Draw It or Lose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	05/15/23	Bryce Jensen	<Brief description of changes in this revision>
1.1	06/03/23	Bryce Jensen	Added Evaluation

Instructions
Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

Executive Summary

Our client, The Gaming Room, wants to expand Draw It or Lose It, their current Android game, to several platforms by porting it to a web-based game. The game should allow for one or more teams with multiple players. A singleton pattern will need to be used to ensure only one instance of the game is running at a time and that the game and team names are not reused. When picking a name, the system will alert the players that a name is currently being used.

Requirements

< Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client's business and technical requirements in a clear and concise manner.>*

- A game will have the ability to have one or more teams involved.
- Each team will have multiple players assigned to it.
- Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
- Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

Design Constraints

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

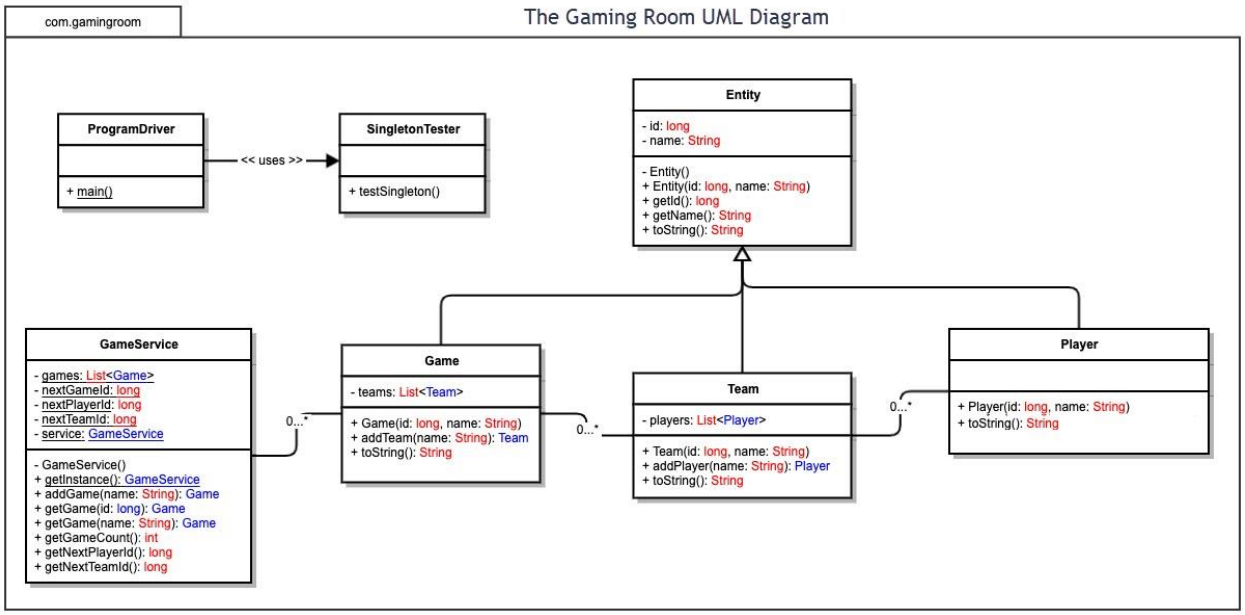
- The languages to be used will be Javascript and HTML5 for the website; the game will be written in Java.
- Singleton patterns must be used for game instance, Team name, and User name checking. Only one instance of each of these may be used at a time.
- Being a web-based game, it should be able to run on most machines. It should be kept "light" to prevent issues upon release.
- No deadline was given in the initial communique. Will follow up with The Gaming Room to determine what that is. Expect an update soon.
-

System Architecture View

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

Domain Model

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>



There are 7 classes. `ProgramDriver`, which `<<uses>>` the `SingletonTester` class. Separately from those two, although all linked in some way: `GameService`, `Game`, `Team`, `Player`, and `Entity` classes.

The `GameService` class uses a singleton design pattern, this is so that only one instance of the class can exist at a time. This is apparent once noticing that the `GameService` class has a private constructor. Only the `getInstance` method can create a `GameService` class, but it would check whether an instance already exists.

Another important method in the `GameService` class is the `addGame` method. This checks – using an iterator for the `games` list – for game objects of the same name. If nothing is found, then a `game` object is added to the `games` list. The `Game` class contains an `addTeam` method and the `Team` class an `addPlayer` method. This acts the same as the `addGame` method above. Using an iterator, it checks for objects of the same name before creating them in the `team` and `player` lists.

The `Game`, `Team`, and `Player` classes are all subclasses of the `Entity` class. This means that they inherit the characteristics of the `Entity` class. The default constructor for the `Entity` class is private. Only the overloaded constructors can be accessed for use.

Several techniques for OOP (object-oriented programming) can be seen in this UML model of this program. Inheritance of the `Entity` class. The `Entity` class also has signs of Polymorphism when overloaded constructors and methods are used. Encapsulation can be observed by setting certain attributes to private only allowing access to them through specific (public) methods.

Evaluation

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client's requirements, and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	While I have no real experience with server-hosting on MacOS, I have read that it can get pricey to host a server on a Mac, and not very many options.	Linux is a very popular server hosting platform. With it being an open-source OS, there are lots of options for webhosting. The costs for running one are generally cheaper than the other platforms.	While less expensive than hosting a server on a Mac, hosting on Windows is still more expensive than it would be on the Linux platform. However, there are lots of options when it comes to software for it.	Most mobile devices can access a server just as well as most other devices, with the only exception being that they may have a less stable connection as they can't be wired to a network. Instead, relying on a wireless connection. Also, their mobile browsers may not have the capability for a long-term connection to a server. I have never heard of a mobile device hosting a server.

Client Side	I believe developing on Mac requires that you use at least one of two languages: Swift or Objective-C. That's it. Those are all your options. Cost will be extreme for equivalent machines on other platforms. Apple products are all more expensive than the equivalents for Linux or Windows. You will get a machine with exceptional build quality and a beautiful, consistent OS. You will NEED MacOS to develop and run something on MacOS.	You can use most of the big languages to develop for Linux-based applications: C++, Java, Python, C, and so many more. Linux will run on basically anything. Anything used in a professional environment will easily be able to handle the very reasonable minimum specs required to run Linux (and even then, there are options of reducing those requirements even further.) Linux is open-source and free to use. So you wont have any costs there.	.NET and C++ are big languages when developing for Windows. There are far more options than that, but these are the big ones. Windows will run on most machines. There are a few limitations but nothing a modern computer wouldn't be able to handle. For licensing cost will be more expensive than Linux, but significantly cheaper than MacOS.	It would be particularly difficult to develop on a mobile device. It isn't impossible, but more time-consuming. If you were to develop on mobile, plan on spending far more time (and therefore cost) on the project. The cost of the actual devices however, will likely be far cheaper than the same number of any other machine on any other Operating Platform.
Development Tools	Apple's Xcode IDE will probably be a necessity. I don't think you need to be running MacOS to develop in the Swift language, but it is necessary to have one to release the product on the Apple AppStore. I believe there are other IDEs that are now capable of using the Swift language. Xcode Enterprise also has a licensing fee of $99 per user per year.	Visual Studio Code is a fancy text editor that has the capability to download all sorts of extensions to improve the development process. It is NOT an IDE out of the gate, however, after downloading the necessary extensions, it can become a very powerful one. This could make it difficult to troubleshoot amongst the development team because of the plugins they may have installed. VSCode is free.	VSCode is always an option, there are tons of specific language IDEs though. Visual Studio is an extremely common IDE for developing on windows, as it has built in C++ and .NET support (and is created by Microsoft.) Visual Studio has a Professional subscription for $45 / user / month but can be used individually for free.	For iOS devices you will need an Apple MacOS product to launch the app on the AppStore. For most other types of devices, any computer with any java IDE will do. There are any number of IDEs for mobile development on any platform.

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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

   My recommended server platform would be a Linux platform. There isn't necessarily a recommended software for the UI. A Linux-based server will ultimately be cheaper than anything else.

2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

   I think that the backend Linux server should be able to manage the game data and have the frontend devices handle the rendering of that data. Having the backend handle it all would make it so "streaming" all that video would take a massive amount of bandwidth. This way, there is a minimal amount of data being sent over the internet.

3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

   A cloud storage method for the game data makes the most sense for this project, any device will be able to access the data at any time and download the current images being used to the device's memory.

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

   I believe Linux uses "least recently used" memory management. This means that as things are used, they are moved to the beginning of a list, as allocated memory begins to get low, the processes or data at the end of the list is removed.

5. **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).>

   This is what a RESTful API is for, the data is accessed (or called) from the client side (mobile devices, PCs, Macs, or anything else that is considered the frontend), then sent from the server (backend) for those devices to "render" that data.

6. **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.>