

Draw It Game

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

**[CS 230 Project Software Design Template](#bookmark)**1

**[Table of Contents](#bookmark1) 2**

**[Document Revision History](#bookmark2) 2**

**[Executive Summary](#bookmark3) 3**

**[Requirements](#bookmark4) 3**

**[Design Constraints](#bookmark5) 3**

**[System Architecture View](#bookmark6) 3**

**[Domain Model](#bookmark7) 3**

**[Evaluation](#bookmark8) 4**

**[Recommendations](#bookmark9) 5**

## [Document Revision History](#bookmark10)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2.3 | 04/19/2025 | Marc Ventolora | Final submission for Project Three |

**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](#bookmark11)

Draw It or Fail It Players take turns drawing a phrase that is allocated at random while other players attempt to guess it in this multiplayer drawing and guessing game. Creative Technology Solutions (CTS), the customer, wishes to use scalable, reusable software architecture to extend the game beyond its present Android environment to other platforms. In order to encourage maintainability and cross-platform compatibility, the suggested method is creating a Java-based prototype utilizing object-oriented principles and design patterns.

## Requirements

The client's main business needs are to allow multiplayer gameplay, keep the game responsive and interesting, and expand the game to operate on platforms other than Android. Using object-oriented programming techniques, writing scalable modular code, and putting in place suitable security and data management features are examples of technical requirements. Network connection between clients and a server must be supported by the program.

## [Design Constraints](#bookmark12)

There are restrictions because the program needs to be distributed and web-based. It must make use of a cross-platform compatible language and framework. Java is selected due to its portability through the JVM. Synchronization and session management are essential since the system must support multiple users at once. It should also take into account the necessity for an intuitive user interface across a range of screen sizes, as well as the restrictions of mobile devices, such as slower processors and lesser memory. Due to security restrictions, user data must be safeguarded both during storage and transfer.

## [System Architecture View](#bookmark13)

## [Domain Model](#bookmark14)

The main elements of the Draw It or Lose It game are depicted in the UML class diagram. In addition to containing references to Player objects, a WordBank for storing drawing prompts, and a Scoreboard for keeping track of points, the GameRoom class controls the overall flow. Every player has a role (guesser or drawer), name, and score. Each round's word assignment and timing are controlled by DrawingRound. The design reflects object-oriented concepts including polymorphism (e.g., player actions), inheritance (e.g., GameEntity as a potential superclass), and encapsulation (each class controls its own data). Scalability and code reuse are supported by this framework.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#bookmark15)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Java-based apps can be hosted on Macs with Jetty or Apache Tomcat. However, it might not be as suitable for hosting production settings because to its high cost and limited server-side support. | Linux's performance, stability, and broad support for Java make it the perfect choice for server-side hosting. It is very adaptable and economical thanks to open-source technologies and community assistance. | Windows allows Java hosting with Apache Tomcat or IIS with plugins. Although it is easier to use, Linux is usually more effective for large-scale deployments. | low processing power and memory, mobile devices are not commonly employed for server-side hosting. |
| **Client Side** | Testing for graphics compatibility and interaction models is necessary when creating client-side support for Macs, but Java's cross-platform features make development simple. | Java has good support for Linux clients; however, testing in various desktop contexts is required. The OpenJDK offers broad compatibility. | most users are familiar with the platform and Windows has good Java support, testing and support are made simpler. | While iOS support is restricted in the absence of third-party frameworks like RoboVM, Java can run natively on Android. needs the user interface to be adjusted for different screen sizes. |
| **Development Tools** | Eclipse and IntelliJ IDEA both function nicely on Macs. Supported terminal tools include Maven and Gradle. | Linux is a good platform for using tools like Eclipse, IntelliJ, and NetBeans. Git and command-line tools are supported. | All of the popular IDEs, including IntelliJ and NetBeans, are supported by Windows. Java plugins for Visual Studio Code are also feasible. | The greatest tool for Android development is Android Studio. Due to the lack of native Java support, iOS development is more challenging and calls for solutions like Xamarin or Gluon. |

## Recommendations

The following architectural suggestions are put forth in order to best serve the objectives of The Gaming Room and Creative Technology Solutions (CTS). These are designed to satisfy the cross-platform, scalability, and performance needs for bringing the Draw It or Lose It game outside of Android.

1. **Operating Platform**:Linux is suggested as the best operating system for this project because of its excellent performance, adaptability, and affordability. It provides good compatibility with Java-based applications and is extensively supported in server contexts. Furthermore, because Linux is open-source, it offers a great deal of control and customization, which is crucial for the deployment of scalable gaming servers. It is a great option for both development and production contexts because of its dependability and community support.
2. **Operating Systems Architectures**:A Linux-based microservices-based architecture is suggested to guarantee maintainability, scalability, and modularity. The system may be divided into smaller, independently deployable services thanks to this architecture, and each one is in charge of different game features including user authentication, gaming sessions, and scoring. Java-based frameworks like Spring Boot will help the development of these services, and RESTful APIs will make it easier for components to communicate with one another. This strategy supports future development and feature extension by guaranteeing that upgrades may be done without affecting the system as a whole.
3. **Storage Management**:Lightweight and scalable databases like MySQL or SQLite are suggested for data storage. These systems are cross-platform compatible and ideal for Java applications. They will effectively handle session information, game states, scores, and user data. While SQLite is a good option for lightweight or mobile installations, MySQL offers great support for multiple users at once. Both choices guarantee that the program can grow and function well under different loads.
4. **Memory Management**:The automatic memory management offered by Java's integrated garbage collection techniques lowers the possibility of memory leaks and increases system dependability. The game can operate smoothly on a range of devices with varying hardware capabilities thanks to the Java Virtual Machine's (JVM) ability to be adjusted to change memory allocation depending on the client or server environment. This guarantees peak performance on low-resource mobile devices as well as high-end servers.
5. **Distributed Systems and Networks**:It is advised to use a client-server architecture with dependable communication made possible by Java networking frameworks and tools like Spring Boot. The clients, which operate on various platforms, will connect via socket connections or HTTP requests, while the server will manage users and game logic. Techniques like session persistence and asynchronous messaging will be used to guarantee real-time responsiveness. This architecture guarantees constant synchronization across all devices and facilitates multiplayer functionality.
6. **Security**: The first priority is security. Strong libraries for implementing SSL/TLS-based secure communications are available in Java. To safeguard user data and game state, HTTPS will be used to encrypt all data sent between the client and server. We'll incorporate extra security features like input validation, safe authentication techniques, and defense against SQL injection and cross-site scripting. Both user data and system integrity will be protected by these procedures.