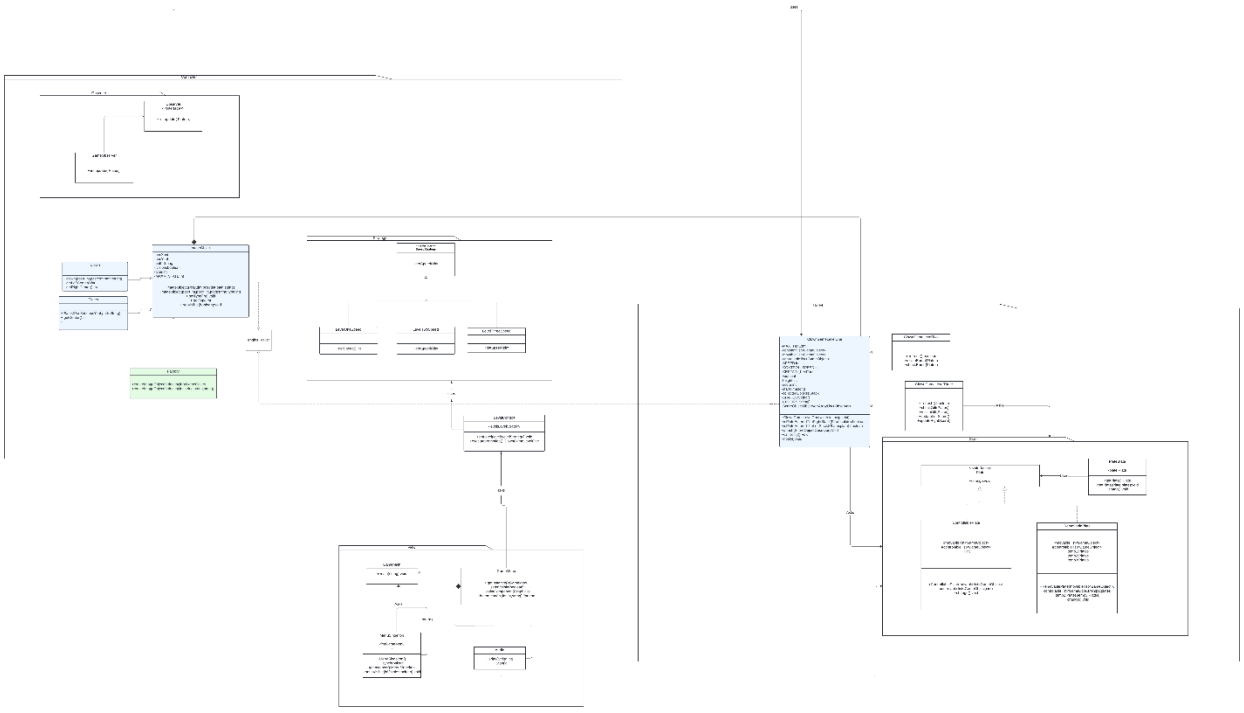
Programming 2

Final Project

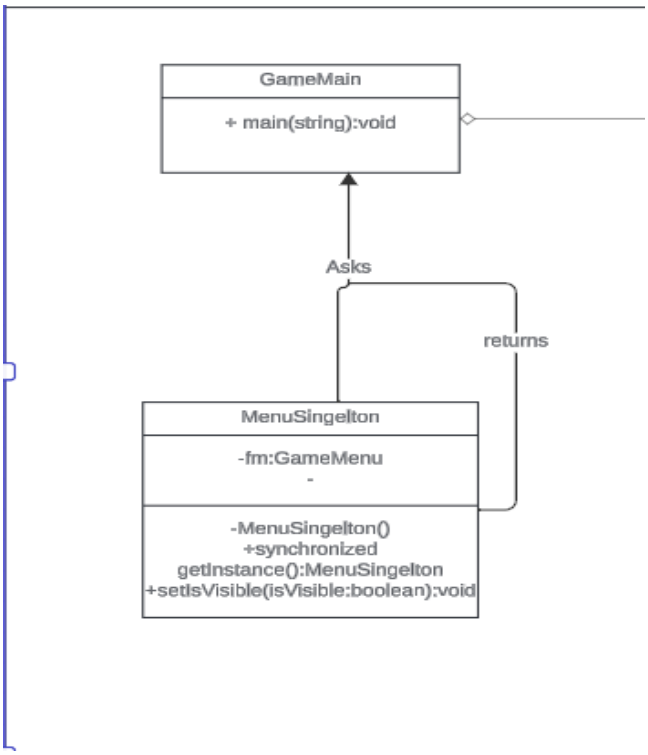
- Tasneem Yousry Mohammed 8013 Group 3 Section 2
- Nada Hanafi Ahmed 8099 Group 1 Section 2
- Reem Mohammed Ali 8388 Group 2 Section 2
- Jowayria Alaa El-din Ibrahim 8328 Group 1 Section 2

UML

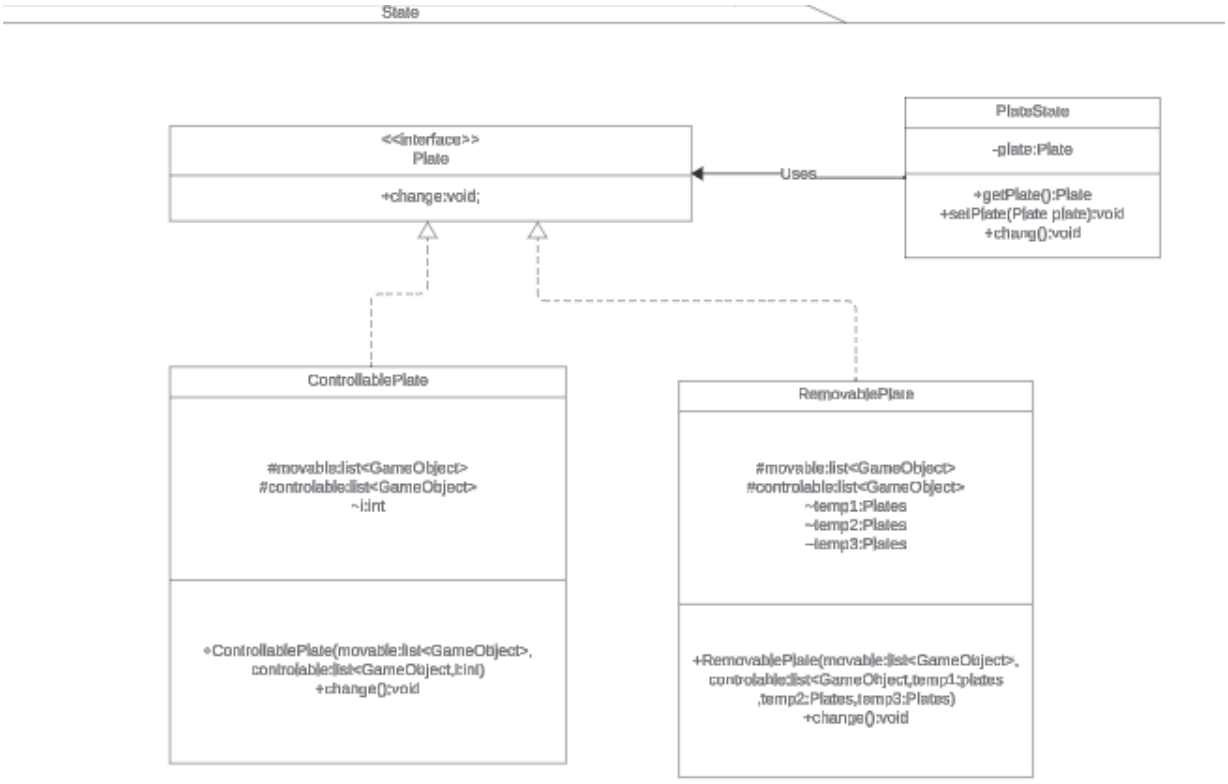


Design Patterns

1. Singleton: The Singleton pattern is a crucial creational design pattern implemented in the GameMenu class. This pattern ensures that only one instance of the GameMenu class is created, promoting efficient resource utilization and maintaining a singular point of control over menu-related functionalities

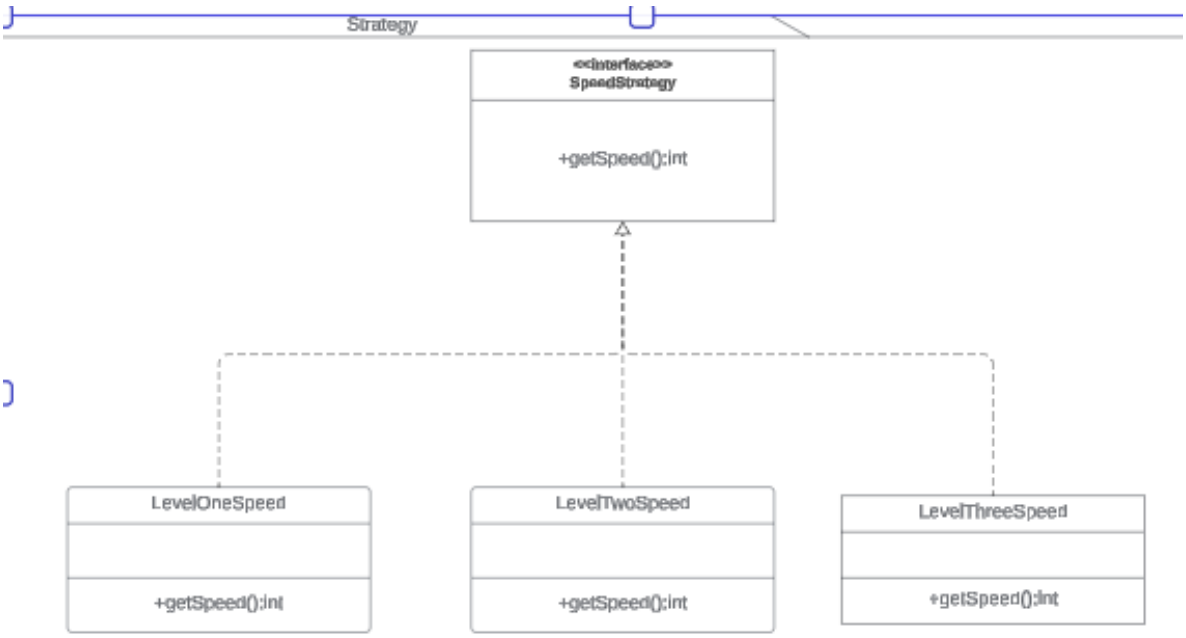


2. State : The State pattern, a behavioral design pattern, is employed to model the dynamic behavior of plates within the game. As plates transition from controllable to movable states, the State pattern facilitates a seamless and organized representation of these changes, enhancing the modularity and maintainability of the codebase.

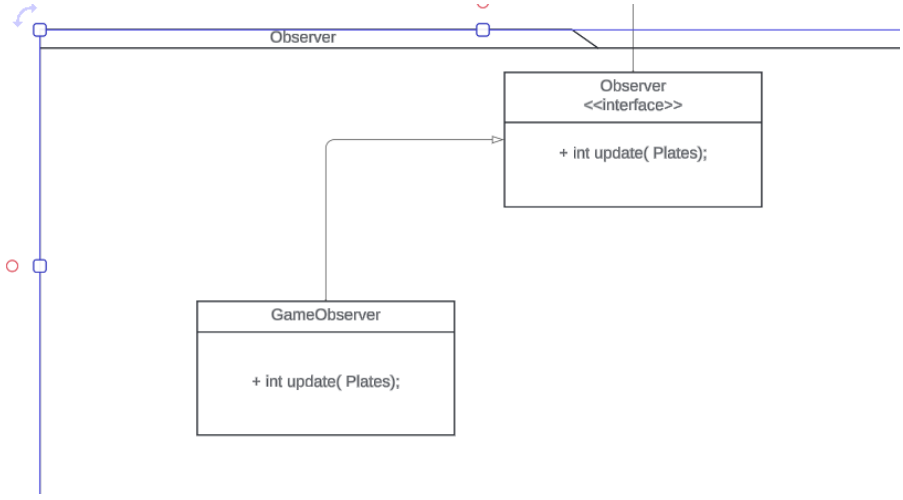


3. Strategy:

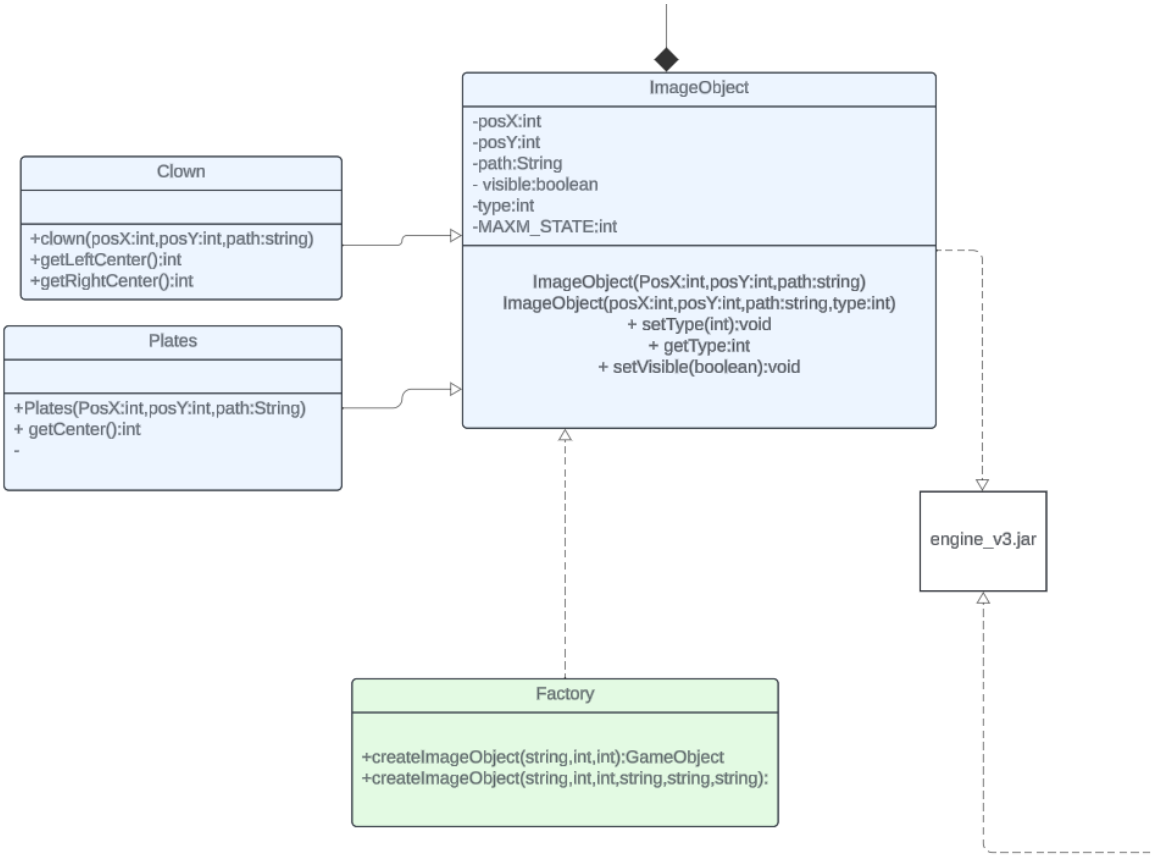
   Incorporating the Strategy pattern, a behavioral design pattern, our implementation addresses scenarios where the speed of the clown game varies across different levels. The SpeedStrategy allows for multiple speed implementations, enabling a flexible and extensible approach to controlling the speed strategy for each game level.



4. Observer: The Observer pattern moniters and responds to changes in the game objects. Implemented in the context of falling objects, the Observer pattern notifies observers about caught objects. This mechanism triggers specific actions based on the type of caught object, exemplified by adjusting the score for bombs and gifts.



5. Factory: Utilizing the Factory pattern, a creational design pattern, our implementation encapsulates the creation of plate and clown images. The ImageObjectFactory efficiently produces instances of these objects, promoting a modular and maintainable approach to object creation



Conclusion

The integration of these design patterns enhances the overall robustness, flexibility, and maintainability of our software system. Each pattern serves a distinct purpose, contributing to the elegance and efficiency of the codebase. This thoughtful design approach lays the foundation for scalability and ease of future enhancements.