Extension Mechanisms Justification

In the redesign of my game, I implemented two key design patterns: the **Factory pattern** and the **Template Method pattern**.

The **Factory pattern** is embodied in the 'GodCardFactory', which manages the creation of 'GodCard' objects, encapsulating unique functionalities. This pattern allows for easy addition of new cards without altering the existing codebase, adhering to the **Open/Closed Principle**. The Factory pattern minimizes code dependencies and isolates creation logic, simplifying the introduction of new 'GodCard' types and enhancing maintainability. Alternatives such as direct instantiation using conditional logic were considered but rejected due to potential violation of the **Single Responsibility Principle** and increased error susceptibility.

The **Template Method pattern** is employed in the 'GodCard' class hierarchy. This pattern defines a skeleton of an algorithm in a base class, allowing subclasses to override specific steps without changing the overall structure. In the context of the game, the 'GodCard' base class defines a template for the behavior of god cards, with methods like 'onBeforeMove', 'onAfterMove', 'onBeforeBuild', and 'onAfterBuild'. These methods provide extension points for each specific god card subclass to implement their unique abilities without modifying the core game logic. This approach promotes code reuse, modularity, and adherence to the **Open/Closed Principle**.

The decision to use the Template Method pattern for god card extensions was based on its ability to encapsulate common behavior while allowing for customization. It provides a clear structure for implementing new god cards, ensuring consistency and maintainability. Alternative approaches, such as using conditional statements within the game logic to handle different god card behaviors, would lead to code duplication and make the system harder to extend and maintain.

The combination of the Factory and Template Method patterns strikes a balance between extensibility, simplicity, and maintainability. It allows for easy integration of new god cards and their unique abilities without requiring significant changes to the existing

codebase. This design choice facilitates future expansions while keeping the codebase manageable and comprehensible.

In conclusion, the use of the Factory and Template Method patterns in the game's redesign demonstrates a commitment to the principles of object-oriented design, particularly the Open/Closed Principle and the Single Responsibility Principle. These patterns provide a robust and flexible foundation for extending the game's functionality, ensuring that it can adapt to new requirements and evolve over time with minimal impact on the existing code structure.