

Draw It or Loose 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 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/28/2024 | Michael Armstrong | Continued to add more instances to keep identifies unique and hold the game in one position of memory |

**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 problem that we faced was to create an environment where only one instance of a game was started, where no one else can use the same game, or team name, until the game was completed. In order to solve the problem we created conditions in the programming to only allow one game name to exist in memory.

## Requirements

The requirements were to only allow one instance of a game or team name to exist in memory. If another player tried to use the same name that is already in use, the service would prompt the user to enter a different name.

## [Design Constraints](#_2et92p0)

This application is written in Java for use in multiple operating systems. This application was originally written for an Android application and is being translated to a web based service to be run on a variety of different operating systems, including Mac, Linux, Windows, Android, and Apple.

## [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 below is used to outline The Gaming Room’s application. Starting with the class titled GameService it contains a list of all games in memory. This class also stores and acquires information such as a players game id, the game id and the team name. The Game class holds the list for the teams, and uses the information from the GameService class. The Team class holds a list of players on a particular team. The team class get this information from the Player class, which holds the player name and ID. The classes listed so far all used by the Entity class that is the final checkpoint to ensure that there is a single entity of a team name and game in use and stored in memory.

The ProgramDriver contains the main method which runs the application, and it uses the SingletonTester class to check that no one else is using a name that is already in use.

"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 uses Apache 2.0 which boasts support for multiple IP addresses and virtual domains | Linux is an open source OS, where it is constantly being upgraded by users and the Linux team | Windows allows businesses to host applications that host websites, allows communication with other OSes. Infrastructure servers that run security gateways, process servers, and more. | When using a Mobile Device, you need to create the server yourself, and post it to the web server using a REST interface. |
| **Client Side** | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.> | <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** | Mac has access to MySQL, and can connect to other database languages. Mac can use HTML, Java, and Python to create content for the client. | Linux uses a multitude of programming languages to create lots of different programs | Windows offers a multitude of programming languages to create lots of different programs. Some include HTML, Java, Python, SQL | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.> |

## 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**: I would recommend expanding to Linux. Linux is a similar operating system to windows and there are a lot of users who are constantly using and upgrading and modifying the operating system. Linux would open a lot of opportunities for The Gaming Room.
2. **Operating Systems Architectures**: Linux architecture consists of the hardware, the Linux kernal, the shell, which is a command line operator, its systems library, and its applications, which is similar to Windows, without the shell.
3. **Storage Management**: Linux uses SSDs and HDDs for file storage. Additional storage can be added, and reconfigured using the command line.
4. **Memory Management**: Linux uses standard memory through the CPU to access its memory. It can be used to create virtual memory and it can be used to allocate memory through RAM
5. **Distributed Systems and Networks**: Linux would be able to communicate with systems like windows through similar VPNs. The program would need to allow file share services so that the two OSes would be able to communicate with each other over the network that The Gaming Room runs. The systems would not be able to access files and memory from any file folder other than The Gaming Room files that are on each computers storage.
6. **Security**: The Gaming Room would create multiple fire walls that would be around the zipped files of its gaming application. Linux offers SELinux, which is an enhanced security rooted into the Linux kernal. It only allows Mandatory Access Controls, which is for the computers administrators to access. This would protect The Gaming Room’s Linux users from Malware and other security threats while they are playing their games.