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 3
Recommendations 5

Document Revision History

Version	Date	Author	Comments
1.0	<02.18.2021>	Juergen Pfau	Added information related to the software 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

The Gaming Room needs a web-based-game that can serve multiple platforms based on its popular Android game, Draw It or Lose It. This game would be like Win, Lose or Draw, a popular game show from the 1980s.

Design Constraints

- Run Game on multiple platforms
- Synchronization of multipeople on platform
- Check unique team identity
- Check there is one instance at a time
-

  These constraints are within the development team to produce within. This needs to run on multiple platform originating from android code and porting this to work across multi platforms which may require a larger dev team.

System Architecture View

Domain Model

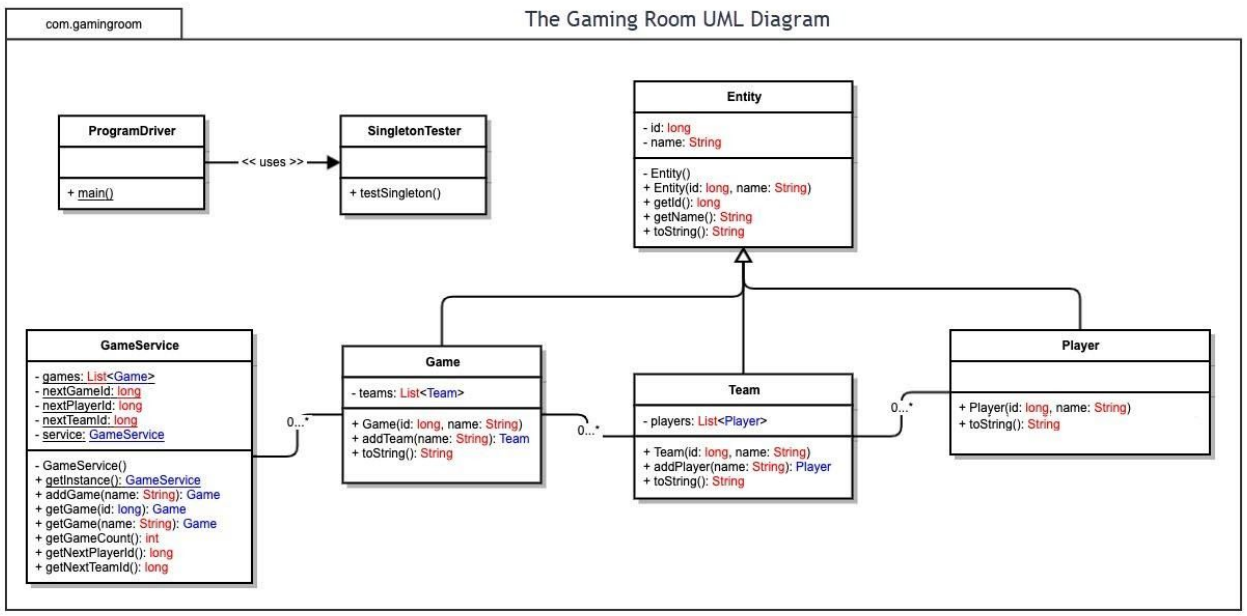
Entity class is inherited by the following classes directly:

Team, player, game.

Game service is connected to both game and team which both reference the player class. The aggregation pattern allows all of these classes to reference oneanother. This includes our singleton tester(Which watches for instances of an operation) and our driver class. The driver class is in a way like a main class. This is where functions are executed in the program. The singleton class insures there is only a single instance at a time of this operation.

- The relationship between the team, game and player task is they all inherit from Entity so that code isn't written redundantly.

  ProgramDriver and Singleton teter allow interconnection that will allow main more access.



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	Mac does have Mac OS X server available for use. Per Apple's website, Mac OS X Server is only $20, so it would be inexpensive to implement. Mac, however, is not as popular as say Linux or Windows for performing these tasks.	Linux is interesting here as it has many distributions that have server capabilities. Linux Server would be low-cost, and open source (which provides a lot of resources). Not many users are savvy with Linux, so you would need someone who is	Windows offers Windows Server. Looking at Microsoft's website, it might be costly to implement, but it is fully functioning. Windows is likely the most used operating system, so finding users to operate Windows server would be a lot easier.	Given that mobile devices do not necessarily have the power that computers do, hosting a fully-fledged server on one may not be the best option compared to computers. Running servers on mobile devices is the most advantageous in terms of cost, as there is little to

		familiar with Linux running the server.		none to get one started.
Client Side	Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Mac would need less time and someone who does not have as much experience with Mac would need more time.	Cost would be low (if there even is a cost) with Linux, as it is open source. Maximum time and experience would be necessary, as Linux is not commonly used and you would need someone who is apt with Linux and allow them time to work, as Linux can be difficult even for someone with experience.	Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Windows would need less time and someone who does not have as much experience with Windows would need more time.	Cost would not be too much of an issue with Mobile devices. Experience may not be too much of an issue, as mobile devices can be easier to work with. More time would be needed, as there are multiple operating systems and multiple mobile devices that would need to be worked on.
Development Tools	Swift would be the more common language used to write applications for Mac. There are multiple IDEs that can be used for Swift, such as Atom (which can be used on multiple operating systems) and Xcode. The team working on the Mac version of the game would also be working on the iPhone/iOS version of the game, as the languages are practically the same. Development tools usually wouldn't cost anything, as they are usually	Eclipse and Atom are commonly used IDEs on Linux. Eclipse is primarily used for Java, although it can support other languages like C++. Atom can also be used for developing in multiple languages. The team working on the Linux version of the game would also be able to work on the Windows and Android versions of the game. Development tools usually wouldn't cost anything, as they are usually	Eclipse and Visual Studio are popular IDEs for Windows. Visual Studio can be used for developing in HTML, C# and JavaScript among others. Eclipse is primarily used for Java, although it can support other languages like C++. The team working on the Windows version of the game would also be able to work on the Linux and Android versions of the game. Development tools usually wouldn't cost	For iPhones, the development tools are like those for Mac, and iOS apps are typically written in Swift, though iOS and macOS are different in terms of appearance and functionality. Android apps are normally written using Kotlin, Java, and C++ languages. Android apps can be written in Eclipse, Android Studio. Developers for iPhone would probably come from a team of Mac, and Android developers would come from the team developing Windows and Linux.

	downloadable directly from their respective developers.	downloadable directly from their respective developers.	anything, as they are usually downloadable directly from their respective developers.	Development tools usually wouldn’t cost anything, as they are usually downloadable directly from their respective developers.

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**: Based upon all my research, I believe that Windows would be the best operating platform to use, as it is the most widely used operating platform, as well as the one that most people have a working knowledge of. There is an abundance of IDEs that can be used with Windows, and the total cost to utilize it is typically lower.

2. **Operating Systems Architectures**: The Windows architecture allows for applications to utilize the platform's kernel processes without directly affecting those processes. In other words, applications can utilize the power of Windows to have a GUI/window set up, access to memory and other vital processes that make the application without inadvertently affecting the processes that make the operating platform work.

3. **Storage Management**: this is covered from either modular storage to cloud storage in Windows. Windows Disk manager allows for easy change and adaptation of local disk space as well as Windows cloud storage programs.

4. **Memory Management**:
   There is a lot of different ways to go about this. I think the most easy would be not to reinvent the wheel. A game engine would suffice and it would eliminate the need to write unclean or redundant code while taking advantage of enterprise-level code for this type of project.

5. **Distributed Systems and Networks**: A client-server distributing system will be utilized here, as we will have each client application depend on the single server application for our game, so that each client application can be developed to that client's system's strengths. A strong server network would also be needed, as this game's success depends on multiple clients connecting to a single server to play one game altogether.

6. **Security**:
   Windows Defender is the build in security measure now pushed through window. This is massively beneficial now for any windows system. Encrypting all data being pushed would help forward security.