Draw It or Lose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	<05/15/23>	Bryce Jensen	<Brief description of changes in this revision>

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

Our client, The Gaming Room, wants to expand Draw It or Lose It, their current Android game, to several platforms by porting it to a web-based game. The game should allow for one or more teams with multiple players. A singleton pattern will need to be used to ensure only one instance of the game is running at a time and that the game and team names are not reused. When picking a name, the system will alert the players that a name is currently being used.

Requirements

< Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client's business and technical requirements in a clear and concise manner.>*

- A game will have the ability to have one or more teams involved.
- Each team will have multiple players assigned to it.
- Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
- Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

Design Constraints

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

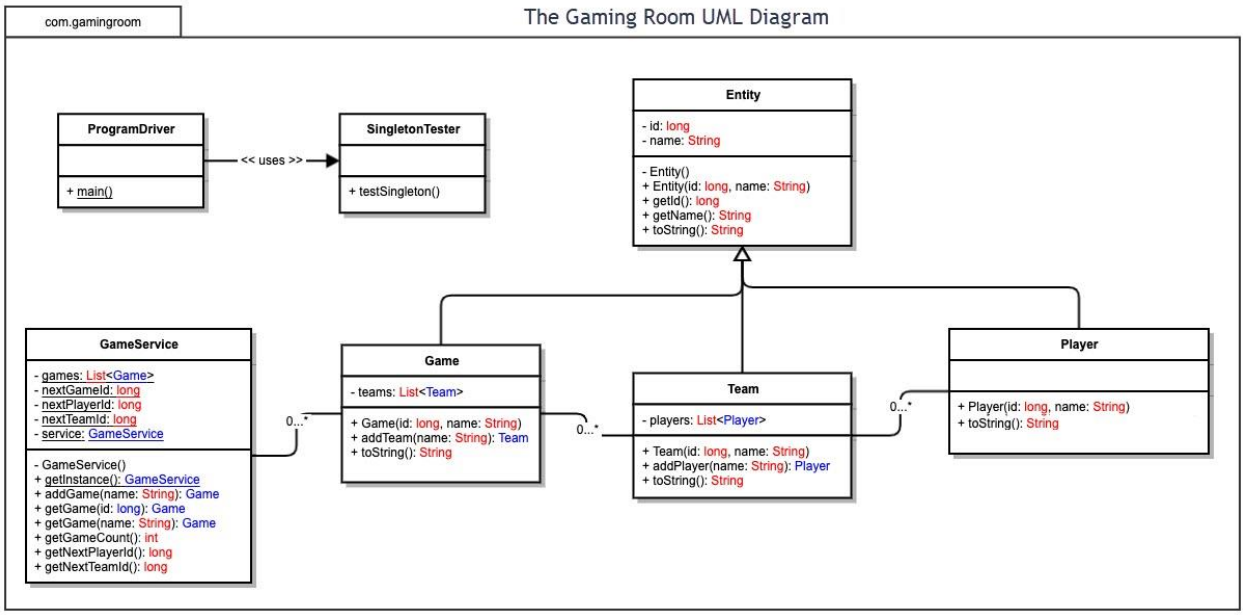
- The languages to be used will be Javascript and HTML5 for the website; the game will be written in Java.
- Singleton patterns must be used for game instance, Team name, and User name checking. Only one instance of each of these may be used at a time.
- Being a web-based game, it should be able to run on most machines. It should be kept "light" to prevent issues upon release.
- No deadline was given in the initial communique. Will follow up with The Gaming Room to determine what that is. Expect an update soon.
-

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

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>



There are 7 classes. `ProgramDriver`, which `<<uses>>` the `SingletonTester` class. Separately from those two, although all linked in some way: `GameService`, `Game`, `Team`, `Player`, and `Entity` classes.

The `GameService` class uses a singleton design pattern, this is so that only one instance of the class can exist at a time. This is apparent once noticing that the `GameService` class has a private constructor. Only the `getInstance` method can create a `GameService` class, but it would check whether an instance already exists.

Another important method in the `GameService` class is the `addGame` method. This checks – using an iterator for the `games` list – for game objects of the same name. If nothing is found, then a `game` object is added to the `games` list. The `Game` class contains an `addTeam` method and the `Team` class an `addPlayer` method. This acts the same as the `addGame` method above. Using an iterator, it checks for objects of the same name before creating them in the `team` and `player` lists.

The `Game`, `Team`, and `Player` classes are all subclasses of the `Entity` class. This means that they inherit the characteristics of the `Entity` class. The default constructor for the `Entity` class is private. Only the overloaded constructors can be accessed for use.

Several techniques for OOP (object-oriented programming) can be seen in this UML model of this program. Inheritance of the `Entity` class. The `Entity` class also has signs of Polymorphism when overloaded constructors and methods are used. Encapsulation can be observed by setting certain attributes to private only allowing access to them through specific (public) methods.

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	<Evaluate Mac for its characteristics, advantages, and weaknesses for hosting a web-based software application.>	<Evaluate Linux for its characteristics, advantages, and weaknesses for hosting a web-based software application.>	<Evaluate Windows for its characteristics, advantages, and weaknesses for hosting a web-based software application.>	Most mobile devices can access a server just as well as most other devices, with the only exception being that they may have a less stable connection as they can't be wired to a network. Instead, relying on a wireless connection. Also, their mobile browsers may not have the capability for a long-term connection to a server.
Client Side	<Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.>	<Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.>	<Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.>	<Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.>

Development Tools	<Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mac.>	<Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Linux.>	<Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Windows.>	<Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.>

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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

6. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>