Squire: A Collaborative Software Development Tool

jank6275, mora5651, boss2849, bolt1003, gall7417, brec9824, snev7821, mars2681

February 20, 2016

Contents

Chapter 1

Introduction

1.1 Program Premise

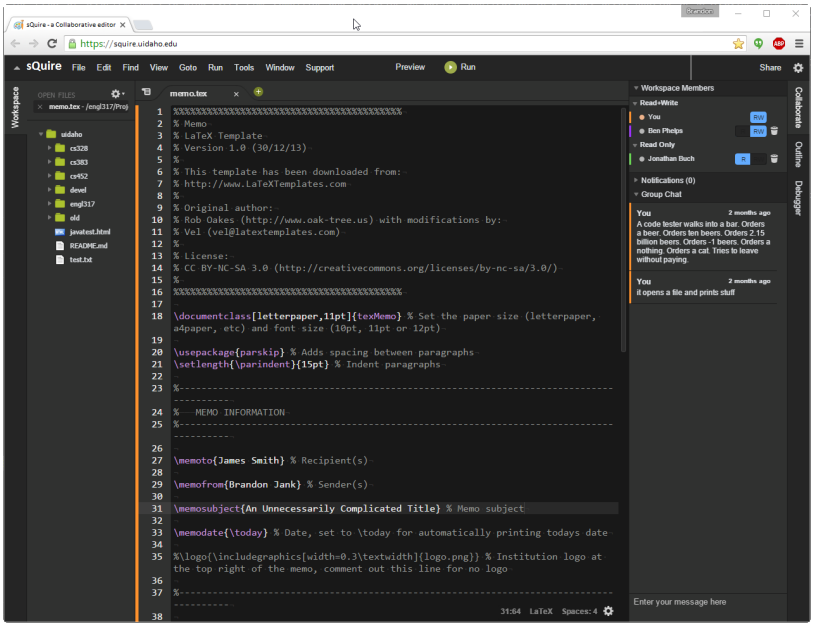

Figure 1.1: "Squire is a web-based collaborative software development environment with a project development center. Squire will allow multiple users to edit files and communicate in real time. Projects can be "stubbed" out by a user and then other users can join and/or vote to support for their favorite projects. After a certain amount of support, planning, and documentation is reached for a project, the project becomes a fully fleged project and then community development can start. Think "kickstarter for code" where people pledge their help with the project and not just financial support."

Chapter 2

Class Diagrams

2.1 Overview (brec9824, jank6275)

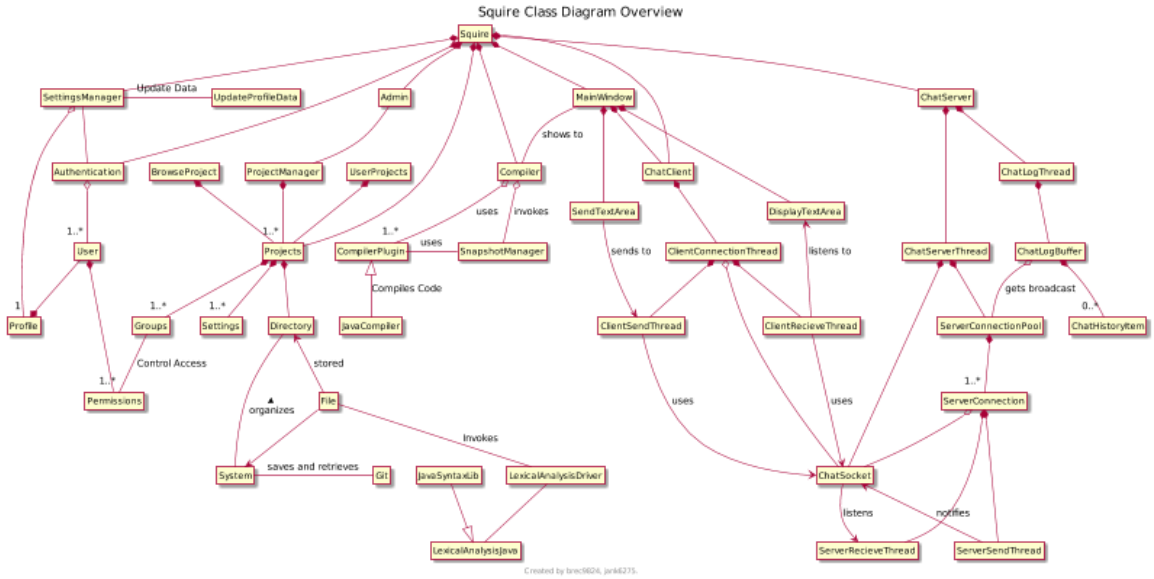

Figure 2.1: Class overview of Squire displaying the connections between each subsystem and their classes.

2.2 Authentication (mora5651)

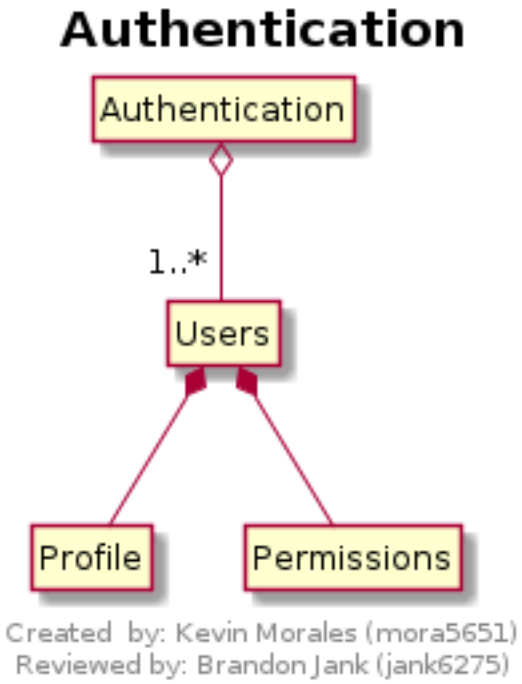

Figure 2.2: Authentication is a security precaution in sQuire that allows the system to verify a user. The authentication process will implement the traditional user/password.

2.3 Project Management (bolt1003)

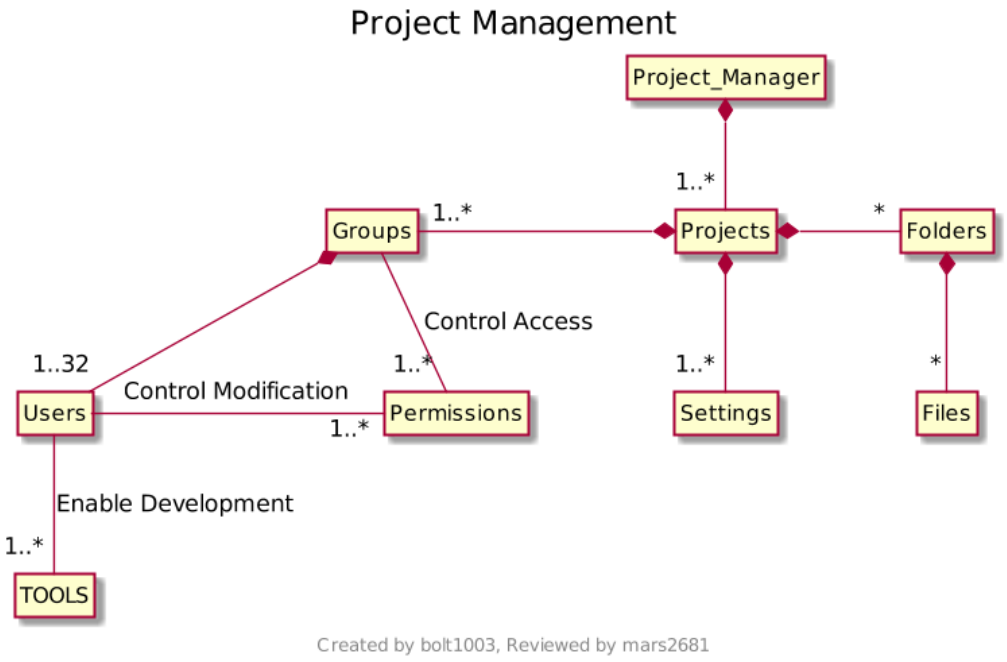

Figure 2.3: Project management allows for the group to delicate permissions and create projects

2.4 Project Ideas (mars2681)

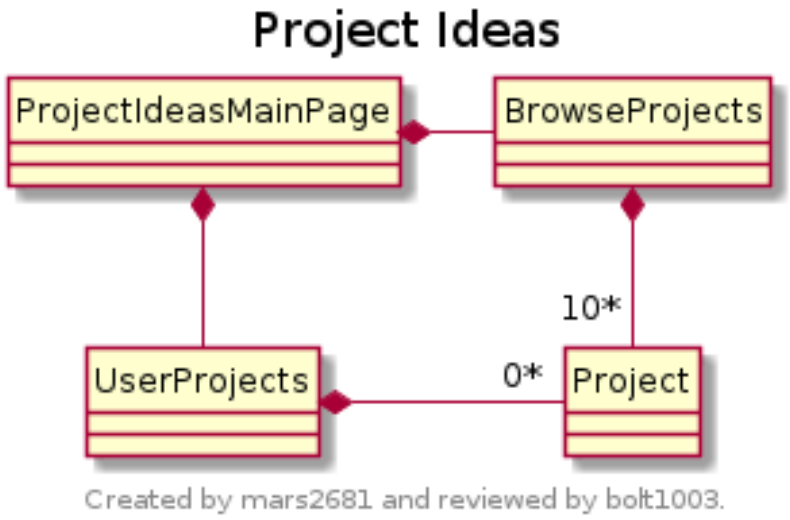

Figure 2.4: The Project Ideas page allows the user to browse other user's projects ideas, as well as create and customize their own project ideas. It starts at the main page. From here, the user can select to view their own projects, and from there create, delete, or edit their ideas. Also at the main page, users can view other projects. Here will be categories of projects, the most popular projects, and new projects. Through browsing these pages, the user can see the description of projects, the amount of support, likes, dislikes, and they can follow and pledge their assistance to the project.

2.5 Settings - Preferences & Profile (brec9824)

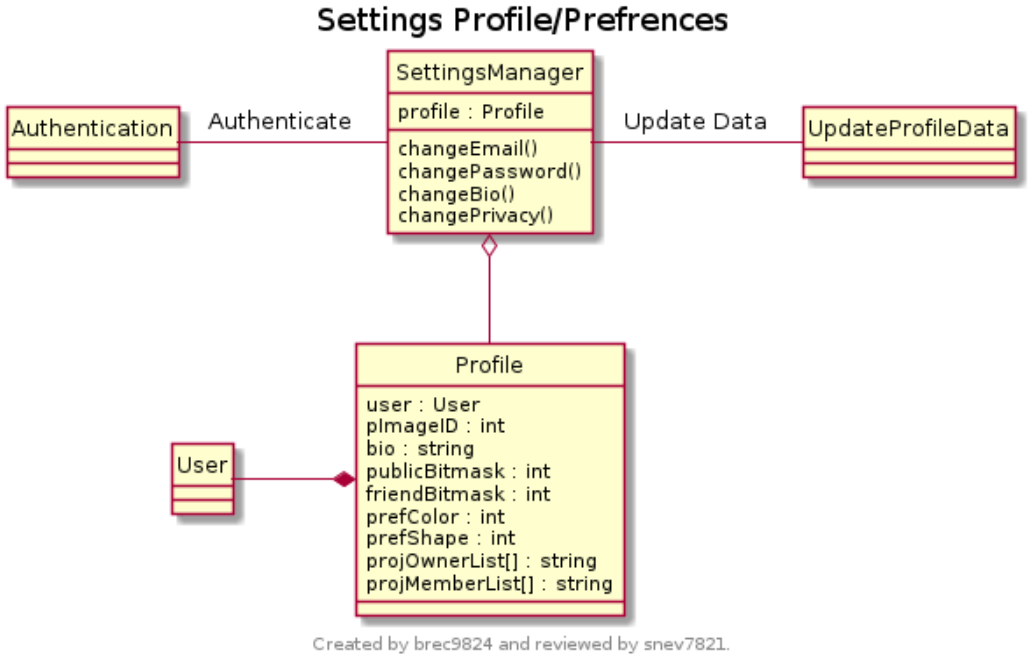

Figure 2.5: Settings Profile/Prefrences allows for profile viewing and management while maintaining speed with the use of push updates. SettingsManager uses Authentication to verify valid input and to authentic the users data. While SettingsManager uses UpdateUser-Profile to push the users data that needs to updated to the server.

2.6 Compiler (boss2849)

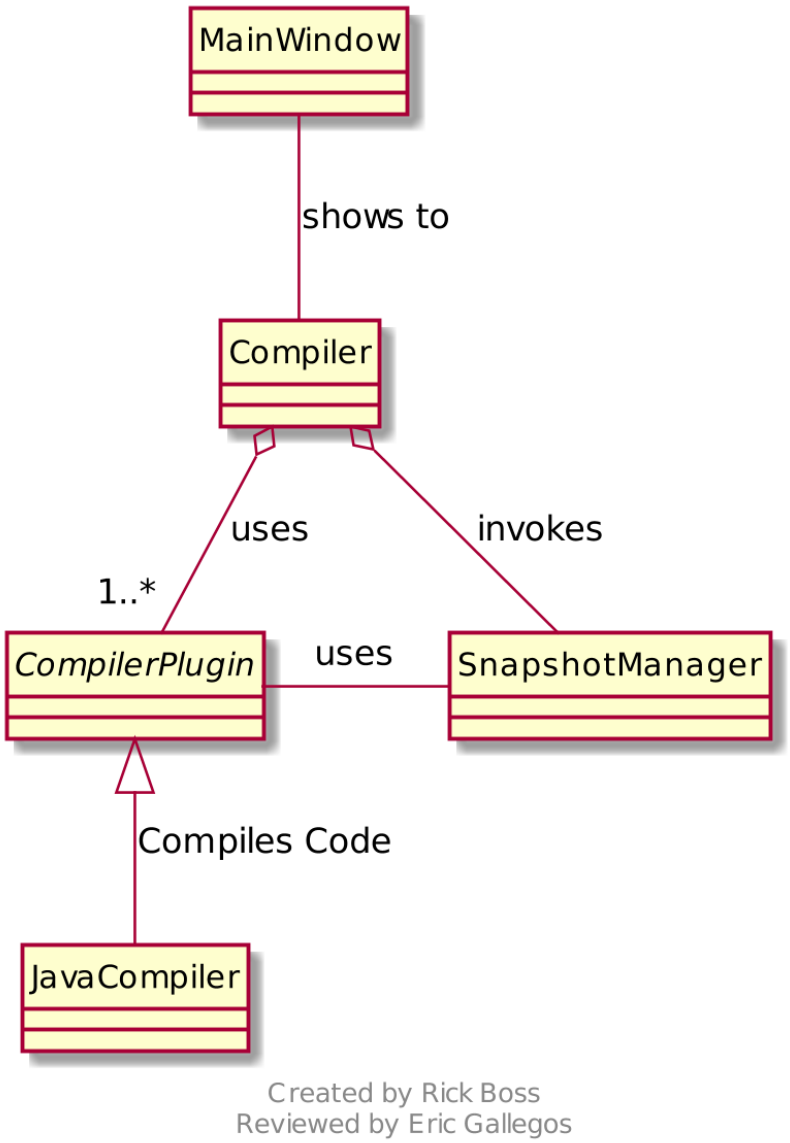

Figure 2.6: The Compiler is invoked from the main window. From here, the Compiler will select the appropriate CompilerPlugin determined by the configured compiler for the project. The compiler also invokes the SnapshotManager, which stores the current state of the code to be used during compilation. In this simple example, only a JavaCompiler plugin is present, but there can be more than one plugins available in the future. The JavaCompiler plugin retrieves the code snapshot from the SnapshotManager and compiles the code.

2.7 Lexer (gall7417)

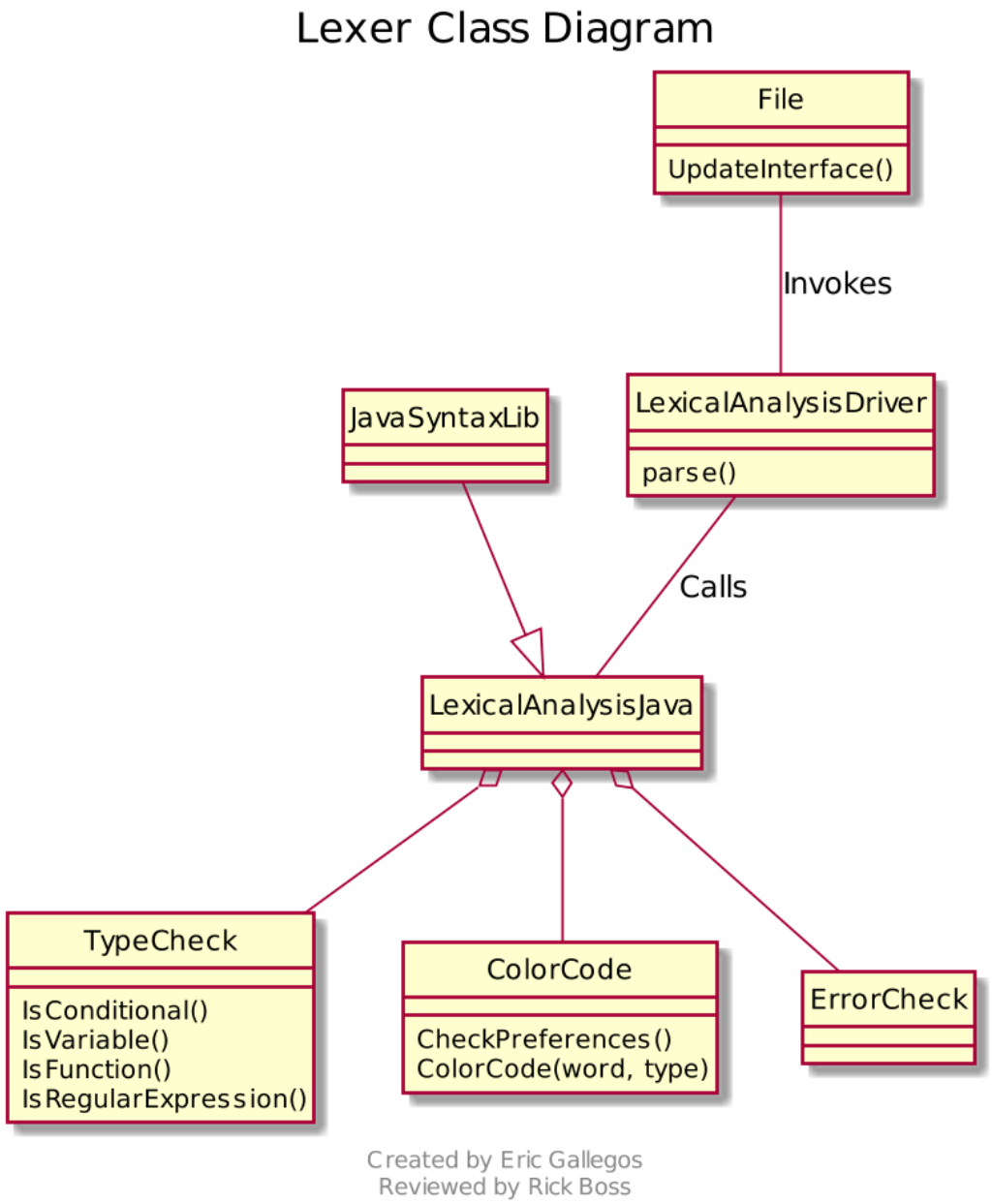

Figure 2.7: The File class regularly invokes the Lexical Analysis class to give feedback to users code input. The Lexical Analysis driver calls the corresponding Lexical Analysis class for the language used (Java). This language specific class checks for errors using the ErrorCheck class, searches for word types using the TypeCheck class, and will assign the various words colors to reflect the word types using the ColorCode class.

2.8 Communication (jank6275)

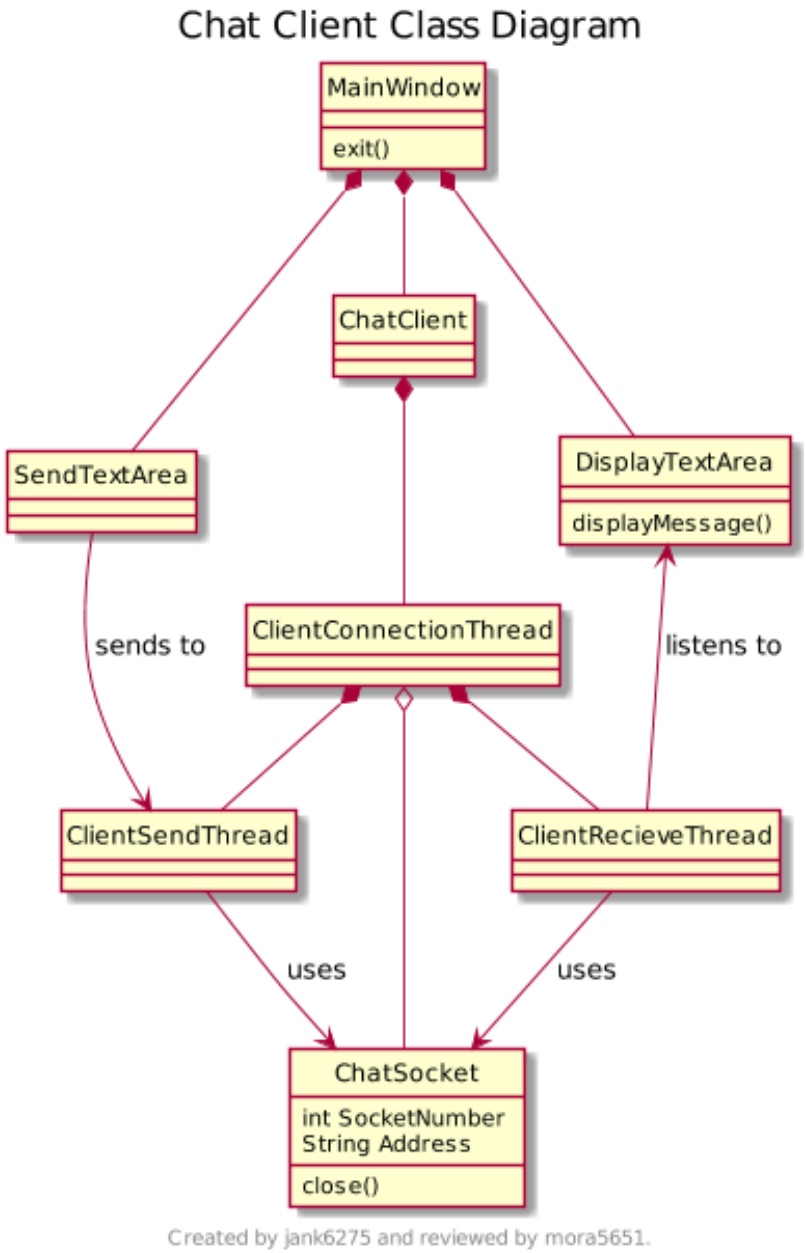

Figure 2.8: The ChatClient class will handle text communication in conjuction with the ChatServer class. The Main Window will be home to the ChatClient. The ChatClinet will consist of client send/recieve threads to handle user input/output in the Main Window via the ChatSocket.

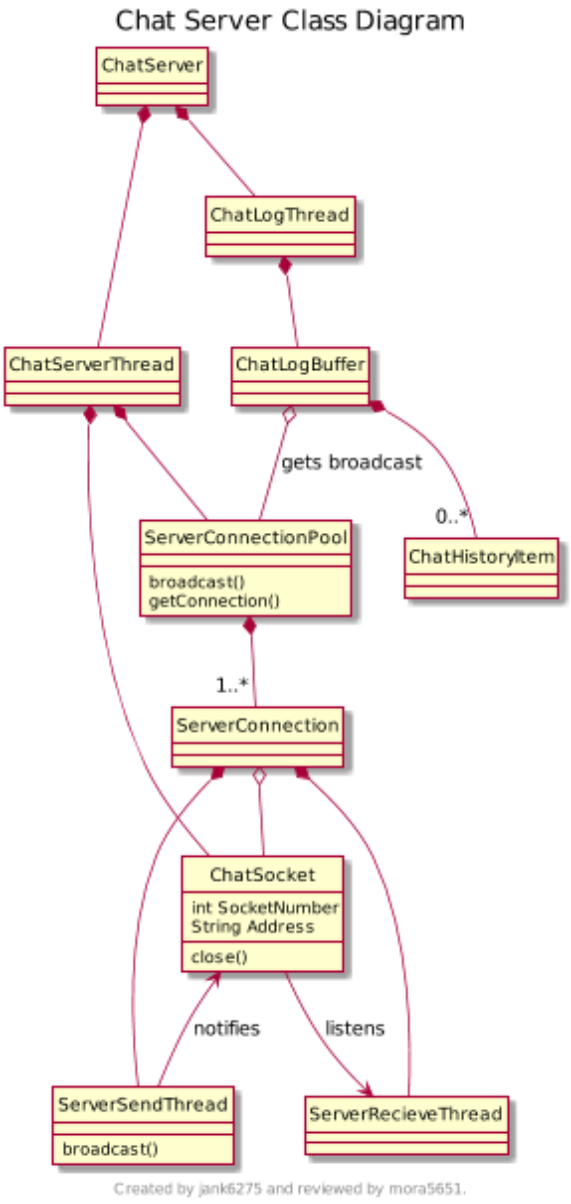

Figure 2.9: The ChatServer class will handle text communication between ChatClient(s). The ChatLogThead records any messages broadcast by chat clients in the ServerConnection-Pool as a ChatHistoryItem. Each ServerConnection consists of a send and recieve thread that utilize the ChatSocket to broadcast messages.

2.9 Project File Editor (snev7821)

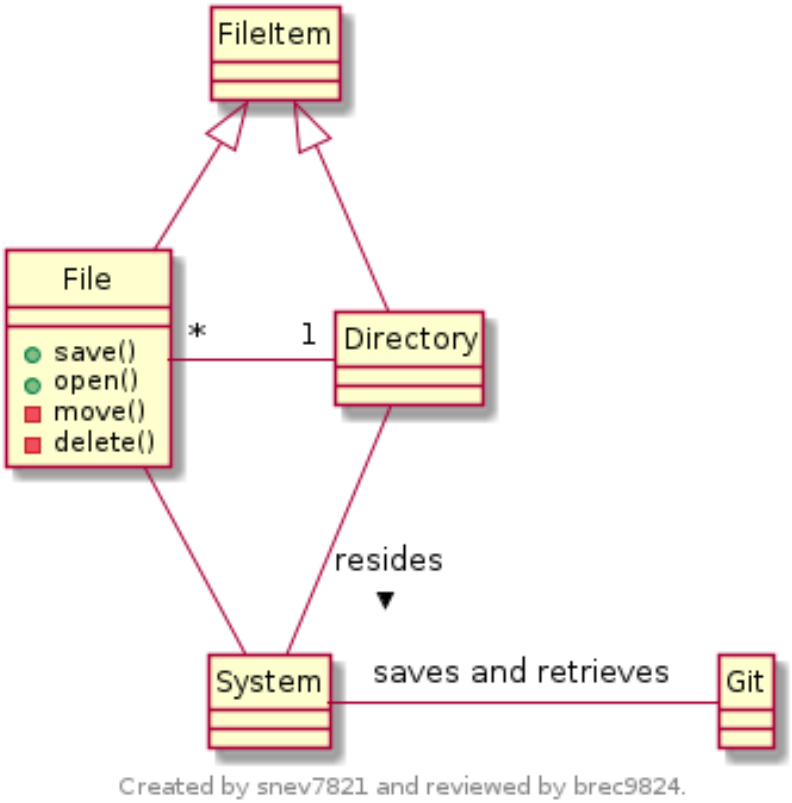

Figure 2.10: The project File Editor is simply the file system for Squire. It provides basic read/write permission for any user related to a project. Only admins of a project may move project files, delete old files, and create new files.