The Gaming Room

CS 230 Project Software Design Template

Version 1.0

Table of Contents

CS 230 Project Software Design Template 1
Table of Contents 2
Document Revision History 2
Executive Summary 3
Design Constraints 3
System Architecture View 3
Domain Model 3
Evaluation 4
Recommendations 6

Document Revision History

Version	Date	Author	Comments
1.0	07/15/21	Victor Ladoano	Environment setup and testing of its functions

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 wants to set up the game in a web-based format and serve multiple platforms. The client wishes that the game has one or more teams, with multiple players assigned to each team. No game, team or player should have the same name and the game should have only one instance in memory at any given time. Creating games, teams and players in lists, allows for a pool of games, teams and players to be assigned when called. Building a check for name using the iterator pattern prior to adding a new game, team or player enables each to have a unique name and using a singleton pattern design, allows for only one instance of game to be in memory.

Design Constraints

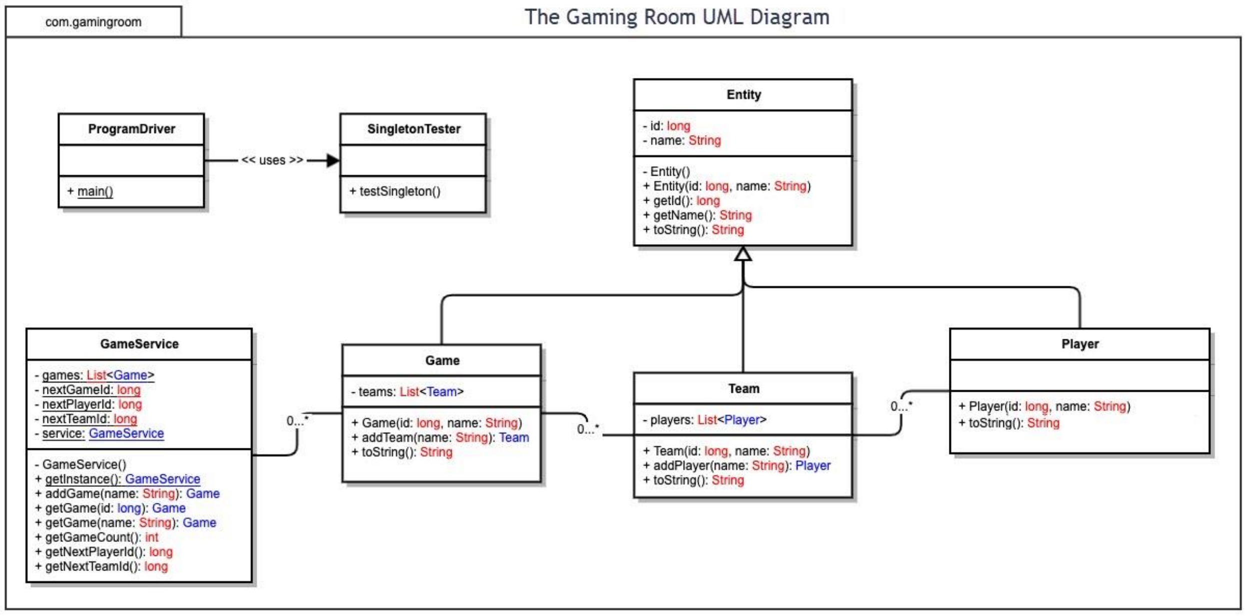
*For the game to be able to support multiple platforms, it is essential to have a good API for communication between client and server.
*As the game consists of rendering images at each level, computer processing can be a problem if it needs to have all pictures in memory. Therefore, design should account for memory management.

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 com.gamingroom package has a base class (parent) with two private variables, one private empty constructor to avoid instance of empty object, a public constructor and 3 public methods to retrieve and display object information (getters). The Game, Team and Player classes inherit from the base class. Game and Team classes add a private list of the next association class, a constructor, a method to add into its private list and an override display method.
The program also contains a GameService class that relates in association with all Entity's child classes. Only one instance of this class will exists at any time in this program. The ProgramDriver class has main() to drive the program and test functionality. The SingletonTester class simply uses main() to test the singleton instance.



<u>Evaluation</u>

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	Its powerful UNIX-based OS makes it a complete solution for webhosting. OpenSSL is used for SSL and TLS protocols on authentication and security. The cost for licensing is cheap but Mac hardware is the most expensive.	Linux is a popular choice for wed-based applications. It is light and stable. No memory leaks. Licensing is free but support will start adding costs	Windows Servers are a robust solution to host a web-based app. Support most development languages. Licensing per core can be expensive depending on the hardware.	Although a portable solution, Mobile Devices don't have the hardware power a server needs to handle hosting a web-based application.
Client Side	MacOS is a platform that has gain popularity for its ease use. Apple offers support and expertise that makes it easy to maintain development of a web-based app.	This platform is not a popular client-side solution because it requires some user expertise for use. While a free platform, support isn't included.	Windows clients are still a popular platform for client-side development. The cost is justified by the available tech support and the expertise that makes it quicker to develop and debug.	Mobile devices are the most popular client option for web-based applications. Their popularity makes it cost efficient to develop for, easy and fast. Many developers now have the expertise to develop for this platform.
Development Tools	A team of developers may be used for different requirements. Being a web-based app, JavaScript would be the best to develop. For a more Mac native app, Objective-C and C++ would be ideal.	.NET and Java are two languages popular for development. C/C++ are also another choice. A team may or may not be used for development on this platform, these are also free solutions for development.	Javascript and HTML are the most popular language for this environment. Although it also supports a vast number of other languages.	Android studio and Swift would be popular choices to develop for this platform. Teams would be used to develop front and back end using different languages and tools.

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 a Windows server environment be chosen to run this application because of its easy and straight forward manner to set up a Web Server. The vast options of IDE's and pre-defined language libraries, makes it easy to expand the game within this environment.

2. **Operating Systems Architectures**: The system offers and should be setup with Failover Cluster so availability of the game is addressed even if there's an issue with one of the servers being used to offer the Web Server services. The architecture should also include Load Balance so the requests to the web server is evenly split through the cluster of servers setup to provide the web service.

3. **Storage Management**: My recommendation for storage management is the use of AWS. Using something like Netapp Cloud Volume ONTAP gives the ability for failover clustering to be implemented for Windows in the cloud. More space and resources can be allocated easily without the need to purchase new hardware for expansion.

4. **Memory Management**: The operating system should be setup with Paging Memory Management to give the required applications that will need to be run in the system, a clear logical memory and avoid segmented physical memory. Page size should be large enough to register crash dump.

5. **Distributed Systems and Networks**: The game will use API infrastructure in its code to be a distributed software among various operating system clients through the web. The failover cluster will be an essential feature to provide availability of the game during outages or system's failure.

6. **Security**: For security the server should have appropriate Secure Socket Layer certificates for secure connection and authentication of the user. For extended security of user authentication, a two-factor authentication process should be set up for further verification of the identity of the user.