Extension Mechanisms Justification

In the redesign of my game, I implemented the **Factory design pattern** through the introduction of the 'GodCardFactory'. This pattern significantly aids in managing the creation of 'GodCard' objects, which encapsulate unique functionalities. The use of a factory allows us to easily add new cards without altering the existing codebase, aligning with the **Open/Closed Principle**. This principle advocates for systems that are open for extension but closed for modification.

The Factory pattern was chosen to minimize code dependencies and isolate creation logic, which simplifies the introduction of new 'GodCard' types and enhances maintainability. Alternatives considered included direct instantiation using conditional logic within the game code. However, this would compromise the **Single Responsibility Principle** by mixing creation logic with game mechanics and necessitating modifications to the game logic for each new card, increasing error susceptibility.

I opted against more complex patterns such as **Abstract Factory** or **Builder** due to the straightforward nature of the current requirements. The chosen approach avoids unnecessary complexity while providing sufficient flexibility for future game expansion, ensuring the codebase remains manageable and comprehensible.

This design choice represents a balance between extensibility and simplicity, facilitating easy adaptation to future expansions with minimal impact on existing functionality.