

Game name iterator

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 10/04/2020 | Devayn Johnston | Re-modeled classes to extend from class Entity where they then performed their own function. |

**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**

We are to consolidate certain aspects/identifiers of our previous design submission into one class which then extends into different instances of a class. We are to make a class entity which is used to pull certain identifiers i.e, the name and id fields for team, player, and game classes. With class inheritance the program design will be much more consolidated and easier to read. The software design will also filter for unique game,player, and team names.

**Design Constraints**

A design constraint to first and foremost think of would be the preferred operating system. Would we be doing a mobile OS or a desktop OS? Are the users tech savvy or newbies? Gamers or casuals. How complicated of an application will we have? If we do cross platform multiple languages will need to be implemented. What is the budget for our project? What software will we be using to develope the application?

**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 demonstrates our Main class which pulls all classes into function. The main class utilizes the singleton tester class so it will print instances. For shared attributes and get methods for the Team, Game, and Player class we created the class entity. This allows us to reduce total amount of code by extending entity into each class which would need the similar attributes. Entity would be considered an abstract class. The Game service class associates with each of the Game, team, and player classes as they do one another as well. Each composed of their own unique methods and attributes.



**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** | Commands are flexible to configure, acces, and make changes with | More cost friendly as Compared to Mac. | More software availability | Server must be stationary. Specifications are better for other devices. |
| **Client Side** | Expertise is lower, cost is similar to windows. | High amount of knowledge required. Low cost. | Minimum expertise/time. Cost same as mac. | Flexible to clients/developers for updates. More difficult to implement. |
| **Development Tools** | Uses common language html/css/js and supports libraries for front end. Uses gen. purpose languages like python,java,php, ruby on rails. | Same as previous.  Uses pyCharm, eclipse, github. | Same as previous. | Same as previous |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**: My recommendation for an operating system for the future of Draw It or Lose It would be windows.
* **Operating Systems Architectures**: The operating systems architecture is made up of 2 layers, user mode and kernel mode. User mode has limited access while the kernel mode is unrestricitive.
* **Storage Management**: Microsoft OneDrive would be best utilized with a windows operating system in terms of its storage management.
* **Memory Management**: For the software each instance of a particular game will need to be stored. Since the software will filter for unique names storage must be implemented. File mapping will allow the operating system to share data. File mapping also allows for changes to be made within a file and once that change is made it is updated. Hence our redesign of the project.
* **Distributed Systems and Networks**: Distributed software systems allow for multiples of components to communicate and coordinateactions and appear as a single system to an end-user. A client server for example could be used when a client wishes to access the software application and when the client makes changes they can be implemented as permanent on the server. This means any changes to the application could be accessed from multiple access points. Worries may arrise in terms of latency which can make for incosistencies of retrieval, availability, and consistency.

**Security**: Making sure that you access the application from a secure browser. Only allowing cookies and such stored data based on certain websites. Maintaining a password which is non relative to yourself. Keeping email addresses and acces points of the software secure. Don't allow unknown users to make changes to the application. Windows itself has it's own security system pre-installed , but an upgrade would not be bad.