Draw It or Lose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
3.0	02/19/2021	Kari L. Cheslock	Software Design Template for Draw It or Lose It Web-Based Application

Executive Summary

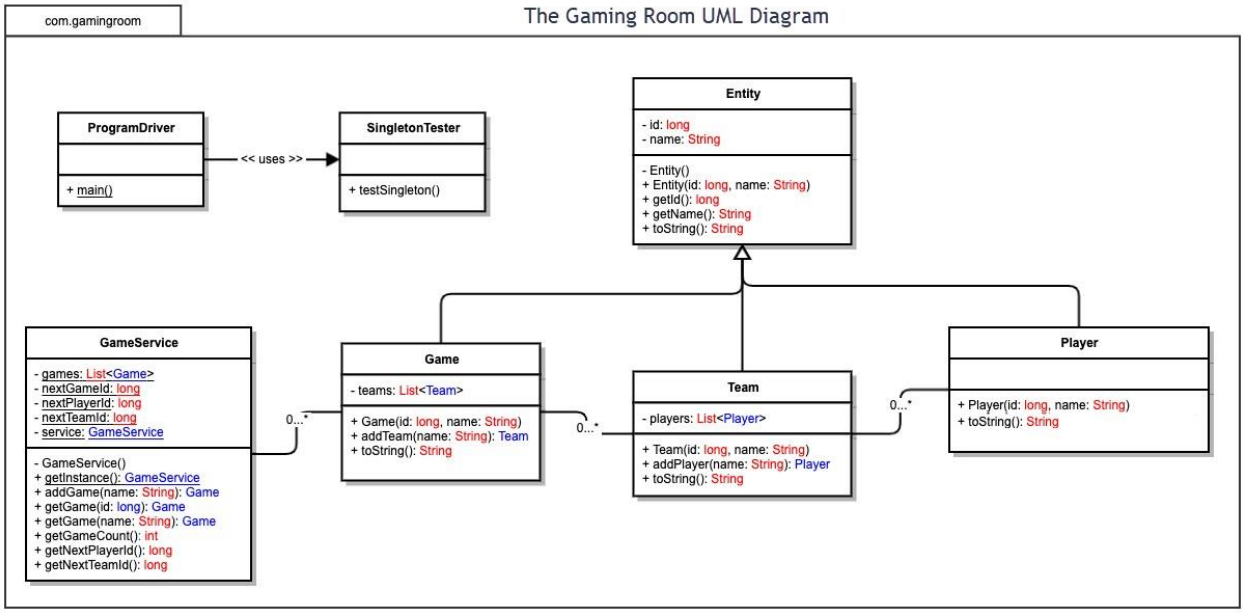
The Gaming Room currently has an Android based game application called Draw It or Lose It. They would like to expand the availability of this game to a web-based application that will be accessible across multiple platforms to increase their customer base. The web-based application should be designed such that only one instance of the game should exist in the memory at any given time. Each game should allow one or more teams and each team should allow multiple players. All game and team names should be unique to preclude multiple instances of the game in memory. This document will detail the design and implementation of the new web-based application.

Design Constraints

- The server will need appropriate storage space for the library of images.
- The server will need to handle scalability of multiple games, teams, and players.
- The server will need to manage secure login requests.
- The app will need to handle administration of multiple users with unique logins.
- The development and administration should stay within the budget of the customer.
- The app will need a mechanism for managing security.
- Appropriate cross-platform tools will need to be implemented for the application development.

Domain Model

The Gaming Room java package will be called com.gamingroom and will start with a class called Entity as the base class with id as a long type variable and name as a string type variable. Having an Entity class as the base class will minimize the code needed for the inherited classes. The Game class, Team class, and Player class will inherit from the Entity class and have methods to add new games, teams, and players, respectively. The user will interact with the Game, Team, and Player classes to add new games, teams, and players without interacting with the base Entity class, known as encapsulation. The implementation of the Entity class is abstracted away from the user. Each game can have multiple teams and each team can have multiple players. Only one instance of the Game class can exist in memory at any given time. Therefore, games, teams, and players will be created with a unique identifier and a name. The GameService class will be used to create unique instances of the Game class. The user will have the option of creating new games, teams, and players through the interface and the appropriate classes will be accessed based on the data type, known as polymorphism. The application will use the implementation of these classes to create unique instances of players, teams, and games.


com.gamingroom
The Gaming Room UML Diagram
ProgramDriver
+ main()
<< uses >>
SingletonTester
+ testSingleton()
Entity
- id: long
- name: String
- Entity()
+ Entity(id: long, name: String)
+ getId(): long
+ getName(): String
+ toString(): String
GameService
- games: List<Game>
- nextGameId: long
- nextPlayerId: long
- nextTeamId: long
- service: GameService
- GameService()
+ getInstance(): GameService
+ addGame(name: String): Game
+ getGame(id: long): Game
+ getGame(name: String): Game
+ getGameCount(): int
+ getNextPlayerId(): long
+ getNextTeamId(): long
0...*
Game
- teams: List<Team>
+ Game(id: long, name: String)
+ addTeam(name: String): Team
+ toString(): String
0...*
Team
- players: List<Player>
+ Team(id: long, name: String)
+ addPlayer(name: String): Player
+ toString(): String
0...*
Player
+ Player(id: long, name: String)
+ toString(): String

Evaluation

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	-Pro: Higher security -Pro: Less susceptible to error -Pro: Based on Unix -Pro: High performance -Con: High cost -Con: No control over OS updates	-Pro: Low cost and low maintenance -Pro: Control over updates -Pro: Stable and reliable, less prone to error -Pro: Easily scalable -Con: Lower security -Con: Requires knowledge of command line commands	-Pro: Flexible price points, cost effective -Pro: Many tools available unique to Windows -Pro: High familiarity, large support community -Con: No control over OS update -Con: More susceptible to viruses, crashes, etc. -Con: Must be kept up to date with security patches, etc.	-Pro: Low cost -Con: Less functionality -Con: Shorter lifespan of devices -Con: Potential compatibility issues between desktop browser and mobile browser -Con: Wireless connection only -Con: Less security
Client Side	-Pro: Many available browsers including Safari, Firefox, and Chrome -Pro: Systems are more robust, require less configuration -Pro: Higher quality user interface -Con: Systems are more expensive	-Pro: Open-source software -Pro: Many available browsers including Firefox and Chrome -Con: Less familiarity among users	-Pro: Systems are less expensive -Pro: Many available browsers including Chrome, Firefox, and Edge -Pro: Highly familiar OS for users -Pro: Easy to configure -Con: Must be kept up to date with security patches, etc.	-Pro: Devices are less expensive -Pro: Portable devices -Con: Less functionality

Development Tools	-Pro: NetBeans or Eclipse IDE's (non-platform specific) -Pro: Java programming language (server), HTML & JavaScript (client) -Pro: Dropwizard framework in Java	-Pro: Command line compiling and app execution -Pro: NetBeans IDE -Pro: Java programming language (server), HTML & JavaScript (client) -Pro: Dropwizard framework in Java	-Pro: NetBeans or Eclipse IDE's (non-platform specific) -Pro: Java programming language (server), HTML & JavaScript (client) -Pro: Dropwizard framework in Java	-Pro: Cross-platform development tools (numerous available with flexible price points) -Pro: Java, HTML, and JavaScript programming languages -Con: XCode or AppCode for iOS development only -Con: Android Studio for Android development only

Recommendations

1. **Operating Platform**:

Based on the above evaluation of the different operating systems, Linux is the recommended operating platform for this application due to its low cost, low maintenance, and high stability.

2. **Operating Systems Architectures**:

The basic architecture of a Linux system consists of the hardware, the kernel, the shell (also known as the command line interface, or CLI), and the various applications and utilities. The hardware is the physical equipment including all peripheral devices. The kernel is the core of the operating system that interacts directly with the hardware. The shell serves as the interface between the user and the kernel and accepts user input from the command line. The utilities are the various programs available to the user. Linux is open source, highly customizable, and highly scalable. Linux is also stable, reliable and less prone to errors. The Linux file system structure is hierarchical and begins at the root directory. The root directory contains all other file directories on the system including, but not limited to, the bin directory which contains all essential user binaries (programs), the boot directory which contains files needed to boot the system, and the home directory which contains a file for each user. Access to these directories is also highly customizable through setting read, write, and execute permissions that can be set by an administrator.

3. **Storage Management**:

Storage management on Linux can be accomplished by using disk partitioning if the system is going to be maintained locally or on the cloud by purchasing the appropriate amount of storage. For this application, cloud storage will be preferable since it allows for expansion as use of the application grows. At a minimum, cloud storage should be used for backing up any local systems for preservation of data. For this application, storage will be used to store the image files as well as the lists of games, teams, and players, both historic and currently in use. The number of games, teams, and players will increase over time, so the storage capacity needs to be able to accommodate for this, which is why cloud storage is preferable since its capacity is essentially unlimited. Also, cloud storage provides highly efficient transfer rates for retrieving stored data, which will be important for operation of the application.

4. **Memory Management**:

Memory in Linux is divided into basic units called pages, which are basically small chunks of contiguous memory. Memory management in Linux includes managing of virtual memory and memory allocation for kernel structures and user programs, among others, by managing page allocation. Memory management is handled primarily through the /proc subdirectory and is highly customizable. Virtual memory is handled through a variety of techniques including demand paging, where only pages currently being used by the user are loaded, and swapping, where unused pages are swapped for pages that need to be used. Memory is also managed in Linux using caches which hold frequently accessed pages. For the Draw It or Lose It application, memory will also be managed through Java, the program the application will be written in, which has a built-in garbage collector that automatically allocates and deallocates memory used by the program. Java can also be programmed to use memory caches to temporarily store data. For this application, the images being used in the game could be loaded into a cache for optimal program execution and to prevent latency.

5. **Distributed Systems and Networks**:

The distributed system for this application will follow the client-server model which involves communication between a client and a server in which a client contacts a server for data and a server responds to the request. These clients and servers will be connected to a network over which the communication will occur. This type of system will work well for the Draw It or Lose It application since it will accommodate communications between various platforms and from various devices, all of which will be in different physical locations. The scalability across a client-server distributed system is unlimited, meaning that any number of clients (or servers) could be added. One disadvantage of this type of system, however, is that a network outage would disrupt service to the user. This system also relies on the user having a reliable network connection on their own devices.

6. **Security**:

Linux offers many security features starting with the basics of authenticating user and password credentials and setting permissions on files for reading, writing, and executing. Linux also offers comprehensive network security and supports many protocols at the network layer and includes firewalling features as well as access control rules for IPv4 and IPv6. Linux also offers several API's and utilities that can be implemented and customized depending on the particular security needs of the system. These include utilities for adding checksums for data verification, utilities for adding encryption, and utilities for secure remote access through a secure shell (SSH) service. The fact that Linux is less susceptible to viruses and malware also adds an additional layer of security and protection for the system. Also, since Linux is open source, all security patches will also be available for free.