

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this revision> |

**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](#_heading=h.35nkun2)

The Game Room is looking to create a web-based app based on the game Win, Lose, or Draw. It is currently available only on Androids, but moving to a web-based app would allow multi platform use. We can create a solution using Java, which is a platform-independent language, and build a web application that supports object oriented programming principles. Using object oriented programming principles supports data privacy which will make our app more secure, and will create a tidy application that runs quickly enough to satisfy demand.

## Requirements

The Game Room is looking for a program that can have one or more teams involved, and each team can have multiple players. The game and team names must be unique, and this needs to be able to be checked against a list of names already in use. That can be done using the iterator pattern. Finally, since only one instance of the game can exist in memory, unique identifiers need to be used for each instance of a game, team, or player. This will be accomplished by use of the singleton pattern.

## 

## [Design Constraints](#_heading=h.1ksv4uv)

The main design constraints are as follows:

-The program must allow for at least one team, and each of those teams must be able to have multiple players. This means we must be able to keep track of who is on what team, and unique identifiers will support this.

-Game and team names must be unique, and therefore there must be software supporting this (singleton pattern, for example). Obviously, unique identifiers assist in this aspect as well.

-Only one instance of the game in memory at any given time- this means we need to be sure that the singleton pattern is in use.

-Must be multiplatform-able. Programming in Java was a good choice for this, because Java is platform independent.

## [System Architecture View](#_heading=h.44sinio)

*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](#_heading=h.2jxsxqh)

The classes relate to each other as such: The base class, Entity, has the subclasses Game, Team, and Player. They all share some methods and attributes that they inherit from Entity. Because they inherit these from Entity, they do not have to be constructed in each subclass. This saves programming time, and program runtime, which can help to keep costs down.

The class ProgramDriver “drives” the whole thing, as it holds the main interface. This class is responsible for starting the game, retrieves the singleton instance from GameService, and drives I/O. It also constructs a SingletonTester instance and this instance calls the testSingleton method from the SingletonTester class.

The object-oriented programming concept of a singleton pattern allows us to create only one instance of the game in memory, and use it across classes. This allows us to keep the code neat and data protected, while keeping clear track of who is on what team, what number of game we are on, etc.

**"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](#_heading=h.z337ya)

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 no longer has a server associated with it, so I would not suggest Mac to host a web-based software application. That said, some of its original features like Caching Server, File Sharing Server, and Time Machine Server, are bundled in any High Sierra or later version. | Linux is very flexible, and once learned, its command line interface makes even as difficult a task as hosting a server, relatively easy. As an added benefit, Linux is open-source, and therefore free. | Windows is well-known, and moderate cost. It has decent built-in security, and has a lot of applications within the Windows family to do things such as host a web server. Its larger base of applications is a main advantage. | Mobile devices simply don’t have the power to host a web-based software, at least not one as in depth as the Gaming Room. They may be relatively cheap and intuitive but they would not be able to keep up with demand, and would lack security, as they would need to implement a peer to peer file sharing protocol, which simply isn’t that safe. |
| **Client Side** | Mac offers a flexible client side experience. While it takes some know how to navigate, it’s not too complicated, and for more experienced users, the command line interface is still accessible. However, Mac products tend to be expensive, and this could be prohibitive. | Linux takes more time to learn than any of the other options. This could create a cost issue in the sense of time spent learning Linux systems. However, Linux itself is free, as it is open source. This would be a point in Linux’s favor. | Windows is very easy to work with, and very intuitive for most, as it is the most popular OS. This means there would be little to no learning curve. The cost is less prohibitive than Mac, although it may be less efficient than using Mac or Linux. It does, however, have more native applications available for development. | <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** | Swift is Mac’s native language, and is obviously compatible with Mac. Other languages such as Java, Scratch, and Python are compatible. For IDEs, you can use Eclipse, Atom, and CLion, PyCharm, and Visual Studio are all compatible with Mac. | C would be a great choice for anyone using Linux, as linux is closely based on C. Additionally, Perl is installed on Linux by default, however, it is becoming outdated. Python and Java are also well-supported in Linux, Java especially. Visual Studio is a great choice of IDE for Linux, and is well supported. | Almost any language or IDE you can think of is easy to work with on Windows. As an added bonus, both Visual Studio and Notepad++ are native to the Windows environment, and run especially well there. | There are some applications you can download to do development in a mobile device environment. The languages supported in these apps are mainly web-design languages such as CSS, HTML, and Javascript. |

## 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 recommend that Linux be used to allow the Gaming Room to expand Draw It or Lose It to other computing environments. Linux is the basis for many other applications, and will run well with Android devices.
2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

A brief overview of Linux architecture: Linux is composed of a kernel, which is the interface between the hardware and the processes. It is a way to implement resource management. Linux runs from a shell, which is a program that takes input from an I/O device, usually a keyboard, and translates that to a command to the operating system. The shell works to create applications. Part of these applications, and Linux’s OS, are system libraries. They access kernel features and implement most of the functions within the Linux OS. They don’t require actual access rights to the kernel, which speeds processing time.

I would choose a multi tier architecture for this application. This is because this application is going to be web-based. Multi tier architecture works as follows: there is the user-interface, client environment, the logic or application level, which is in the middle, and finally the server-side database. Multi-tier architecture allows for faster scalability, and multi-tier systems generally run faster. They are also more secure, because there is a layer between the client and the server.

1. **Storage Management**:

Linux has its own storage management system, known as the Linux filesystem. There are some cloud storage providers that work with Linux. pCloud is the most recommended, and it is very secure. pCloud has many back up options, so you can store things outside of the cloud. It is cost-friendly, too. Dropbox and MEGA are also storage environments that are native to Linux.

1. **Memory Management**:

Linux uses memory management techniques that are all part of its memory management system. Linux uses paging methods to call specific blocks of memory to its virtual memory. This is a unique approach, but it works quickly. Instead of using a swap file, which is generally slow, it uses dedicated partitions within the system. The user can choose the size of these partitions. The user can also choose specific settings for the memory management system, depending on what fits their needs best.

1. **Distributed Systems and Networks**:

Draw It or Lose it will be ran through a distributed system. That is to say, that because the application is web-based, it will be run through lots of different computers. There will be a database, likely a serverless one that will hold the various information that the client servers will be accessing. Because it will be ran through a multi-tiered system, the actual application logic will not be on the server level, but in the middle level by itself.

What this will look like is the application is accessed via the web, which means via the HTTP, and not through a device’s storage. The information presented to the user is on the client side, and the logic tier takes the information collected from the client side, and processes it on its own, in our case using the language Java. From there, the logic tier communicates with the server side tier to get the needed information.

Using the web, this should be relatively easy, but there would be many concerns if there was a service outage. The main concern would be if there was a service outage on the server end, as that would take out any use of the game. A service outage on the logic level would be equally devastating. Service outages here and there on the client side would not be as great a concern, because that doesn’t eliminate access to the game for other users. Just the ones affected by the service outage, and those are usually beyond a development team’s control- e.g. power lines going down.

It is possible to use RESTfu APIs with Linux. This would increase security and make data transfer more efficient. However, as this architecture is already multi-tiered, using a RESTful API on top of a multi-tiered system may be redundant.

1. **Security**: Linux is widely considered to be more secure than any other operating platform. Linux has 3 security levels, and the user can choose which they would like to run. Security level 1 has a list of basic restrictions, the important features being that it prevents raw access to file partitions (this prevents anyone who does get access to the disk to go through the file system, which would make changes easier to see). Security level 2 has the additional restriction that reads to the raw disk partitions are prohibited. Security level 3, the highest security level, has the additional restriction that changes to the IP filter are not permitted.

Citations:

*5.4.2 BSD Kernel Security Levels*. Документация - unix.org.ua. (n.d.). Retrieved February 19, 2023, from https://docstore.mik.ua/orelly/other/puis3rd/0596003234\_puis3-chp-5-sect-4.html#:~:text=Using%20kernel%20security%20levels%2C%20you,this%20fact%20in%20your%20logfiles.&text=Level%201%20is%20used%20for,really%2Dreally%20secure%20mode.%22