

Draw It or Lose It

# **CS 230 Project Software Design Template**

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 | 11/09/2022 | Tyler Price | Added new class Entity  Classes Game, GameService, Team and Player all inherit from Entity class  Use Iterator method to move through game list |

**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](#_sbfa50wo7nsh)

The company The Gaming Room is looking to expand their game to multiple platforms to increase the number of possible clients. Their game, “Draw it or Lose it”, is currently on the Android app store, but through communication with necessary companies and between The Gaming Room leaders and project members, the game will be on Apple app stores in the near future.

## [Design Constraints](#_2et92p0)

1. Team names and player names must be unique in every instance
2. Must be able to run on multiple platforms and be compatible to play cross platform as well
3. Uses Java as language
4. Every game needs to have multiple teams be able to play at the same time
5. Every team needs to be able to have the multiple players playing at the same time

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

The UML diagram displays 7 classes. The classes Game, Team and Player all inherit from the Entity class and then share information between each other shown by the zero to many relationships in the diagram. The GameService class also does this but through class Game. This diagram displays how each player and team playing the game will have their own unique id number and name for each individual player/team. The SingletonTester class tests to see if there is a single instance of the game running at a time, and the ProgramDriver class houses the main function.

**"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** | For a Mac system, it will require someone who specializes in running software on the operating system to keep it running. If this is possible, the Mac OS will allow the client to get the most out of the servers. However, having someone specializing it keeping the OS running could be costly, but other than that Mac would be a great choice for the server side. | Being the most popular OS used for running and maintaining servers, the cost will be less and it will be more manageable. However, you will still need someone dedicated to running the server which could increase the costs of the project. | Windows would be the easiest to maintain and shouldn’t require someone to specialize in running the server to keep it up. However, the initial investment in the OS is the most expensive out of all of the Operating Systems. But another strength of this OS is the accessibility it gives to their clients. | While mobile devices reach a larger client audience, for the best performance on the server side would lie in any of the previous operating systems. |
| **Client Side** | Mac will be the more labor-intensive clients. There needs to be an OS specialist that would need to make sure that the code is as simple as possible to ensure cross platform compatibility and that the code is compatible with the operating system. | The cost of using Linux doesn’t differ much from using Mac due to the resemblance in needing someone to constantly maintain the project due to Linux not being a widely used platform. | Windows should take less time to set up and use due to how widely and commonly used the OS is. It also gives the team tools that will help with development that will cut down on the amount of time it will take for the team to develop the project. | Like Windows, mobile devices are very highly used. The only issue may arise from the testing in making sure that the application is working properly and looks good across all platforms throughout the development of the project. |
| **Development Tools** | Languages used: HTML/JavaScript/CSS/C++/Java/Python.  Tools used: Eclipse/GitHub/Visual studioSQL. | Languages used: HTML/JavaScript/CSS/C++/Java/Python.  Tools used: Eclipse/GitHub/Visual studioSQL. | Languages used: HTML/JavaScript/CSS/C++/Java/Python.  Tools used: Eclipse/GitHub/Visual studioSQL. | Languages used: HTML/JavaScript/CSS/C++/Java/Python.  Tools used: Eclipse/GitHub/Visual studioSQL. |

## 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**: Windows would be the best choice for the game’s operating platform. This platform is the most cost-effective choice amongst the other options and gives it the best chance to be operable across other environments.
2. **Operating Systems Architectures**: Windows offers user mode (general use) and kernel (encryption of data. Requires a hard connection)
3. **Storage Management**: SSD (solid-state drive) would be the best storage for the application.
4. **Memory Management**: Paged allocation is used to separate old and new memory from each other. It is separated into fixed frames and then uses virtual memory space to allocate the data into smaller sizes.
5. **Distributed Systems and Networks**: Connecting different servers and distributing the application across them is the smartest way to avoid high traffic in the game servers. This will also increase the efficiency of the game and decrease the load on the servers when there is a lot of game traffic.
6. **Security**: Secure Socket Layer and cloud security will be the best options on windows to prevent hackers from extracting game/player data or from hackers messing with the game itself.