

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 |
| --- | --- | --- | --- |
| 3.0 | 04/15/22 | Douglas Bolden | Final Commit |

**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 Client, The Gaming Room, doesn’t know how to set up the environment. I will be creating a concept for their online gaming app, Draw it or Lose it. I will be giving my developed program to my technical manager when the concept is finished.

## [Design Constraints](#_2et92p0)

Budget is currently TBD.

HTML, CSS, PHP, and PHP will most likely be needed to implement this in the correct fashion on a web server.

The timeline for this will be determined when we are in development.

We would like your feedback during the development process.

The device will need to have the latest Java updates installed to function properly.

We will need to discuss the game, and exactly what needs to be done in the future!

## [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)

This program contains a Program Driver, which uses a Singleton Tester. The Program Driver, “ProgramDriver,” starts the program, and then the Singleton Tester, “SingletonTester,” is used to make sure that there is only one instance for each item that is passed to it! The Super Class, “Entity,” contains all the needed information that the subclasses, “Game, Team, and Player,” will use to produce useful results! “GameService” is used to create an instance of the game so that more than one instance can’t exist in memory. This is extremely useful, as it allows the program to run using a small amount of memory, making it easy to manage and allowing the minimization of security problems and memory leaks. “GameService” also sets up an index of the games that are started and makes them easier to view. “GameService” will also provide a way to display the team ids, and the player ids as well! “Game” will set up the index for the teams, and then provides a way to display the teams. “Team” will set up the index for the players, and then provides a way to display the players. “Player” will provide information used to create a player in the “ProgramDriver” class, and then provides a way to display the player’s information.

**"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 is used by the US Military for its great security. Runs on Apache servers. Servers do cost a lot. | User-updated security, as it is open source.  Runs on Apache servers.  Servers have varying prices. | Frequent Updates for security. Website may go down for maintenance. VITAL for ASP, .NET, Microsoft Access, and MSSQL Databases. | Most mobile devices will have security updates but will usually remain updated for those with the storage available. Power will be the biggest issue. |
| **Client Side** | Must have updated Java to develop and run the application. Time restraints are to be debated. Mac is the most expensive option for running this application. ‘Hello.java’ will be used. | Must have updated Java to develop and run the application. Time restraints are to be debated.  Linux pricing is in the middle of all the other device types listed here. ‘sudo’ will be used. | Must have updated Java to develop and run the application. Time restraints are to be debated.  Windows is the cheapest option.  Java must be updated and will use the Java program. | Must have updated Java to develop and run the application. Time restraints are to be debated. This can be the most expensive, or the cheapest. If Java is supported, any device could be used, in theory. A java app will be used. |
| **Development Tools** | For this type of software, the best tools to use would definitely be:  Eclipse  IntelliJ IDEA  NetBeans  BlueJ  We will be developing in the Java programming language. | For this type of software, the best tools to use would definitely be:  Eclipse  IntelliJ IDEA  NetBeans  BlueJ  We will be developing in the Java programming language. | For this type of software, the best tools to use would definitely be:  Eclipse  IntelliJ IDEA  NetBeans  BlueJ  We will be developing in the Java programming language. | For this type of software, the best tools to use would definitely be:  Eclipse  IntelliJ IDEA  NetBeans  Java Development Apps  We will be developing in the Java programming language. |

## 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 Server will be the easiest and cheapest to develop for. The source code could be ported to the other Platforms easily. There are ways to check what system is being used, but porting has less chance of having bugs or security flaws. Windows Server is just the cheapest for both time and cost.
2. **Operating Systems Architectures**: Ample Memory for an application like this is in the Windows Server. It allows an easy to manage list of files and a vast amount of possible programming languages.
3. **Storage Management**: Solid State Drive allows for the quickest reading and writing speed. This could be installed, and the Windows Server installed on it. This would allow for quick IO operations.
4. **Memory Management**: Windows Server will make Memory Management the easiest, as it contains many applications for Memory Management, in GUI form.
5. **Distributed Systems and Networks**: Linux servers may be the best approach for the server, as it makes it possible to keep the game servers up for an extended period, has less server crashes, and has easy-to-deploy maintenance, when needed. HOWEVER, since most of my recommendations have been for the Windows Server Operating System, it may be best to stick with it. It allows for easy communication, but it does have a few issues with routing.
6. **Security**: Windows Server makes for a hassle-free security operating system. Updates are automatically installed, so all the user must do is restart their computer after an update has been installed. These updates will protect the user from external threats.