Patterns used:

- **Facade pattern**
  - **The UI Updater acts as a proxy between the UI elements and the rest of the classes interact with it to update the UI.**
- **Singleton pattern**
  - **To avoid multiple instances of UIUpdater, we have created it as a singleton. This will ensure there is only one instance of the UIUpdater that makes changes to the UI.**
- **Factory pattern**
  - **Factory pattern is used in GameFactoryFactory and GameFactory to create the game.**

Feature 1. How does the GameController know that a game is selected and what does it need to do before it can call match.start( )? Set up the required infrastructure for GameController to start the match. You have the option of extending the menu from the last iteration or using the original URL based system for selecting the game.

The gameController contains PreGameSetup. This is an observer for all PreGameRequisites, following observer pattern. Once all PreGameValidators are satisfied, it will be notified and will set isGameReady to true. The Game controller will know that all conditions have been met and can start the game.

Feature 2. How does MatchController know when the game can begin and what does it need to do before it can call mainloop.play( )? Assume war is always a two-player game. Set up the required infrastructure for MatchController to start the match.
PreGameController publishes a StartGameEvent every time a game is ready to be started. Other classes like MatchController are subscribed to these events via observer pattern, and will get notified (update event) when the game starts.

3. At the start of play there should be a deal button and the title of the game should be set to "War".
At the start of play (StartGameEvent), MatchController will call the UIUpdater's methods to displayDeckOption and set the title

4. Pressing the deal button results in two even piles of shuffled face down cards. There are many simple shuffle algorithms, any is fine.
DeckShuffler will subscribe to the DealEvent, which is created when the deal button is pressed. This will trigger either of the shuffling strategies, and update the UI to reflect the same.

5. During play, the player's alternate selecting the card at the top of their pile. Their card is placed face up near the center of the table. Players are ignored if they select a card out of turn or from a pile that does not belong to them.

SelectGameEvent is triggered every time a card is selected. This is managed by the SelectEventHandler, which will query the gameController to see which player is supposed to be making the move, and ignore it if its the wrong player.

6. When a player wins the cards in the center they select the winning card and the cards are transferred are transferred to the bottom of their pile. A player that selects a card incorrectly is ignored.
Same as above. SelectEventHandler will add cards to bottom of the pile, or ignore as required.

7. When the center cards are tied a war begins. Each player selects the top of their pile three times moving those cards to a center pile. They then each select a card to show face up. A player that selects a card incorrectly is ignored.
When the war begins, the WarRules is used out of the strategy pattern. The SelectEventHandler still ignores the incorrect player.

8. The display of the players score always represents the number of cards in their pile.
GameRuleEngine and SelectEventHandlers will call updateCardCounts function of the UIUpdater and update the score of the players. (on score updated event)

9. When a player runs out of cards the other player immediately wins. Set the title of the game to "Player X Wins".
The SelectEventHandler determines when a player has won. It will call the WinHandler, which will call the UI Updater to set the title, and show the deck button.

10. Extra game play rule: if a player improperly selects a center card (they are not the winner of the cards) all of the cards in the center go to the other player.
The SelectEventHandler will be notified of SelectGameEvent, and will trigger GameController to change the score. This will trigger a score change event and the SelectEventHandler will determine that a player has won.

11. When the game is finished (one player wins) show the deal button. This will be helpful.
The win handler will show the deal button using the UI Updater.

12. When the deal button is selected for a new game, deal the existing cards (not new ones) to the two players
When the StartGameEvent is triggered, the MatchController will call the DeckShuffler with the existing deck to create a shuffle.