**Observer Pattern**: The `EndMenu` and `PauseMenu` classes use the Observer pattern to notify other parts of the application when certain events occur. They define events (`OptionsChanged`) and allow other classes to subscribe to these events and be notified when the events are raised.

Prototype Pattern

### Board Class Overview (Singleton)

To summarize, the `Board` class maintains:

- Piece positions.
- Skipped squares for en passant.
- Castling rights.
- Methods to interact with and query the board state (e.g., `Inside`, `Empty`, `InCheck`).

### GameState Class Overview

The `GameState` class manages:

- The current player's turn.
- The result of the game.
- Legal moves and move execution.
- Overall game flow and state persistence.

### 1. `MainWindow` Class

**Relationships:**

- **ChessRules.Player**
  - **Reason:** `MainWindow` uses the `Player` enum to manage the current player's turn in the game.
- **ChessRules.GameState**
  - **Reason:** `MainWindow` manages the game state, which involves creating a new game state, loading a game state from a file, and handling moves within the game. `GameState` encapsulates the current state of the game, including the board and the player's turn.
- **ChessRules.Board**
  - **Reason:** `MainWindow` interacts with the `Board` class to set up the initial board configuration, draw the board state, and load the board state from a file.
- **ChessRules.Move**
  - **Reason:** `MainWindow` handles player moves and needs to store and execute these moves. The `Move` class represents the moves that can be made in the game.

- **ChessRules.Positions**
  - **Reason:** `MainWindow` uses the `Positions` class to keep track of the positions on the chessboard where pieces are located, as well as where they can move.
- **ChessRules.Promotion**
  - **Reason:** `MainWindow` handles pawn promotions, which are a special type of move represented by the `Promotion` class.
- **PromoMenu, EndMenu, PauseMenu, MainMenu**
  - **Reason:** `MainWindow` interacts with various menu classes to manage different UI states, such as showing the promotion menu, end game menu, and pause menu. These classes represent different UI components that are displayed based on the game state.
- **System.Windows.Controls.Image, System.Windows.Controls.Rectangle**
  - **Reason:** `MainWindow` uses `Image` and `Rectangle` controls to display pieces on the board and highlight legal moves.
- **System.IO, System.Text**
  - **Reason:** `MainWindow` handles saving and loading game states to and from files, requiring file I/O operations.

### 2. `GameState` Class

**Relationships:**

- **ChessRules.Player**
  - **Reason:** `GameState` uses the `Player` enum to keep track of which player's turn it is.
- **ChessRules.Board**
  - **Reason:** `GameState` encapsulates a `Board` object that represents the current state of the chessboard.
- **ChessRules.Move**
  - **Reason:** `GameState` manages the execution and validation of moves, represented by the `Move` class.

### 3. `Board` Class

**Relationships:**

- **ChessRules.Piece**
  - **Reason:** `Board` contains a 2D array of `Piece` objects, representing the pieces on the chessboard.

### 4. `Piece` Class

**Relationships:**

- **ChessRules.Player**

    - **Reason:** `Piece` uses the `Player` enum to keep track of which player (black or white) the piece belongs to.
- **ChessRules.PieceType**
    - **Reason:** `Piece` uses the `PieceType` enum to represent different types of chess pieces (e.g., Pawn, Rook, Knight).

### 5. `PromoMenu`, `EndMenu`, `PauseMenu`, `MainMenu` Classes

**Relationships:**

- **ChessRules.Player**
    - **Reason:** These menu classes use the `Player` enum to display information about the current player or configure options based on the player.
- **ChessRules.PieceType**
    - **Reason:** `PromoMenu` uses the `PieceType` enum to allow the player to select a piece type for pawn promotion.
- **Events (SelectedPiece, OptionsChanged)**
    - **Reason:** These menu classes raise events to notify the `MainWindow` of user selections or actions, implementing the Observer pattern.