**Guess the Number Game Documentation**

**1. Introduction**

The "Guess the Number" game is a simple Java application that allows users to guess a randomly generated number within a specified range. The program employs various design patterns to enhance code structure, flexibility, and maintainability.The main goal is to create a flexible and extensible structure for the game, allowing you to easily make changes to the logic of the game, validation strategies and visualization.

1. **importance of using design patterns in software development.**

* Improving code readability and understandability
* Reduced complexity and easier maintenance
* Code reuse
* Flexibility and extensibility
* Ensuring compliance with design principles
* Facilitating team development
* Improving testing

**3. Design Patterns Used**

Singleton Pattern:

Class: GameData

Purpose: Ensures a single instance of the game data is available throughout the application.

Strategy Pattern:

Classes: GuessValidationStrategy, BasicGuessValidationStrategy, RangeGuessValidationStrategy, GuessDecorator, EvenNumberGuessDecorator, GuessAdapter

Purpose: Provides a family of algorithms for guess validation, allowing dynamic selection and extension of validation strategies.

Observer Pattern:

Classes: GameObserver

Purpose: Enables automatic updates to view components when the game data changes, promoting a decoupled architecture.

Decorator Pattern:

Classes: GuessDecorator, EvenNumberGuessDecorator

Purpose: Dynamically adds responsibilities to guess validation strategies without altering their core logic.

Adapter Pattern:

Class: GuessAdapter

Purpose: Adapts RangeGuessValidationStrategy to be used as a general GuessValidationStrategy.

Factory Method Pattern:

Class: GuessValidationStrategyFactory

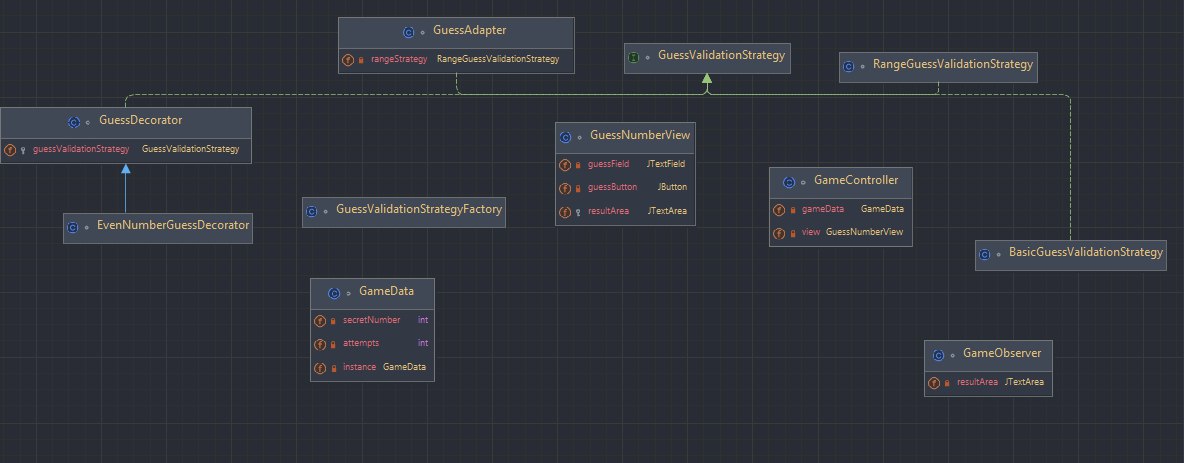
Purpose: Encapsulates the creation logic for guess validation strategies, allowing for easy extension.

MVC Pattern:

Classes: GuessNumberView, GameController

Purpose: Separates concerns related to modeling, user interface, and user input handling, improving modularity.

**4.UML diagram**

****

**5. Usage**

The program initializes a singleton instance of GameData and sets up the main view (GuessNumberView) with a text field, a "Guess" button, and a result area.

Users can input their guesses and press the "Guess" button.

The game controller (GameController) processes the guess using a chosen validation strategy and updates the view accordingly.

The observer (GameObserver) updates the result area with notifications from the game data.

**6. How to Run**

Compile the Java files: javac GameController.java GuessNumberView.java GameData.java GuessValidationStrategyFactory.java GameObserver.java

Run the program: java GameController

**7. Notes and Recommendations**

The range of the secret number (1 to 100) and other game settings can be adjusted by modifying the resetGame method in the GameData class.

Additional validation strategies can be added by creating new classes that implement the GuessValidationStrategy interface or extend existing strategies.

**8. Future Improvements**

Implement a more sophisticated user interface with graphical elements.

Extend the game with multiple levels, scoring, or additional game mechanics.

Conclusion

The "Guess the Number" game demonstrates the effective use of design patterns to achieve a well-organized, flexible, and maintainable software structure. By employing these patterns, the program allows for easy extension, modification, and adaptation to changing requirements.  
  
**9.Reference  
  
https://refactoring.guru/ru/design-patterns/singleton  
https://refactoring.guru/design-patterns/strategy  
https://refactoring.guru/design-patterns/decorator  
https://refactoring.guru/ru/design-patterns/adapter  
https://refactoring.guru/design-patterns/observer  
https://refactoring.guru/design-patterns/factory-method**