

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 | 09/18/22 | James Jones | Addition of Entity class |

## [Executive Summary](#_sbfa50wo7nsh)

The requirements require that only one instance of a game can exist in memory at any given time. One way to accomplish this a singleton pattern can be used to ensure that only one game of a unique name can exist at one. This will ensure that there will not be multiple duplicates within memory. This helps with optimization by ensuring that memory isn’t cluttered.

## [Design Constraints](#_2et92p0)

As a web-based application one of the biggest constraints will be budget management and scalability. Since Draw It or Lose It is web-based there will be some optimization issues that need to be overcome in order for the game to run on numerous different platforms. Users will expect the game to perform on their computers as well as on their smartphones ensuring the game has mobile compatibility will take additional time and funds. The game will need to scale to mobile platforms in order to ensure consistent growth. The app also needs to be compatible on Microsoft, Mac, and Linux adding additional frameworks that will need optimization. Building a game with this much cross-platform compatibility will have a large impact on the budget.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

**"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.**

Beginning with the new parent Entity class which has the id and name variables as well as the accessor methods for those variables. The id and name variables are protected this is an example of encapsulation within this program. This ensures that the id and name cannot be changed which is why there are not mutator methods but they can be easily seen without having to go into the Entity class. The Game, Team, and Player classes inherit those variables and functions from Entity as children classes.

This helps save time due to not having to constantly recreate these variables within each class.

## [Evaluation](#_2o15spng8stw)

The Game class inherits from the team class which in turn inherits from the player class. This allows for creating multiple, small, and easy to understand classes. This helps with organizing, testing, and troubleshooting the program. The Player class get player information, the team class inherits that information from Player adds it to a player list, The Game class inherits that information and adds the player to the team list. The GameService class which is a grandchild class of Entity. This class inherits all the way through the chain and can create a new game or bring up a previous game when given name and id. The ProgramDriver class holds the main function and will run the game but currently is connected to the SingleTester class which is the first test case for the program.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac OS X has the stability of Linux while having many user friendly designs. Mac is built off the same systems as Linux giving it the similar stability and performance. This hybrid design is making Mac a preferred choice for many. | Linux is generally more budget friendly, however isn’t as easy to use as other platforms. Linux also has show incredible stability and performance making it a safe platform choice. | Windows is known for being easy to use, making this the preferred platform for many users. There is however less stability as Linux this being a big drawback. | Mobile devices are everywhere, this is a big thing to consider. Nearly everyone has a smartphone with increasing expectations for compatibility with every app/website. This creates problems due to each mobile platform being unique. Individual mobile platforms are easy and relatively cheap to work with however the need for compatibility across numerous different mobile platforms can be difficult and expensive. |
| **Client Side** | Mac is tightly integrated with IOS making communication between Mac and Iphone very easy. This makes targeting the IOS customer base easy, however Mac has very limited options for handling apps developed for Microsoft. Since Microsoft has the biggest share of computer users this could be an issue. | Linux is open source giving many options for development, however many popular apps aren’t available on linux. There is also very few pre-built Linux machines meaning a very limited customer base. The biggest pro for linux is budget however the lack of accessibility for the average person is lacking due to few pre built system. | Microsoft has the largest share of computer users making this an important go to platform. Microsoft also has far more flexibility due to the shear amount of users worldwide. This has led to multiple companies building Microsoft based computers, which means there is a large range of quality among hardware. This can be a problem when ensuring smooth operation for an app. | Mobile devices are very popular giving access to a large client base. However there are multiple different platforms that an app would need to be specifically worked towards. IOS and Android are two examples of different platforms that would need to be optimized in order to have a competitive app. This will also create the need for touch screen compatibility. |
| **Development Tools** | Apple created Swift a programming language built specifically for mac. VS Code has an extension for swift.  Using the RESTful API and dropwizard though we can use HTML and connect to Java in order to be able to serve Mac, Linux, and mobile applications from Windows or any other order. This can be done with a single team however multiple teams will decrease development time but may raise costs. | Linux uses the C programming language. Eclipse, VS code, and Atom all support C. | Microsoft created C# but is compatible with numerous other languages such as C++, F#, R, Java, javascript, and many more. Visual studio IDE was created by Microsoft making it the best IDE for Microsoft app development. | JavaScript can be used for just about any mobile platform. You will also need Swift for IOS. Python has libraries allowing for android development. Java can used for either of these platforms. VS code can be used for both platforms making it a solid IDE choice |

## Recommendations

1. **Operating Platform**:

I would recommend using the Windows platform. The Windows platform has far more support than other platforms making it easier to work in and allowing for multiple type support.

1. **Operating Systems Architectures**: Microsoft uses a Graphical User Interface (GUI) making the platform easier for the average user as opposed to command line in Linux. This means the computer takes input from the mouse by positioning the pointer over an icon and clicking. Windows uses the Win32 or Win64 API a very familiar API for most programmers.
2. **Storage Management**: Windows has the Windows Storage Management API built specifically for storage management and uses C/C++.
3. **Memory Management:** Windows gives each 4/8 gb memory it’s own virtual address space depending on whether you are using the 32 or 64 bit windows (Alvinashcraft, 2022). The virtual address isn’t the actual physical address, the system uses a page table to translate the virtual address to the physical address (Alvinashcraft, 2022). This allows for easy memory access but also adds in some protection to memory.
4. **Distributed Systems and Networks:** Distributed software allows for sharing resources. This makes it far more feasible to optimize an application across multiple different platforms. This increases potential scalability of the application since processing capabilities will be increased. There is the potential for loss of communication between connected computers however due to outages or loss of internet connection.
5. **Security**:

Microsoft has built in security for their platform called Windows Security. This software constantly checks for malware, viruses, and other potential threats to security(Microsoft, 2022). The memory management system also adds some additional security by not giving direct access to physical memory (Alvinashcraft, 2022). Additional security measure should be taken through the app such as username and passwords for each user. This can be accomplished through the Dropwizard framework, which is a collection of libraries specifically for providing common functionality for production ready distributed applications. There is also an abundance of third party security software built specifically for windows, allowing for additional protection. Windows being such a large platform unfortunately has a lot of potential security threats. Other platforms such as Linux and Mac have much fewer threats for this reason. Cross functionally a windows based threat shouldn’t be able to have much effect on a Mac operating system. However, additional security such as third-party software should be used to help decrease the potential risks to end users.

Bibliography

Alvinashcraft. (n.d.). *Scope of allocated memory - win32 apps*. Win32 apps | Microsoft Learn. Retrieved September 18, 2022, from https://learn.microsoft.com/en-us/windows/win32/memory/scope-of-allocated-memory

*Microsoft*. Microsoft Support. (n.d.). Retrieved September 18, 2022, from https://support.microsoft.com/en-us/windows/stay-protected-with-windows-security-2ae0363d-0ada-c064-8b56-6a39afb6a963