

The Gaming Room

# **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 | 11/10/2021 | Amanda Vazquez | Template updated based on information provided by client |

**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 we are working with, The Gaming Room, wishes to make their game “Draw It or Lose It” into a web-based game that functions on multiple platforms. They currently have the game available as an Android app.

## [Design Constraints](#_2et92p0)

First, we will have to use a cross platform game engine, such as libGDX. The design must include an iterator that runs through existing game and team names to ensure each new name is not already in use using the singleton pattern. The design code will also have to include an instance variable since only one instance of the game can exist at any given time. My team will also have to design the Team and Player classes so that there can be multiple teams and multiple players in the game.

## [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 Player, Team, and Game classes all inherit from Entity, which links these classes because they inherit from the same parent class, an example of the inheritance principle. Entity being the superclass is an example of polymorphism. The class ProgramDriver is by itself and doesn’t inherit from nor pass down information to any other class. It utilizes the SingletonTester class. ProgramDriver is an example of abstraction since there are no objects created from this class. Encapsulation is used where we see all the minus signs in the diagram, which helps hide information that the user does not need.

**"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)

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** | Mac has quality hardware and ease of use. Hardware options, however, are limited. Mac server is easy to use and powerful. Top of the line hardware. Very secure software. | Linux is the most secure and is open-source. However, there is a learning curve with Linux, and it is complicated to use. All users benefit from a strong community of resources and advocates. Less resources needed to run Linux. | If using Windows Pro, corporate networking is built in, so you don’t have to get extra add-on software. Windows has tons of great hardware options. One disadvantage is Windows has security issues and is vulnerable to malware, spyware, and ransomware. | A cost-effective option with better compatibility. Variation in devices might prove difficult for implementation. |
| **Client Side** | A moderate amount of expertise and time required. Mac is not very difficult to use and is very user-friendly. Mac is on the expensive side. You must have a Mac system built by Apple in order to use. | The most expertise and time required. Linux may be challenging for user to first grasp. Linux is free and can be used on Dell computers, even older models. Anyone can download Linux and use it so it is cost effective. Linux has great speed. | A light amount of expertise and time required. Windows is the easiest to use and the most user-friendly. Windows is popular for gaming. Windows is somewhat affordable. | Not too expensive. A moderate amount of time and expertise required. Difficult to implement. |
| **Development Tools** | High cost. Developer tools include Visual Studio, Sublime Text, Homebrew, iTerm2, Visual Studio Code, Notepad++, Xcode. Programming languages include Swift, C, C++, Python, | CentOS can be useful.  Linux is free to use. Different programming language options with Linux are C++, Python, Java, and C. C++ is most commonly used for game development so that’s a good option. Linux is the least expensive option. | Somewhat high cost. Developer tools include Eclipse, Pycharm, Visual Studio, etc. Java and JavaScript are great, as well as C#, PHP, and Ruby. | Swift is a programming language used to create apps for iOS and OSX. The preferred IDE for Mac apps is Xcode. Java is the main language used to develop Android apps through Android Studio(IDE). To distribute an app on a mobile device, Apple requires a developer account which costs $99/year. Android runs even higher- $70- $320 per month. |

## 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**: My recommendation for The Gaming Room is for the game to run on Windows. I chose this operating platform because Windows 10 has great performance when it comes to gaming and games typically run better on Windows than on any other operating system. Windows 10 has scored very highly in every performance test. Windows is a very popular option for gamers. It’s also very user-friendly. Windows requires minimal expertise and minimal cost to utilize for our program.
2. **Operating Systems Architectures**: Windows is affordable and runs smoothly. Windows has reliable performance and great security. Windows has a variety of quality software.
3. **Storage Management**: For storage management, this game will utilize cloud-based storage. The storage for the game will not live on the server, it will be hosted by the cloud virtually. Windows also provides us with a disk cleanup tool.
4. **Memory Management**: This game will use the file system to store 200 high-definition image files into one larger file to keeps things organized. Our game will use the indexed allocation method for proper memory management. Since we only have 200 image files, I think this allocation method would be fine since we don’t have to worry about the program having to read too many index blocks. As for managing free-space, I think grouping would work well for our game program.
5. **Distributed Systems and Networks**: This game will use a client-server distributed system so that both data and transaction processing are divided between one or more computers connected by a network. This will provide a string foundation and network for multiple players to be able to connect with one server, as this game has a multi-player option.
6. **Security**: An SSL certificate will definitely be necessary to keep user data secure for any websites we create for our game. Also, Microsoft has Windows Defender, an antivirus protection plan, already built into Windows 10. We need to also be sure to make use of the principle of least privilege as a security measure. Another effective security measure this game will be using is encryption of data. So that way, if there were ever a breach, the data is scrambled.