Draw It or Loose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	05/28/2025	Christian Geisler	Here I will give my best advice for the design and changes to the game “Draw It or Lose It” to ensure scalability and that it works well across multiple platforms.

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

The Gaming Room is trying to expand their popular Android game, “Draw It or Loose It”, into an expandable web application that is compatible with Mac, Windows and Linux as well as mobile platforms like iOS, all in hopes to reach a much larger audience. This software design provides a solution using object oriented programming principles in Java which leverages design patterns like Singleton and iterator to ensure integrity, scalability and maintainability. The game will also support multiple teams and players per game with unique names for each and a solid management of game instances. This solution lais he foundation for a reliable, multi-platform game that also streamlines future development and maintenance.

Requirements

- The application must allow multiple teams per game and multiple players per team.
- Game and team names must be unique.
- Only one instance of GameService should exist in memory
- Each game, team and player must have unique identifiers.
- The solution must be scalable to support web based and distributed environments to ensure multi-platform access.

Design Constraints

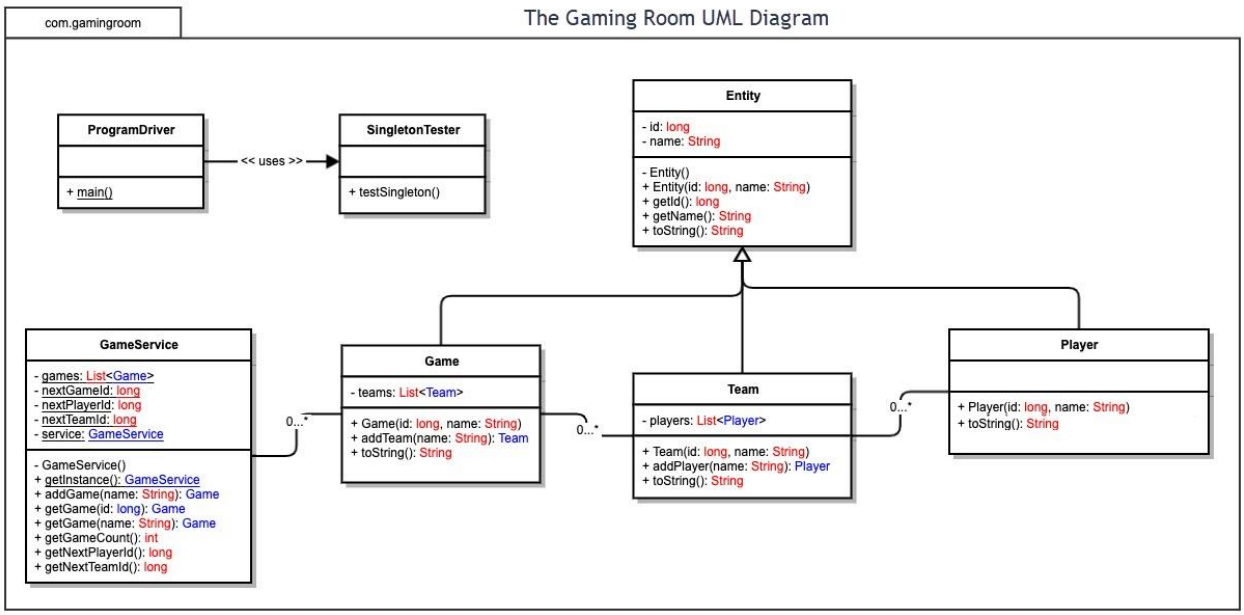
Developing Draw It or Loose It as a web-based application presents several design constraints. The most important thing is making sure everything works well across different operating systems, so the codebase needs to be flexible and easy to scale. There also needs to be a way to make sure there’s only one instance of GameService running at any time. Names of games and players also cannot be reused, so an authentication system will check for duplicates automatically before letting anyone create a new one. Every game, team and player also get a unique ID to keep everything organized.

System Architecture View

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

The UML diagram below lays out how the main parts of the game work together. There is an Entity class at the top that has basic details like ID and name, and all the other important classes like Game, Team and Player all inherit from the entity class so they all get their core features automatically. There is also a Game class that as a list of Teams and each team has a list of Players. The GameSevice class acts like the manager for everything. Its set up as a Singleton, which means there’s only ever one use at a time. When the app adds a new game, team or player it checks through the existing ones to make sure names don’t get duplicated. This design uses key OOP principles like inheritance and encapsulation, and patterns like Singleton and iterator to keep the code clean and easy to read and maintain for future updates.



Evaluation

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	Macs aren't the greatest for server hosting, but they can handle it for development and testing. They are stable with solid security, but using macs for actual server purposes can be more expensive and a lot less common.	Linux is the most popular choice for web servers. It's open source, reliable, and free which makes it perfect for scaling up. It's also a hosting method used by other big tech companies like Google, Facebook, and Netflix.	Windows servers are also powerful and work well with Microsoft tools, but they usually cost more to run.	Mobile devices aren't meant for running servers. Instead, they are only meant to access the app.

Client Side	Making sure the app works on Mac mostly means testing it in Safari and Chrome. Development time and costs aren't much different from other platforms, but it's good to double check things look right on Mac screens.	There aren't a ton of Linux desktop users, but for those that do use it, you want to make sure the app works smoothly in Firefox and Chrome. The main cost is just putting in some extra testing for Firefox.	Most people use Windows, so it's worth spending extra time testing for compatibility in browsers like Edge and Chrome. Supporting Windows just means making sure everything works for a wider range of users.	You have to make sure the app is responsive and easy to use on any phone or tablet. This means extra testing on iOS and Android, but it's worth it for the larger audience.
Development Tools	Java, Eclipse IDE, IntelliJ IDE, and sometimes Xcode if you're working on Mac. But most other standard tools work just fine on Mac.	Java, Eclipse, and IntelliJ, plus all the command-line tools that come with Linux.	Java, Eclipse, IntelliJ, and Visual Studio Code are all popular on Windows. Most major Java tools are compatible with Windows.	For Android, you'd use Android Studio and Java or Kotlin. For iOS, it's Xcode and Swift. But for a web app, you just need to make sure it's optimized for mobile browsers.

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 going with Linux as the main operating system, its open source, reliable and free, which makes it perfect for scaling up and running web apps. Most big tech companies use Linux for hosting because it's stable and works great with Java.

2. **Operating Systems Architectures**: A typical setup would use a Linux server with a Java based backend for the main application logic. This makes it easier to run distributed web applications, connect to databases, and handle requests from users on any device. You could also run this on a cloud server like AWS or Google Cloud for even more flexibility and scaling.

3. **Storage Management**: Using a database like MySQL or PostgreSQL is the best option. These databases are reliable, secure, and easy to scale if the game gets more users. They'll handle storing all the data for games, teams, and players, making it simple to query and update information as needed.

4. **Memory Management**: Linux is really efficient at handling memory for server applications. Java also has built in garbage collection, so it manages memory automatically and helps prevent leaks or crashes.

5. **Distributed Systems and Networks**: To let people play on different platforms, the app should use standard web tools like HTTP/HTTPS and maybe WebSockets for real time updates. Running the backend in the cloud also makes it easy to handle network traffic, deal with outages, and keep everything connected.

6. **Security**: Security is a must, so all data sent between users and the server should be encrypted using HTTPS. User info should also be stored securely in the database, with access controls so only authorized users can see or change data. Regular security updates, input validation, and strong authentication will help keep everything protected across all platforms.