

**};**

@Override

public Class getColumnClass**(**int columnIndex**)** **{**

**return** types **[**columnIndex**];**

**}**

@Override

public boolean isCellEditable**(**int rowIndex**,** int columnIndex**)** **{**

**return** canEdit **[**columnIndex**];**

**}**

**});**

jScrollPane1**.**setViewportView**(**leaderboardTable**);**

**if** **(**leaderboardTable**.**getColumnModel**().**getColumnCount**()** **>** 0**)** **{**

leaderboardTable**.**getColumnModel**().**getColumn**(**0**).**setResizable**(false);**

leaderboardTable**.**getColumnModel**().**getColumn**(**1**).**setResizable**(false);**

**}**

currentStandings**.**setText**(**"Current standings"**);**

sortMethod**.**setText**(**"Sort Method"**);**

alphabeticSortButton**.**setText**(**"Sort Alphabetically"**);**

alphabeticSortButton**.**setBackground**(**Color**.**lightGray**);**

winSortButton**.**setText**(**"Sort By Wins"**);**

winSortButton**.**setBackground**(**Color**.**lightGray**);**

wLRatioSortButton**.**setText**(**"Sort by W/L Ratio"**);**

wLRatioSortButton**.**setBackground**(**Color**.**lightGray**);**

filter**.**setText**(**"Filter"**);**

filterTextArea**.**setText**(**"Enter names separated by ,"**);**

filterButton**.**setText**(**"Apply Filter"**);**

filterButton**.**setBackground**(**Color**.**lightGray**);**

javax**.**swing**.**GroupLayout layout **=** **new** javax**.**swing**.**GroupLayout**(**getContentPane**());**

getContentPane**().**setLayout**(**layout**);**

layout**.**setHorizontalGroup**(**

layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** layout**.**createSequentialGroup**()**

**.**addContainerGap**()**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addComponent**(**jScrollPane1**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 279**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**currentStandings**))**

**.**addGap**(**18**,** 18**,** 18**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** **false)**

**.**addComponent**(**filter**)**

**.**addComponent**(**sortMethod**)**

**.**addComponent**(**wLRatioSortButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**winSortButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**alphabeticSortButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**filterTextArea**)**

**.**addComponent**(**filterButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**))**

**.**addContainerGap**(**javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**))**

**);**

layout**.**setVerticalGroup**(**

layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addContainerGap**()**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** **false)**

**.**addComponent**(**currentStandings**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**sortMethod**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addComponent**(**alphabeticSortButton**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addComponent**(**winSortButton**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addComponent**(**wLRatioSortButton**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addComponent**(**filter**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**)**

**.**addComponent**(**filterTextArea**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**)**

**.**addComponent**(**filterButton**))**

**.**addComponent**(**jScrollPane1**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 275**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**))**

**.**addContainerGap**(**javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**))**

**);**

pack**();**

LeaderboardController lbc **=** **new** LeaderboardController**(this,** currentUser**);**

**}**

public javax**.**swing**.**JTextField filterTextArea**;**

private javax**.**swing**.**JButton alphabeticSortButton**;**

private javax**.**swing**.**JLabel currentStandings**;**

private javax**.**swing**.**JLabel filter**;**

private javax**.**swing**.**JButton filterButton**;**

private javax**.**swing**.**JScrollPane jScrollPane1**;**

private javax**.**swing**.**JTable leaderboardTable**;**

private javax**.**swing**.**JLabel sortMethod**;**

private javax**.**swing**.**JButton wLRatioSortButton**;**

private javax**.**swing**.**JButton winSortButton**;**

private javax**.**swing**.**table**.**DefaultTableModel model**;**

public void addAlphabeticButtonActionListener**(**ActionListener listener**)** **{**

alphabeticSortButton**.**addActionListener**(**listener**);**

**}**

public void addWinButtonActionListener**(**ActionListener listener**)** **{**

winSortButton**.**addActionListener**(**listener**);**

**}**

public void addWLRatioButtonActionListener**(**ActionListener listener**)** **{**

wLRatioSortButton**.**addActionListener**(**listener**);**

**}**

public void addFilterButtonActionListener**(**ActionListener listener**)** **{**

filterButton**.**addActionListener**(**listener**);**

**}**

public void populateTable**(**ArrayList**<**String**[]>** entries**)** **{**

model **=** **(**javax**.**swing**.**table**.**DefaultTableModel**)** leaderboardTable**.**getModel**();**

model**.**setRowCount**(**0**);**

**for(**int i **=** 0**;** i **<** entries**.**size**();** i**++)** **{**

String**[]** line **=** entries**.**get**(**i**);**

model**.**addRow**(**line**);**

**}**

**}**

**}**

# LeaderboardUIFactory

package UI**;**

**import** Player**.**User**;**

public class LeaderboardUIFactory **extends** UIFactory **{**

LeaderboardUI lbui**;**

User currentUser**;**

public LeaderboardUIFactory**(**User cu**)** **{**

currentUser **=** cu**;**

**}**

@Override

public void create**()** **{**

lbui **=** **new** LeaderboardUI**(**currentUser**);**

**}**

**}**

# LogInUI

package UI**;**

**import** LogIn**.**LogInController**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**GridLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JPasswordField**;**

**import** javax**.**swing**.**JTextField**;**

**import** javax**.**swing**.**SwingConstants**;**

public class LogInUI **{**

private JFrame frame**;**

public JButton logIn**,** newUser**;**

public JTextField uNameI**;**

public JPasswordField passI**;**

private JLabel info**,** uL**,** pL**;**

private String username**;**

private JPanel mainPanel**;**

private LogInController lic**;**

public LogInUI**()** **{**

frame **=** **new** JFrame**(**"Log In"**);**

lic **=** **new** LogInController**(this);**

frame**.**setSize**(**500**,** 400**);**

mainPanel **=** **new** JPanel**(new** BorderLayout**());**

JPanel inputPanel **=** **new** JPanel**(new** GridLayout**(**2**,** 1**));**

JPanel uPanel **=** **new** JPanel**(new** GridLayout**(**2**,** 1**));**

JPanel pPanel **=** **new** JPanel**(new** GridLayout**(**2**,** 1**));**

JPanel buttons **=** **new** JPanel**(new** GridLayout**(**1**,** 2**));**

info **=** **new** JLabel**(**"Welcome. Please enter your username and password."**,** SwingConstants**.**CENTER**);**

uL **=** **new** JLabel**(**"Username"**);**

pL **=** **new** JLabel**(**"Password"**);**

uNameI **=** **new** JTextField**();**

passI **=** **new** JPasswordField**();**

info**.**setForeground**(**Color**.**WHITE**);**

mainPanel**.**setBackground**(**Color**.**BLUE**);**

info**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

logIn **=** **new** JButton**(**"Log In"**);**

logIn**.**addActionListener**(**lic**);**

newUser **=** **new** JButton**(**"Register"**);**

newUser**.**addActionListener**(**lic**);**

frame**.**add**(**mainPanel**,** BorderLayout**.**NORTH**);**

frame**.**add**(**inputPanel**,** BorderLayout**.**CENTER**);**

frame**.**add**(**buttons**,** BorderLayout**.**SOUTH**);**

mainPanel**.**setPreferredSize**(new** Dimension**(**500**,** 100**));**

buttons**.**setPreferredSize**(new** Dimension**(**500**,** 100**));**

uPanel**.**setPreferredSize**(new** Dimension**(**500**,** 100**));**

pPanel**.**setPreferredSize**(new** Dimension**(**500**,** 100**));**

inputPanel**.**setPreferredSize**(new** Dimension**(**500**,** 200**));**

mainPanel**.**add**(**info**,** BorderLayout**.**CENTER**);**

uPanel**.**setBackground**(**Color**.**lightGray**);**

pPanel**.**setBackground**(**Color**.**lightGray**);**

uPanel**.**add**(**uL**);**

uPanel**.**add**(**uNameI**);**

pPanel**.**add**(**pL**);**

pPanel**.**add**(**passI**);**

inputPanel**.**add**(**uPanel**,** BorderLayout**.**NORTH**);**

inputPanel**.**add**(**pPanel**,** BorderLayout**.**SOUTH**);**

buttons**.**add**(**logIn**,** BorderLayout**.**EAST**);**

buttons**.**add**(**newUser**,** BorderLayout**.**WEST**);**

logIn**.**setBackground**(**Color**.**lightGray**);**

newUser**.**setBackground**(**Color**.**lightGray**);**

frame**.**setVisible**(true);**

**}**

public void empty**()** **{**

mainPanel**.**setBackground**(**Color**.**red**);**

info**.**setText**(**"Username/password cannot be empty or contain commas!"**);**

**}**

public void setText**(**String msg**)** **{**

mainPanel**.**setBackground**(**Color**.**red**);**

info**.**setText**(**msg**);**

**}**

public void setVisible**(**boolean b**)** **{**

frame**.**setVisible**(**b**);**

**}**

**}**

# LogInUIFactory

package UI**;**

public class LogInUIFactory **extends** UIFactory **{**

public LogInUIFactory**()** **{**

**}**

public void create**()** **{**

LogInUI liui **=** **new** LogInUI**();**

**}**

**}**

# MainMenuUI

package UI**;**

**import** MainMenu**.**MainMenuController**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**GridLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JPasswordField**;**

**import** javax**.**swing**.**JTextField**;**

**import** Player**.\*;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**SwingConstants**;**

**import** javax**.**swing**.**WindowConstants**;**

public class MainMenuUI **{**

private JFrame frame**;**

public JButton joinGame**,** friendList**,** leaderboards**,** chatRoom**,** logOut**;**

private JTextField uNameI**;**

private JPasswordField passI**;**

private JLabel info**,** uL**,** pL**;**

public User currentUser**;**

private MainMenuController mmc**;**

public MainMenuUI**(**User u**)** **{**

currentUser **=** u**;**

frame **=** **new** JFrame**(**"Main Menu"**);**

frame**.**setSize**(**500**,** 439**);**

JPanel mainPanel **=** **new** JPanel**(new** GridLayout**(**2**,** 1**));**

JPanel topBar **=** **new** JPanel**(new** GridLayout**(**1**,** 2**));**

JPanel infoPanel **=** **new** JPanel**(new** BorderLayout**());**

JPanel buttons **=** **new** JPanel**(new** GridLayout**(**2**,** 2**));**

JLabel usernameL **=** **new** JLabel**(**"Welcome, " **+** currentUser**.**getUsername**()** **+** "!"**,** SwingConstants**.**CENTER**);**

info **=** **new** JLabel**(**"Please choose how you wish to proceed."**,** SwingConstants**.**CENTER**);**

uNameI **=** **new** JTextField**();**

passI **=** **new** JPasswordField**();**

info**.**setForeground**(**Color**.**WHITE**);**

//usernameL.setForeground(Color.WHITE);

infoPanel**.**setBackground**(**Color**.**BLUE**);**

info**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

usernameL**.**setFont**(new** Font**(**"Quikhand"**,** Font**.**BOLD**,** 17**));**

mmc **=** **new** MainMenuController**(this);**

logOut **=** **new** JButton**(**"Log Out"**);**

logOut**.**addActionListener**(**mmc**);**

joinGame **=** **new** JButton**(**"Join Game"**);**

joinGame**.**addActionListener**(**mmc**);**

friendList **=** **new** JButton**(**"Friend List"**);**

friendList**.**addActionListener**(**mmc**);**

leaderboards **=** **new** JButton**(**"Leaderboards"**);**

leaderboards**.**addActionListener**(**mmc**);**

chatRoom **=** **new** JButton**(**"Chat Room"**);**

chatRoom**.**addActionListener**(**mmc**);**

infoPanel**.**setPreferredSize**(new** Dimension**(**500**,** 100**));**

topBar**.**setPreferredSize**(new** Dimension**(**500**,** 100**));**

buttons**.**setPreferredSize**(new** Dimension**(**500**,** 200**));**

frame**.**add**(**mainPanel**,** BorderLayout**.**NORTH**);**

frame**.**add**(**buttons**,** BorderLayout**.**SOUTH**);**

mainPanel**.**add**(**topBar**);**

mainPanel**.**add**(**infoPanel**);**

infoPanel**.**add**(**info**);**

topBar**.**add**(**usernameL**);**

topBar**.**add**(**logOut**);**

buttons**.**add**(**joinGame**);**

buttons**.**add**(**friendList**);**

buttons**.**add**(**leaderboards**);**

buttons**.**add**(**chatRoom**);**

logOut**.**setBackground**(**Color**.**lightGray**);**

joinGame**.**setBackground**(**Color**.**lightGray**);**

friendList**.**setBackground**(**Color**.**lightGray**);**

leaderboards**.**setBackground**(**Color**.**lightGray**);**

chatRoom**.**setBackground**(**Color**.**lightGray**);**

frame**.**setVisible**(true);**

frame**.**setDefaultCloseOperation**(**WindowConstants**.**DO\_NOTHING\_ON\_CLOSE**);**

frame**.**addWindowListener**(new** java**.**awt**.**event**.**WindowAdapter**()** **{**

@Override

public void windowClosing**(**java**.**awt**.**event**.**WindowEvent windowEvent**)** **{**

int check **=** JOptionPane**.**showConfirmDialog**(null,** "Are you sure you want to exit?"**,** "Confirm Exit"**,** JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**check **==** JOptionPane**.**YES\_OPTION**)** **{**

System**.**exit**(**0**);**

**}**

**}**

**});**

**}**

public void setVisible**(**boolean b**)** **{**

frame**.**setVisible**(**b**);**

**}**

public void chatClosed**()** **{**

mmc**.**inChat **=** **false;**

**}**

**}**

# MainMenuFactoryUI

package UI**;**

**import** Player**.**User**;**

public class MainMenuUIFactory **extends** UIFactory **{**

User currentUser**;**

public MainMenuUIFactory**(**User cu**)** **{**

currentUser **=** cu**;**

**}**

public void create**()** **{**

MainMenuUI gui **=** **new** MainMenuUI**(**currentUser**);**

**}**

**}**

# UIFactory

package UI**;**

**import** java**.**io**.**IOException**;**

public abstract class UIFactory **{**

public abstract void create**()** **throws** IOException**;**

**}**

# CS4125User Package – CS4125User

package cs4125User**;**

**import** UI**.**UIFactory**;**

**import** UI**.**LogInUIFactory**;**

**import** java**.**io**.**IOException**;**

public class CS4125User **{**

public static void main**(**String**[]** args**)** **{**

UIFactory uif **=** **new** LogInUIFactory**();**

**try** **{**

uif**.**create**();**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"I/O Error\n" **+** e**);**

**}**

**}**

**}**

# Added Value

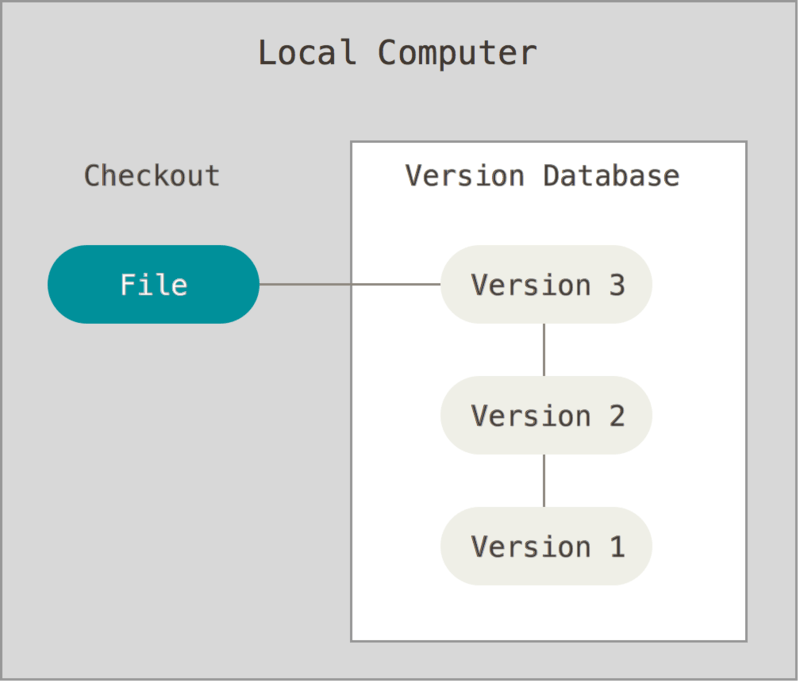
# Git

For our version control system, we looked at various version control systems that we could implement. We decided on Git, a distributed version control system.

# Local Version Control Systems

This is the simplest approach to version control, as it may be achieved copying files into a (time-stamped) sub directory. It is a common approach yet is fails to suit our needs as it is very error prone and does not support concurrent development well. Overwriting files and lack of concurrent development support may lead to loss of data.

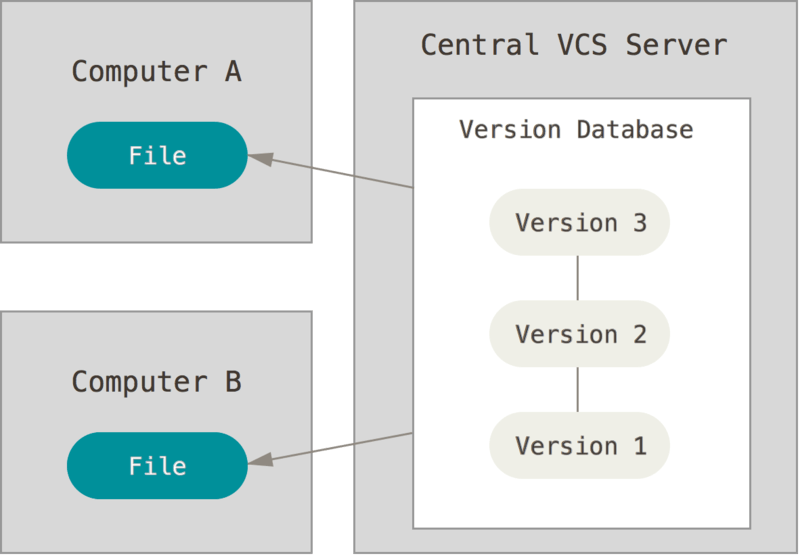
Developed Local VCSs with a simple database such as RCS would neither suit our needs as there still would be no support for concurrent development.



# Centralised Version Control Systems

The next step to consider is a centralised Version Control System (CVCS) as it supports collaboration with other developers. These systems, such as Subversion or Perforce, utilise a single server that contains all versions of the files. It is far easier to administer a CVCS than it is to keep track of local databases on every client.

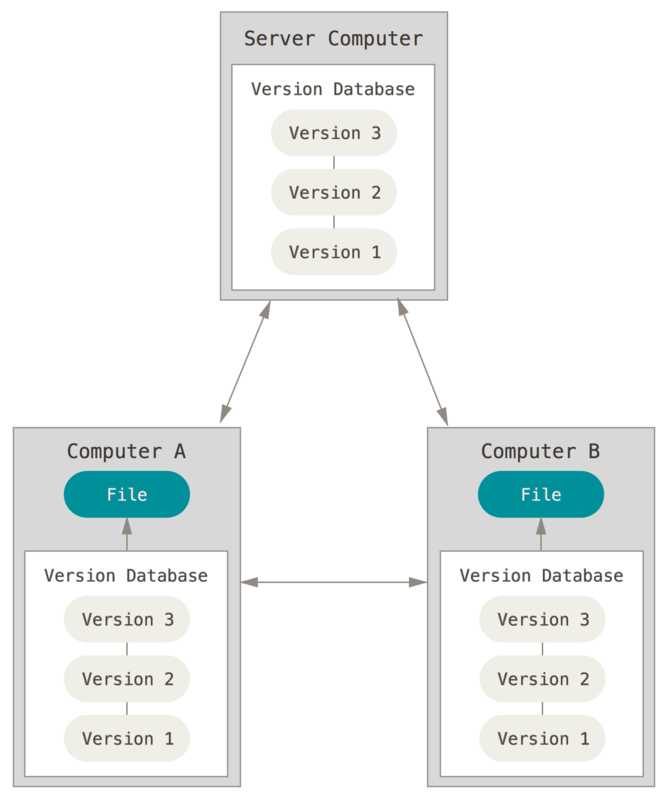
However, the downside comes as there is a single point of failure with a centralised server. If the server goes down, nobody can pull any data or make any changes to the files. In the case of hardware failure on the server, data may be lost if sufficient backups haven’t been made.



# Distributed Version Control Systems

This leads on to Distributed Version Control Systems (DVCSs). Pertaining to the reason why we chose a DVCS is that a popular DVCS is Git, in which some group members were already familiar with.

Like a CVCS, a DVCS highly supports collaboration with other developers, even as far as having several remote repositories to collaborate with different groups of people on the one project. But in contrast to a CVCS, when a client wants to make changes to some files, they mirror the full repository containing the version database. Thus if any server dies, it can be restored by any client that was collaborating with it.



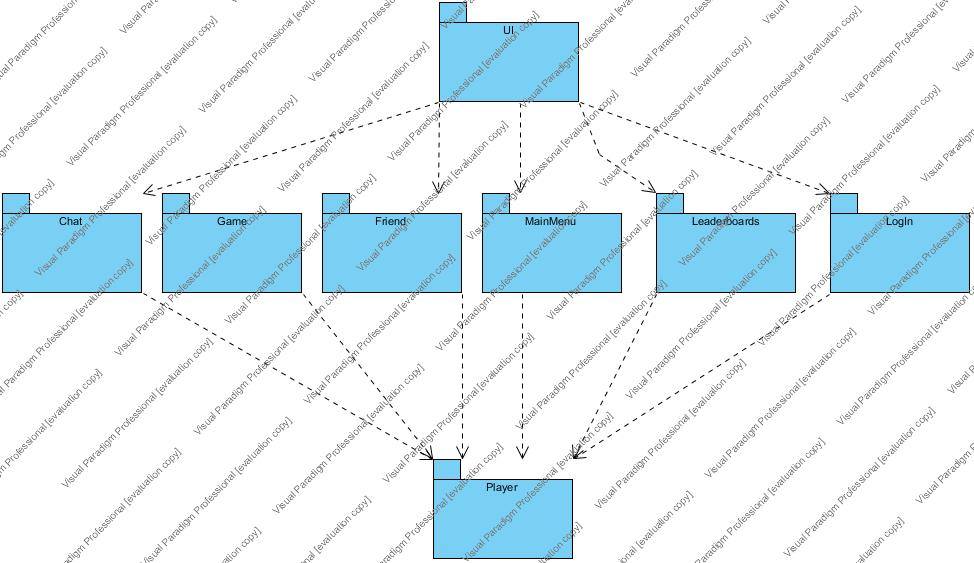
We chose to use GitHub to support online project hosting using Git, as group members were familiar with GitHub already and it best suits our collaboration and version control needs.

# Concurrency

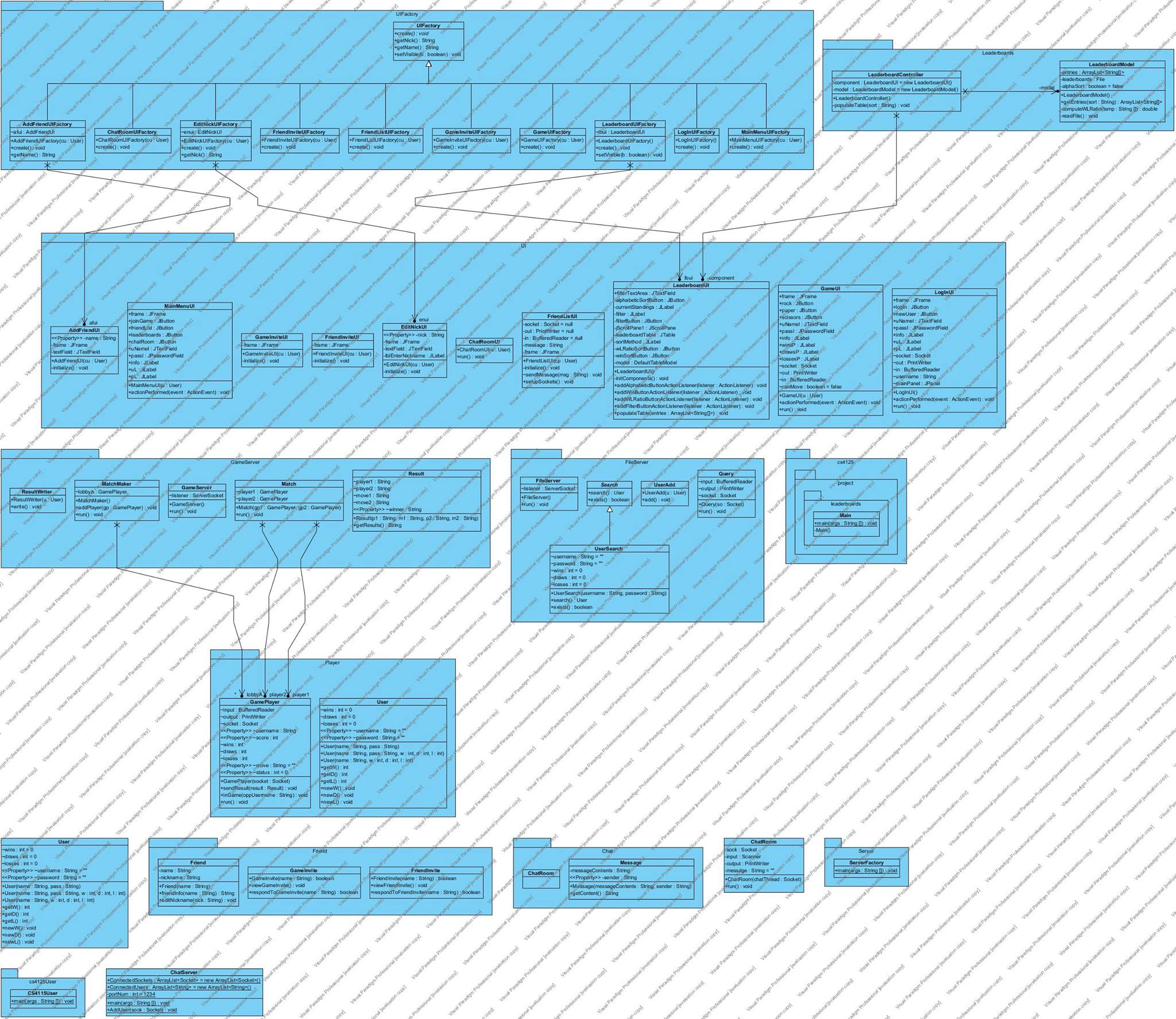
Concurrency is the key component that makes the project function. All four of the main functions of the client (game, chat, friend list and leaderboard) utilize threads to obtain data from the server as soon as possible, and each server makes use of threads to obtain real-time data and process it.   
  
The servers themselves run threads in order to be always able to accept new connections. On the connection of a new client to the server, a new thread is initialized to either handle their requests or send responses. Threads are especially vital in the ChatServer and the GameServer, which involve communication with another live player. As soon as both players send in their moves to the server, a result is generated and is sent back to each player. Multiple games can be running at once thanks to threads working in parallel with the rest of the program. In the ChatServer, once a new member joins, all current members in the chat get a notification and a new entry in the “Online Users” list in the UI. When one user sends a message, the server distributes it to all online users.  
  
When a thread is no longer needed, it is interrupted. Each thread runs on a “while(!Thread.interrupted())” loop, meaning that once it’s interrupted, it leaves the loop and it stops running. This is crucial to prevent server overload, as multiple open threads becomes very resource-demanding.

# Design and Architecture Recovery

# Architectural diagram

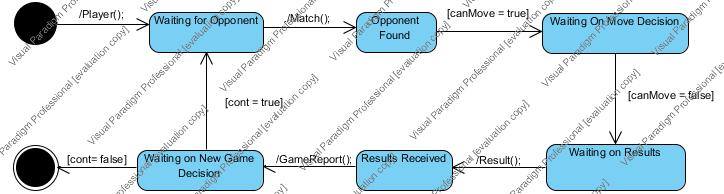


# Class Diagram



# State Chart

State Chart for Game.



# Description of Patterns and Added Value

# Discussion

Numerous design patterns are implemented throughout the project. The client system utilizes a combination of the observer and factory patterns. The UI system is the main example of the factory design pattern. Each UIFactory, excluding the UIFactory itself, creates an instance of the UI which it is dedicated to initializing. For example, the LogInUIFactory initializes a LogInUI with no arguments as it’s the first menu that the client generates. This makes the system easier to expand in the future as it does not lead to searching through code to make necessary changes.  
  
There are two main examples of the observer design pattern. The observer design pattern is a behavioural design pattern which defines a one-to-one dependency between, meaning when an object changes state, all of its dependants are notified and automatically updated. The first example also relates to the UIFactory. Classes such as the MainMenuController declare the abstract class UIFactory globally, and when a button is pressed to open up a new menu, MainMenuController initializes the UIFactory using one of its subclasses, for example, the ChatRoomUI. This allows the client to generate a new frame and update the interface with it. We had the option of swapping the panels of a single frame each time a new UI is initialized, but we wanted to have the option of keeping two frames open.

The second example of the observer design pattern is seen utilized by the GameModel class. As it receives word from the server regarding the result of a game, the GameModel takes its GameReport abstract class that it declared globally and initializes it with either the Win, Draw or Loss subclasses. The report() method is called and the initialized subclass creates a JOptionPane with output corresponding to the result of the match.

# Critique

We used the noun verification technique to generate a series of classes which appeared to be appropriate for our project. However, during implementation, it became clear that some classes were more abstract than we originally imagined, or served a purpose that another class already accomplished. We also realized that we did not consider classes which would prove to be vital to the project’s success.  
  
Choosing to adopt the Agile Software Lifecycle model meant that we had plenty of flexibility when it came to reacting to changes that had to be made to the project. This would prove to be a crucial contribution to the successes of the project.  
  
We also overestimated the importance of particular classes and packages. The Player package ended up only having one class contained in it; the User class, and we dedicated time and resources to expanding it only for it to be unnecessary. We should have been more careful while defining the role and importance of each class throughout the project.  
  
The majority of user interfaces (UI’s) were originally designed in a single class, but were later converted to Model-View-Controller (MVC) during the second iteration. This was mainly due to the team’s inexperience designing UI’s using MVC, and while they were implemented successfully, it was a slow process and the time spent implementing it could have been spent on implementing a new use case if each UI was originally built using MVC. As all team members have gained valuable experience building using MVC through this project, it will be something that can be considered for the future.

Design patterns were also not considered for the project until late in the iteration cycle. Many design patterns were considered but only the observer and factory design patterns were deemed appropriate for the project. Implementing the decorator design pattern required overhauling the infrastructure of the Match and Player class for minimal benefit. The implemented design patterns did help with tidying up the pre-existing code.  
  
There were a number of problems with the original design of the client:

1: The original plan involved a single server that handled all the client’s requests. Early on during implementation, it became apparent that three individual servers were required to handle file-related queries, the game/matchmaking and the chat.

2: Being able to view friend availability would prove to be very time-consuming to implement due to the server architecture change mentioned in 1. It would involve connecting to the server every time the user wished to quit the client and maintaining a separate text file listing all online users.

3: Inviting a user to a game, along with creating a private game, also proved to be very expensive to implement time-wise due to the problems named above. It would require both users to be online, and it would also require an overhaul of the MatchMaker class or a private game server of its own.

4: The same problems in 3 also applied to some of the use cases intended for the chat system, namely chatting with a friend or another particular user.

5: The Player package was required by both the server-side and the client-side, and there was no simple way to remedy this other than including an instance of the package in separate projects, one for the client and one for the server.

# References

# Websites and learning

* Git (About Version control) - <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>
* GitHub Version Control - https://github.com/
* CS4125: Serenity Gaming – a sample project from Sem 1 2014 – 2015
* CS4125: Awesome Gaming – a sample project from Sem 1 2015 – 2016
* Astah Modelling Workbench - <http://astah.net/>
* Visual Paradigm Modelling Workbench - https://www.visual-paradigm.com/

# Images

* Rilakkuma cover page image - <https://www.facebook.com/>, <http://www.san-x.jp/>
* Version control images - <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>
* Agility Software Lifecycle model - CS4125: Serenity Gaming