Rubikan Design

by

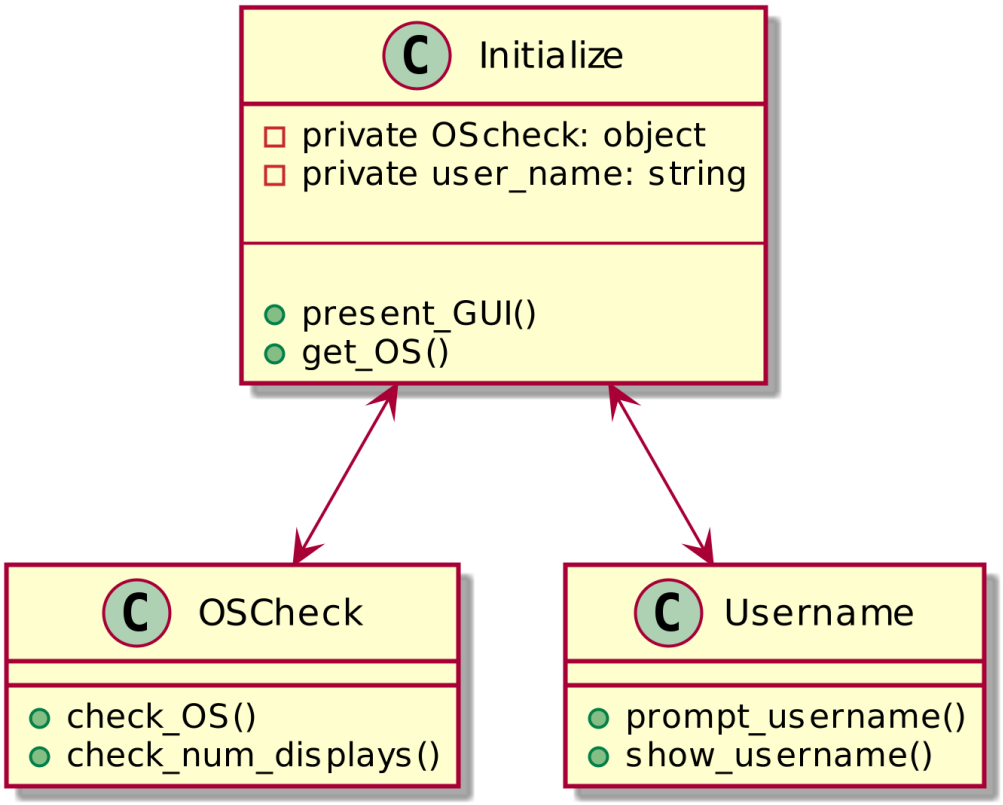
Christerpher Hunter

Nova Southeaster University

July 25, 2021

Design

Startup



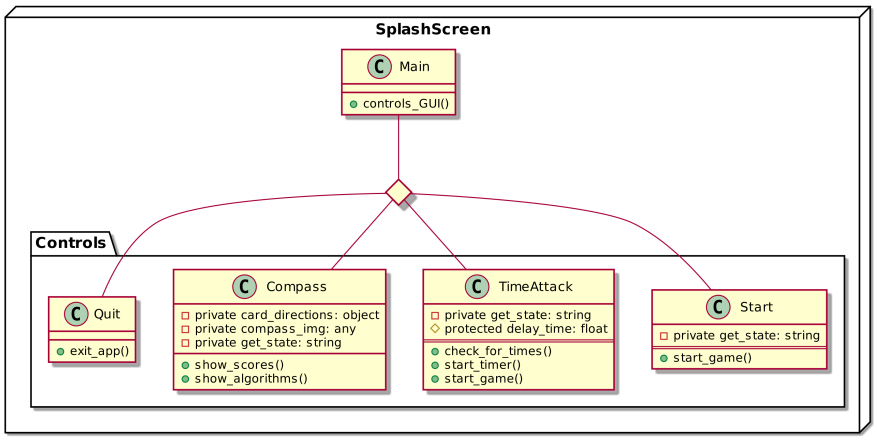
The purpose of the class Initialize is to launch the GUI window, get the username from the user (similar to an Arcade), and check the operating system. The GUI window will present the Standby State presented in the next diagram. The username function will prompt the user for a name to use when commiting scores. The operating system check will get the name of the operating system and which distribution, if necessary.

The intent of having a module to display the GUI before the module that contains the buttons and actions is due to the manner of implementation of the GUI operation. The idea here is to set up the frame and backdrop for the application first and then load the features afterward. This will allow future enhancement of the application to be simpler. The Standby State will reside in waiting and then immediately load into the Initialization class without delay. This is due to a fluid and manipulative design.

The class Username is used in several places throughout the application in order to keep track of scores. Each completed attempt will be coupled with a time and a username. The username in, every instance, will be provided by class Username. The class Username will prompt the user at the beginning of each launch of the application. Another user, at any time during the Stanby State, can add a new username. Upon the completion of a game, if there are two or more usernames, a confirmation prompt with the time and the username list will be shown. The present user will need to choose, or optionally, enter a username; this is required to enter a winning time into the database.

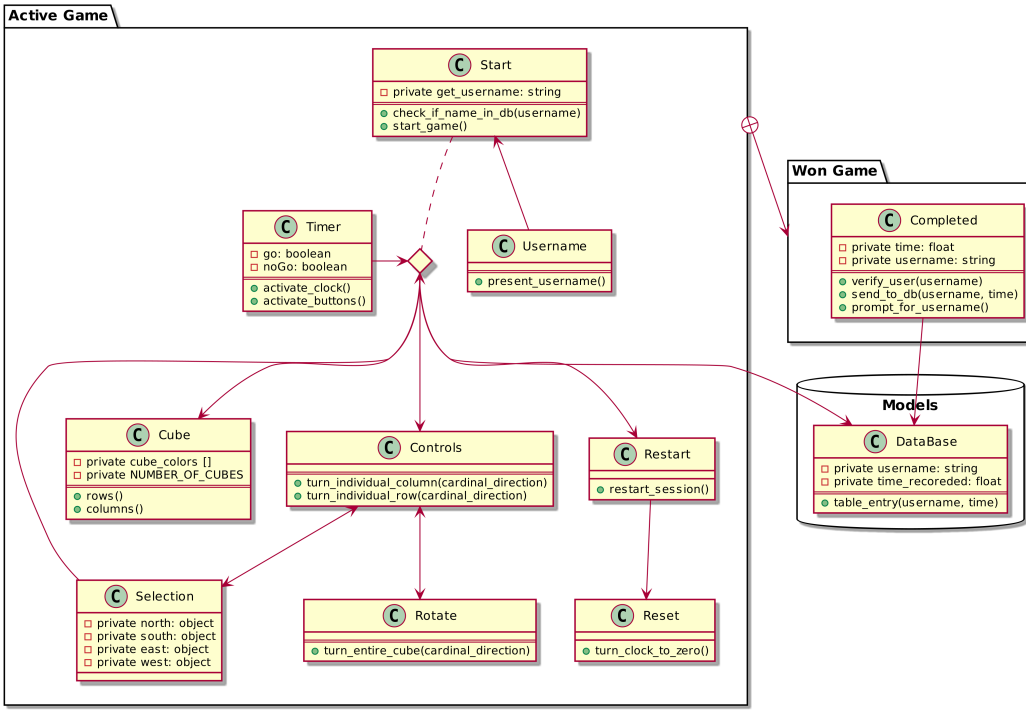
The class OSCheck is intended to get the name and version of the operating system from which the application launches. Should the operating system come back as a Linux operating system the class OSCheck will then attain the specfic distribution. The purpose of checking the operating is two fold. Firstly, the class MultiMonitor will communicate to the class OSCheck to obtain the operating system; this is needed for the class MultiMonitor to function as expected. Secondly, to appropriately display the application and its popups the manner in which frames are displayed must be ascertained and is directly related to the operating system the application launches from.

Standby State



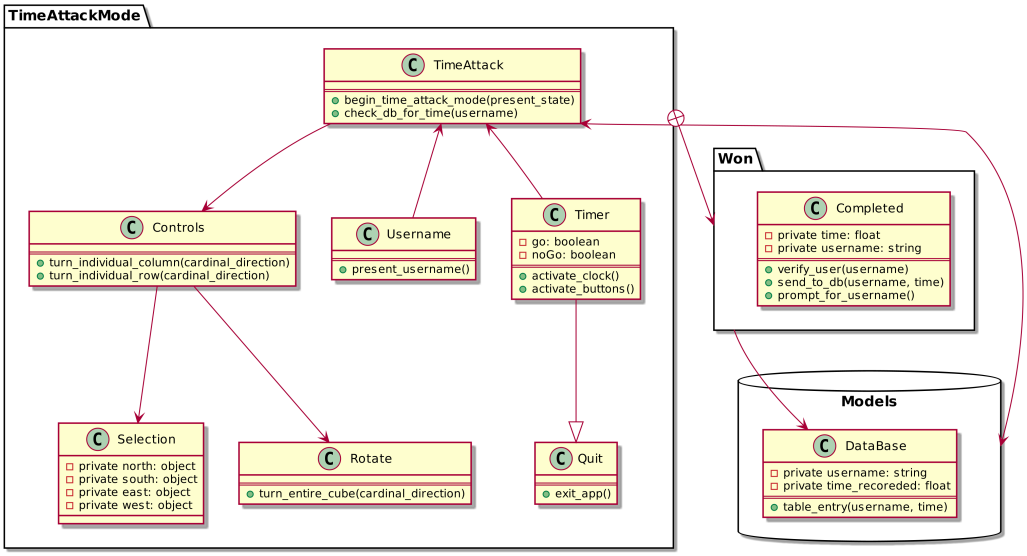
The splash screen serves as a template for which all non-timed activities are available and accomplishable. The class Main has one method, the controls_ GUI, that communicates with the Controls module. In the Controls module there exists four control classes. The class Compass is a clickable button in the top left of the window that pops up another window displaying the previous times and username that reside in the database. The class TimeAttack is similar to the start button in that it deactivates other buttons and moves the application from the Standby State to the Time-Attack State. The class Start will initiate a normal game, deactivate other buttons, and move the application from the Standby State to the Active State. The class Quit is design to exit the application at anytime during any state.

Active State



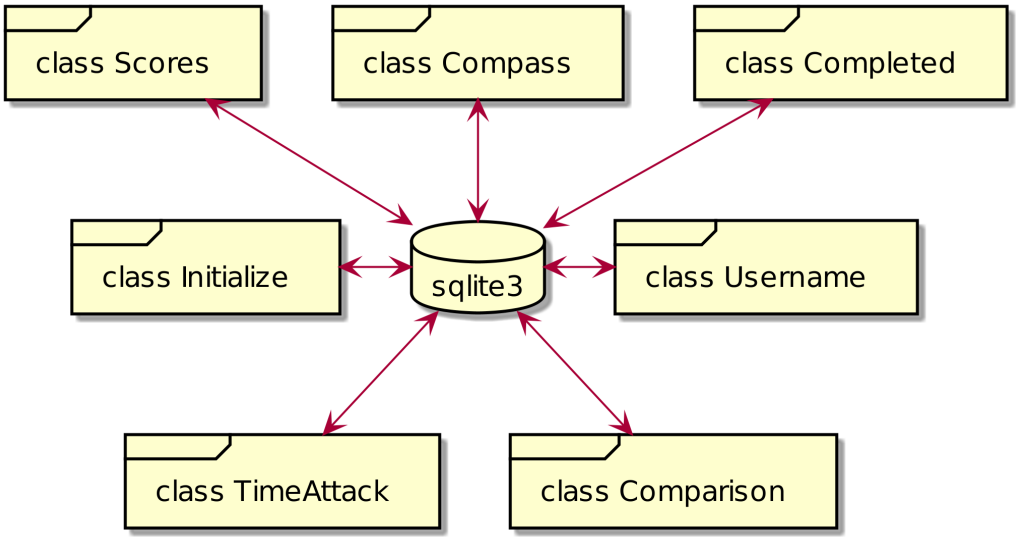
The class Start is the main way to initiate the active state. When the active state is initiated the class Start will prompt the class Username for the username of the present session. The class Start will then check the database for a prior entry of the passed username. If no username is entered a prompt for a username will be presented upon a succesful solving of the puzzle via the class Completed. Once the class Start requirements are satisfied the class Timer initiates the clock to begins counting up from zero. Upon initiation of the clock the class Controls is activated and the user will be able to manipulate the cube via the class Cube. The class Cube contains the rows and columns that are to be manipulated. The class Controls take an object direction via the class Selection when a user selects a column or a row on the graphical cube. The class Rotate is solely responsible for rotating the entire cube. The class Restart will represent a graphical button that is intended to reset the cube. The class Resart automatically prompts the class Reset and will reset the clock. If puzzle is solved, after any allowed amount of time, the class Completed will take in the time and username, with the appropriate username checks, and send that information to the class Database.

Time-Attack State



The time-attack can only be activated from the standby state. There are only two differences between the time-attack state and the active state. First, the class TimeAttack checks the class Database for a time, given a username. If there is no time associated with the username, from past attempts during the active state, the time-attack state cannot be entered. Second, the clock counts down instead of up. The timer starts at the shortest previous completion time associated with the given username. The purpose of this mode is to best the users previous best time. Should the time-attack mode be completed successfully, the new time will be calculated and sent to the class DataBase.

Data-centered Architecture



The class DataBase pictured above is shown with all the possible classes that will interact with the database. The details of the database and its interactions throughout the application are emphasized where needed.