Draw it or Lose it

CS 230 Project Software Design Template

Version 3.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	7.13.20	John Elder	Original SDT
2.0	8.2.20	John Elder	Updated to include Evaluation
3.0	8.16.20	John Elder	Updated to include Recommendations

Executive Summary

The goal is to create a game in which teams can guess what is being drawn. Proposal for what to implement further is as follows:

- Games should have one or more teams
    - Must have a way to tie a team to a specific game
    - Allow the game to start with only one team
- A team must consist of more than one player
    - Do not allow a team to be utilized until it has at least 2 players
- Game and team names must be unique
    - Check to make sure that there are not existing instances of the desired team name (Singleton)
- Only one instance of a game can exist in memory at a time (Singleton)
    - Prevent additional games from being created

Design Constraints

- Storing the library of drawings is needed for this game.
    - To do so, storing the library of them as individual instructions utilizing the Java 2D API on the server side of the application and loading the method for drawing them onto the clients machine once a drawing is selected would allow for accurate timing of the drawing to develop.
- The client-side program should be written in Javascript. It has the most compatibility with internet browsers.
- The server would be best implemented in Java, because it could be run on multiple operating systems.
- The method of communication between them is aided by the JavaScript Engine Rhino

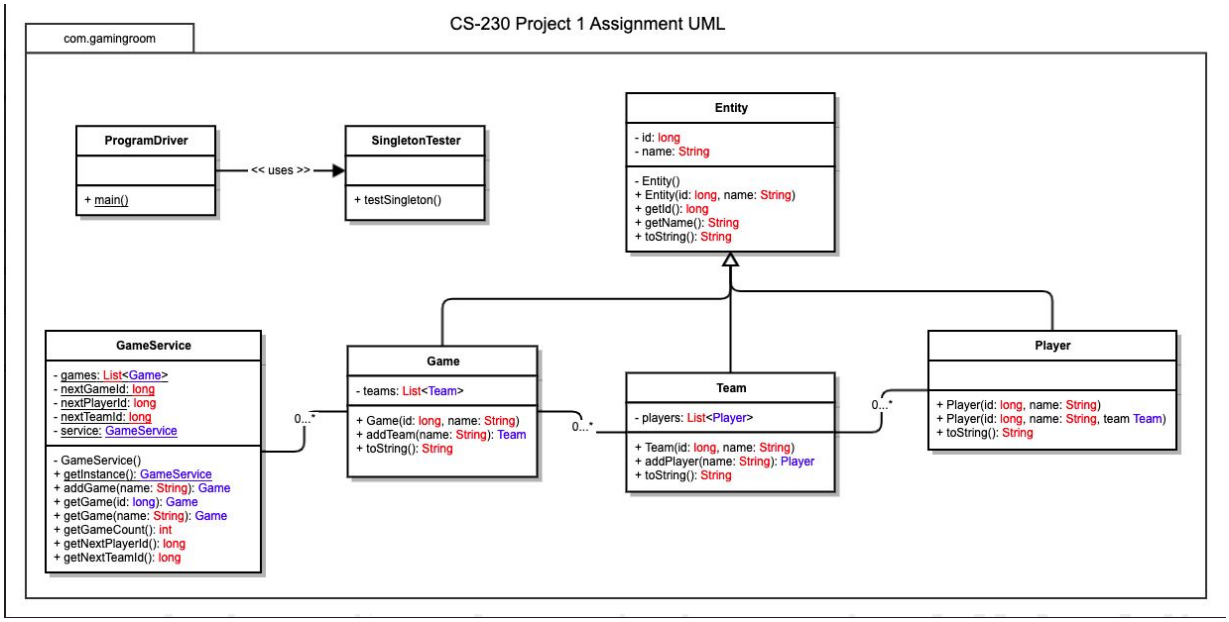
System Architecture View

Domain Model

- A class called **GameService** is utilized to maintain a single instance of any **Game**s, **Team**s, or **Player**s
- The class **Entity** will be created to establish
    - the attributes of *name* & *id*
    - encapsulation through also defining *getter* & *setter* methods for both
- The class **Game** inherits **Entity**, and is utilized to construct unique games that are being played
- The class **Team** inherits **Entity**, and is utilized to construct unique teams to play in games
- The class **Player** inherits **Entity**, and is utilized to construct unique players to make up teams

This setup is implemented to maintain unique instances of each of these classes. This way, we will avoid strange behavior that could result from having the same team in multiple games, same player on multiple teams, etc.

UML diagram:


CS-230 Project 1 Assignment UML
com.gamingroom
ProgramDriver
+ main()
<< uses >>
SingletonTester
+ testSingleton()
Entity
- id: long
- name: String
- Entity()
+ Entity(id: long, name: String)
+ getId(): long
+ getName(): String
+ toString(): String
GameService
- games: List<Game>
- nextGameId: long
- nextPlayerId: long
- nextTeamId: long
- service: GameService
- GameService()
+ getInstance(): GameService
+ addGame(name: String): Game
+ getGame(id: long): Game
+ getGame(name: String): Game
+ getGameCount(): int
+ getNextPlayerId(): long
+ getNextTeamId(): long
0...*
Game
- teams: List<Team>
+ Game(id: long, name: String)
+ addTeam(name: String): Team
+ toString(): String
0...*
Team
- players: List<Player>
+ Team(id: long, name: String)
+ addPlayer(name: String): Player
+ toString(): String
0...*
Player
+ Player(id: long, name: String)
+ Player(id: long, name: String, team Team)
+ toString(): String

Evaluation

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	Apple's Mac OS is not necessarily geared towards server-side use but does have good support from its creators and has the simplicity of only being run on specific hardware. The licensing costs associated with running a server on a Mac is relatively low. The macOS Server application can be found on the Mac app store and costs $19.99, but the hardware to run the software will be costly.	Linux is an open source platform, and that leads to lower costs of utilization for our server-side hosting. However, it has a lack of support from its creators due to the variability of distributions. Licensing would be nearly free, depending on if we decide to utilize any software that does cost money. Linux does have server based deployment options like Apache.	Windows Server has a deployment service creatively named "Windows Deployment Services" that could be utilized. On top of that, most of our team is likely to understand Windows due to its popularity. However, windows has the increased cost of purchasing a license for running the operating system. For example, a license of Windows Server 2019 Standard edition is $972.	Hosting a server on a mobile device is generally not a good idea, as it needs to be connected constantly.
Client Side	Because we are utilizing a uniform HTTP interface on the server side, the variation that we must deal with between Mac, Windows and Linux is lessened greatly. Typical Mac users on the client side will be using either Safari or Chrome, and the application should be targeted towards these two.	Because we are utilizing a uniform HTTP interface on the server side, the variation that we must deal with between Mac, Windows and Linux is lessened greatly. Typical Linux users could be using a plethora of applications, but Chrome and Firefox will be the most likely. Targeting towards	Because we are utilizing a uniform HTTP interface on the server side, the variation that we must deal with between Mac, Windows and Linux is lessened greatly. Windows users may be using IE or Edge, but likely Chrome or Firefox. Again, targeting these is preferable, with some attention	Mobile users will be downloading an app itself. For this we will need a few testing devices from both Android and iOS to make sure that the deployment is functioning properly. On top of that, the teams we have coding these two may have very different skillsets and proficiencies compared to those dealing with the

		those two and doing some troubleshooting with Opera, Chromium and Vivaldi would be beneficial.	given to IE and Edge.	web-based game. This will cost more and slow down production time
Development Tools	Developing the web-based game for deployment on a Mac machine, I would recommend utilizing either PhysicsJS or Phaser. If using PhysicsJS, it should be run as an AMD Module with RequireJS. If using Phaser, an editor would be required and I would recommend either Sublime Text or Atom. Both are cross platform, but there is the licensing cost of Sublime Text for $70, while Atom is opensource. These may impact the development team if they're not already familiar with these tools.	Developing for deployment on a Linux machine, I would again recommend the same PhysicJS and Phaser. Again, PhysicsJS requires the use of RequireJS and Phaser requires an editor to implement. These options could force the development team to have to learn more about these tools if they are not familiar.	Developing for deployment on a Windows machine, I would again recommend the same PhysicJS and Phaser. Again, PhysicsJS requires the use of RequireJS and Phaser requires an editor to implement. These options could force the development team to have to learn more about these tools if they are not familiar.	Developing for the deployment on mobile platforms will require the SDK's that are available for both Android and iOS, Android Studio and iOS SDK respectively. Android studio can be run on either Windows, Linux, and Mac. iOS SDK can only be run on Mac. If the development team is not already familiar with either of these it will take some learning to accomplish using them, but luckily both have good support resources.

Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: I would recommend utilizing Windows for the server side of Draw It or Lose It. It will allow for more straightforward development, as well as support for the system itself down the line.

2. **Storage Management**:

3. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

4. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

5. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>