

Draw It or Lose It

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 6/19/22 | Seth Mascho |  |

## [Executive Summary](#_sbfa50wo7nsh)

Eclipse is a simple, easy to use programming environment that you can use to build your game, Draw It or Lose It. It allows you to publish the project into a web ready file in just a few simple clicks.

## [Design Constraints](#_2et92p0)

Each game must have one or more teams involved for the game to be played. Each team can have multiple players assigned to it. Each game must have a unique name for a player to check if a game already exists. Teams also must have a unique name to reduce confusion.

Each game only lasts 4 rounds, with each round lasting one minute each. Each drawing is rendered in a steady rate and are fully rendered after 30 seconds. If after the 30 seconds, the team currently guessing loses their turn, and the other teams have 15 seconds each to offer one guess.

## [Domain Model](#_8h2ehzxfam4o)

Game, Team, and Player all extend the base class Entity. Entity has 2 private variables, a long name id, and a String called name, as well as public accessors that allow users to retrieve the values of each variable. Entity also has a toString method that prints out its id and name.

Since these classes all extend Entity, each class also has an id and a name and the accessors that come with Entity.

The Game class has a list of each team currently in the game, with a method to add a new team with a name. Game also overrides the base Entity toString method.

The Team class has a list of each player on the team, with the ability to add a new player to the team using a name. Team also overrides the base Entity toString method.

The Player class is a bare bones extension of Entity, only changing the toString method.

The GameService class contains a static Instance of itself for every class to access. It keeps track of each game, team, and player, giving out a new id each time a new entity is created. GameService is where a new game can be created, which paves the way for new teams and players to be added to said game.

ProgramDriver is where the GameService Instance is initialized, and it uses SingletonTester to ensure that the Instance of GameService has been initialized correctly.

**"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** | Mac users have access to free, pre-installed server application, such as the Apache web server. The Apache web server, as well as DynDNS aren’t built for high traffic websites. The initial cost of getting into mac is expensive | Linux has many free, lightweight ways of hosting a web server. It doesn’t require a large amount of system resources to run, unlike windows or mac. | Requires more resources to run a server because of system limitations that may cause the server crash should the hardware be older. | Simple to setup a server but it comes at the cost of performance. Don’t expect the server to handle many people. |
| **Client Side** | For Mac OS X Server, the 10-client license is 500$, while the unlimited is 1000$. However, if you wish to upgrade to the newest version of OS X Server, you will need to re-buy the newer version. With how expensive it is, I don’t believe many people will be buying it, which means it’ll cost extra to higher someone who can use it. | A user could set up and host the server themselves, but it would require knowledge of the Linux Commands. Or you could use a hosting sight for as low as 10$ per month up to 210$ per month. | A Windows Server license cost anywhere from 20$ per month for the standard edition to 125$ per month for the datacenter edition. It does have an easy-to-use interface, more hands-on support for novice users and a fast setup time. Seems to be cheaper and easier to use than Mac. | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.> |
| **Development Tools** | There doesn’t seem to be a lot of options for Mac. Only 2 options popup with a google search. | Linux has many free open-source web application frameworks. | Windows has the ASP.NET Framework where you can build web apps from scratch, but windows also recently released a free tool for building websites. | There are plenty of development tools on the play store for both android and iOS, but the quality of each app may vary. |

## Recommendations

1. **Operating Platform**: Linux is probably the best system to host a server on. It offers low memory usage, several different free Linux distros, such as Ubuntu and Debian. All operating systems can connect to the server, so there aren’t any issues with connectivity.
2. **Operating Systems Architectures**: For Linux, the operating system architecture is as follows, the kernel, system libraries, hardware layer, system, and shell utilities. The kernel is responsible for each major action of the Linux OS. System libraries implement the OS functionality and don’t need code access rights. The hardware layer contains server devices such as the CPU, HDD, and RAM. The shell is the final layer, which takes in commands from the user and runs them through the kernel. There are 2 main types of shells, graphical and command-line shells. Graphical is closer to what Windows offers, and command-line only has the command prompt, which the user writes to.
3. **Storage Management**: A raid controller would be a good system for managing storage. A raid controller essentially combines multiple different hard drives into one drive and stores a second copy of your data. This can help should one of your hard drives die, potentially allowing you to retrieve the data from the dead hard drive. A raid controller increases may also increase performance in some applications.
4. **Memory Management**: Ideally the server is Linux based, the overall memory requirements are lower than the others. For example, Microsoft recommends 4gb of memory, whereas one version of Linux, Ubuntu, a popular choice for Linux servers, recommends 2gb. It would stand to reason that with the server being so light weight, the server could dedicate more memory to Draw It or Lose it, and still run with no problems. This low memory usage is achieved with a few different management systems, such as Zones, which combines memory into zones according to the possible usage, or Nodes, which houses all the zones of a certain type.
5. **Distributed Systems and Networks**: Distributed systems and Networks provide support for communication and storage for a large amount of people and software processes. It works by splitting work across multiple computers on a network, coordinating their efforts to complete a job more efficiently than a single device.
6. **Security**:

There are 2 options that I can think of. One option is to not have a player account, and only have a player create a temporary name that they use only when playing. This means that there isn’t any information stored about the player. No password or important information is needed.

The other option is to store the user’s information in an encrypted file, somewhere on a

separate ‘user\_information’ server. There are multiple different encoding formats, the most common are UTF-8. UTF-16 and UTF-32. Through this ‘user\_information’ server, have ways of verifying the user, such as a password, an ability to send a temporary code to the user’s phone or email, both of which will be censored from the user, incase they are not the owner of the account.