

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 11/07/2023 | Paul Kolsut | First design meeting |
| 3.0 | 12/09/2023 | Paul Kolsut | Final design |

**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 Gaming Room company has tasked us with creating a web-based version of the gaming app. The working app is called Draw It or Lose it and it is very similar to the game Win, Lose or Draw. The game renders images from a library and the team’s job are to solve the puzzle. They have 30 seconds to solve it and if not, the other team has a 15 second time limit to guess and solve the puzzle. One of the design problems that we must face is moving the software from the application to web-based application. Other challenges that we will face are unique names for the team, game and only one instance of the game can exist in memory at any given time. To resolve these issues, we will be implementing a singleton design pattern. The purpose of this design pattern is to ensure there is only one instance of the class, and it must be externally accessible. We will implement the singleton design pattern by making the singleton class constructor private. We are going to use Object Oriented Programming and will be implementing multiple classes for each object and the ProgramDriver class with the main() method and will be set to drive the program’s execution.

## Requirements

*The game is supposed to be available online. Game will have the ability to have one or more team involved. Each team will have multiple players. Game and Team names must be unique. Only one instance of the game must exist in the memory.*

## [Design Constraints](#_2et92p0)

Technical Constraint:

* We are going to utilize java language.
* Singleton pattern is going to be used to develop this game.
* We are going to utilize object-oriented programming and multiple classes are going to be created.
* Game should be available for every system since it will be available online.
* Game should be required to work even with slow internet connections.

Business Constraints

* Client needs our team to streamline the development.
* Our Java team is going to work on this project.
* Projects needs to be accomplished on time.
* Projects needs to stay within a budget.
* We are going to utilize images from a large library of stock drawings

## [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)

There are 7 classes ProgramDriver, SingletonTester, Entity, GameService, Game, Team and Player. ProgramDriver has a main method, and it is used to run the software. SingletonTester is just a test class to make sure that singleton design works. We are left with 5 classes that are used to build the game. Game class holds an information about the game, Team class holds an information about the teams and the same thing about Player class it holds the information about the players. Then there is an Entity class that is an abstract class that basically creates an entity, and Game, Team and Player class inherit the information from the entity class. Inheritance is part of Object-Oriented Programming. What it means that the other classes can inherit attributes and methods from the super class. The id and the name from Entity class is used in all three classes Game, Team and Player. GameService, Game, Team and Player class are associated and there is no dependency between these classes. Multiplicity exists between these classes, and it is a zero to many. GameService can exist without any games, but multiple games can use the GameService class. Same thing goes for Game to Team and Team to Player. Team can have 0 players, but multiple player objects can use Team class.

**"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.

## 

## Recommendations

1. **Operating Platform**: I would recommend using Windows as it is the operating software that supports most of the software for server and client-based programs. Since it is an 0operating system that controls 70% of the market it makes most sense to utilize it for the first development. Expertise is easily available since most of the large enterprise companies utilize it.
2. **Operating Systems Architectures**: Windows architecture consists of two layers: User mode and Kernel mode. Kernel mode has complete access to hardware and computer system resources. Kernel consists of Executive, microkernel, kernel mode drivers and hardware abstraction layer. Executive services are divided into various subsystems. They are mainly responsible for memory management, I/O management, thread management, networking, security, and process management. Kernel mode drives interact with hardware devices, HAL is a layer between computer operating system and hardware devices. Microkernel is responsible for CPU. User mode has integral subsystems and environment subsystems. Integral subsystems include fixed system support process (login process and session manager), service processes (task scheduler and print spooler service, security, and user applications. Environment subsystems act as a link between the user mode and the OS kernel function.
3. **Storage Management**: The main types for storage management are hard disk drives and solid-state drives. Since the client is not going to utilize as much space, we can invest in SSD hardware. SSD is much faster than HDD, a standard HDD will read and write at typical 80MB/s to 160 MB/s. SSD drive can read and write at 200 MB/s to 550MB/s using SATA. If client invest in NVMe m.2 standard that offers speeds exceeding 5000 MB/s. SSDs are best when used with a system that needs frequent read-only access to files vs. HDD that are best used with storage of large files. Since we do not want the users to edit images SSD makes more sense.
4. **Memory Management**: Client wants to utilize 200 high-definition images that will have to be displayed to the user to solve the puzzle. When a single user is using the system there should be no issues but what if 100 or 1000 user will want to utilize the game at the same time. Windows server will have to utilize swapping and manage the processes. If the client is going to utilize SSD that read and write speeds very from 200MB/s to 550MB/s. The higher the speed of the SSD the easier will be for the server to manage the memory and utilize it fast and efficiently. We can also compress the images to JPEG or PNG format and that will lower the size of the file. The server will not have to read an 8MB file but a much smaller size. For example, if we can compress the files to 1MB and utilize SSD that has 550MB/s then 550 images can be processed from main memory. We can also utilize a process where a system doesn’t have to bring the same picture from the memory. If another user was already using the picture, then the same picture can be utilized for another game. In this case we do not have to use all 200 pictures only pictures that are being used. We must think about fragmentation problem that might slow down the system. Do we want to do maintenance once a month and clean up the drive and make sure files are not fragmented. Fragmentation can slow down the system and it will affect the game performance.
5. **Distributed Systems and Networks**: Client is going to utilize RESTful API. This technology will allow access for users, admins, and guests to participate in the game. Admins will be able to create games and users will be able to join the game. Guests accounts will be for games who do not want to create logins but want to join the game. RESTful API will allow various platforms to communicate with each other. It won’t matter if the user is using Linux, Mac, or Windows. RESTful API will receive the information from the user and communicate the move to the other users.
6. **Security**: For the game we are going to utilize login and password plus an authenticator. Every user will have to download an authenticator on the phone or computer and after login on the game he/she will have 60 seconds to type in sequence of numbers. When login, password and authenticator match client will login into the server and will be able to start a new game or join an existing game.