

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/17/23 | Tiffany McDonnell | The first draft of implementing Draw It or Lose it |
| 1.1 | 02/03/23 | Tiffany McDonnell | Updated draft with evaluations. |
| 1.2 | 02/15/23 | Tiffany McDonnell | Updated draft with recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has asked for a web-based version of the game application Draw It or Lose It. The game will have the ability to contain one or more teams each with multiple players. The names of the game and team will all be unique as well as only one instance of the game existing in memory at any given time.

To create a single instance of the game a singleton design will be implemented. An iterator design will be used to ensure that each game name and team name will be unique.

## Requirements

* *The game must have the ability to have one or more teams be able to play*
* *Each team must have multiple players assigned to each*
* *All instances of game names and team names must be unique*
* *Only one instance of a game can exist within memory at any given time*

## [Design Constraints](#_2et92p0)

* Must be a web-based version that reflects the current gaming application version
* Hardware requirements will be based on software decisions and will come later
* Only one game name or team name instance of any specific name can exist regardless of platform
* Measures are taken to ensure the web version can be played on multiple browsers.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The main part of the UML in this model is the ProgramDriver. The ProgramDriver contains a the main method of the application. It also has a “has a” relationship with and uses SingeltonTester. The SingeltonTester class has a single testSingleton method that is used to validate there is one and only one GameService instance.

GameService contains a 3 private long variables that are used to get next GameId, PlayerId, and Team Id. There is also a single service variable that will hold a single GameService. The GameService class also holds a private games list member. It has a private constructor that helps signify that this class will have a singleton pattern in it. The private members also show the use of encapsulation in how the other classes can not access them. The class also shows a zero to many relationship with the Game, Team, and Player Classes.

The Game, Team, and Player classes all have a “is a” relationship with the Entity class. Due to this fact they have a getId and getName method that they get with Inheritance from the parent class Entity. These child classes use abstraction in order to create their own constructors. While the parent class Entity uses polymorphism to override its constructor with a different constructor based on its parameters.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS is originally based on a Unix operating system. It has a nice flow between software and hardware. Has decent security making malware less of an issue. Runs very well while multitasking .  Amazing capabilities for integration with other Apple products.  As a disadvantage MacOS if extremely limited with hardware upgrade capabilities. | Linux is an open-source operating system that can be used by anyone. It doesn’t take much to run Linux, so it requires less memory to operate appropriately. Linux is also considered favorably safe. I have a security authorization that requires an admins password to install an application before any virus could be brought onto the computer.  A disadvantage Linux has is due to a steep learning curve in is not considered very user-friendly. | Due to the high popularity of Windows, there is an abundance of different software and each is different in its own way. It has great capabilities to upgrade its hardware. The uncertainty could be considered a disadvantage while choosing what is best for the job can be an advantage.  Another disadvantage would be its poor security and it has high resource requirements. | Mobile operation systems allow for secure access to data. Apps can continuously run in the background causing the operating system to slow down. Running a web-based game as and the app will help to lower strain on the operating system. |
| **Client Side** | Mac is considered user-friendly and simple to use. The format can be very eye appealing to most users.  A disadvantage on the client side is that MacOS can become costly. Mac usually comes with some features already installed for ease of use for the user. Not built for gaming and can be a con for certain users. Apps available for MacOS is are also more limited than that of other operating systems. | Due to being open source, it is a cost-effective operating system. The operating system is considered very safe, therefore there is less worry about security on the user side.  A disadvantage would be difficulty finding vendor support when changes are needed can potentially slow down progress. | Due to there being just an abundance of software with windows, it becomes a more cost-effective way to go. Since it is so commonly used most people have a familiarity with it making it easy to use. | Users have more freedom to multitask, unlike other operating systems. Mobile support is very high and convenient. A disadvantage would be that some demographics have issues with updated technology making it a mobile device difficult to use. They also allow ads in many apps which can be frustrating to the user. |
| **Development Tools** | MacOS was created by Apple. Based on Unix and developed specifically for Apple mac computers.  It uses written with an objective C language. One IDE that could of been used is Notepad++. Original API’s include Carbon, Cocoa, Java, and BSD-POSIX | Linex kernel is written in C. Knowing that it is written in C some IDE’s that could have been used are Dreamweaver, Code:: Blocks, Eclipse, CLion, Visual Studio Code, NetBeans, Codelite, or Atom. | Windows was created by Microsoft Inc. and purposely designed to be used by all companies.  It was written in C++ and kernel in C. This being the case Visual Studio, Eclipse, NetBeans, Visual Studio Code, or Code::Blocks would all make great choices for IDE. Original API’s include Win32 and NT API | In recent times some IDE that has come about to help the development of mobile operating systems is Android Studio and Xcode. |

## Recommendations

1. **Operating Platform**: Recommendations for the Operating Platform for Draw It or Lose would be Linux. Linux is opened sourced and can help drastically cut costs that can be used then for something more beneficial. Linux is widely known across the field so it can be picked up easily by new developers that potentially get hired as the application updates. Being on a Linux also helps add in some extra security measures since it is one of the most secured operation platforms out there.
2. **Operating Systems Architectures**: The Architecture recommendation would be to have both front end and back end. The use of back end would be used for the crucial processing part that allows the application to run. The servers would be able to hold the most important data that is needed for Draw it or Loss it to be what it is. On the client side, we can store small bits of data like user input and render the images retried from the server.
3. **Storage Management**: Storage Management should be upheld on the server side. Storage will be used for the main function of the application. Storage should be used using a cloud server that helps to be used more coarsely with other operating systems.
4. **Memory Management**: Memory Management should be done on the client side. It should be handled with a user’s local RAM memory. It will be used for user input and other temporary files needed to run.
5. **Distributed Systems and Networks**: For everything to function the way it is supposed a RESTFUL API would be utilized. This will ensure that multiple operating systems could download the application. It will help with the communication between the user and the server. This will also allow the server to store any vital information after a variety of checks that will have to be stored from the user that will be needed to be passed down to a separate user later on.
6. **Security**: The security of the server for Draw It or Lose It will consist of a role-based authentication. This will all only an Admin to go in and change any vital things with the application. This authentication will also keep things simplified on the user’s end. They can only access what is necessary for them to touch and not interfere with anything that could cause a malfunction.

***Citations***

Lavieri, E. D. (2019). *Hands-on design patterns with Java: Learn Design Patterns that enable the building of large-scale software architectures*. Packt Publishing.

GeeksforGeeks. (2022, January 28). *Difference between windows and macos*. GeeksforGeeks. Retrieved February 4, 2023, from https://www.geeksforgeeks.org/difference-between-windows-and-macos/

*Advantages of linux - javatpoint*. www.javatpoint.com. (n.d.). Retrieved February 5, 2023, from https://www.javatpoint.com/advantages-of-linux

Libretexts. (2022, January 7). *01-A.8: Advantages and disadvantages of linux*. Engineering LibreTexts. Retrieved February 5, 2023, from https://eng.libretexts.org/Bookshelves/Computer\_Science/Operating\_Systems/Linux\_-\_The\_Penguin\_Marches\_On\_(McClanahan)/01%3A\_Introduction\_to\_Linux/1.08%3A\_Advantages\_Disadvantages\_of\_Linux#:~:text=Disadvantages%20of%20Using%20Linux,-The%20main%20concern&text=It%20is%20a%20bit%20more,from%20Windows%2C%20or%20even%20MacOS.