1. State Pattern (Tetris State)

The Tetris class appears to implement the State pattern to manage different states of the game, such as "start" and "gameover". In this pattern, the behavior of the Tetris game varies depending on its current state. State-specific behaviors are encapsulated within separate classes or methods, allowing for a more organized and manageable code structure. This design pattern facilitates the ease of adding new states or modifying existing ones without significantly altering the core game logic.

2. Strategy Pattern

The Strategy pattern is employed in the Tetris game to define a family of algorithms, particularly seen in the implementation of palette swapping and score counting methods. Each algorithm is encapsulated and made interchangeable, promoting flexibility and adaptability in the game's functionality. This pattern is also observed in other features such as Speed Increase, Dark Mode, Sound Effect, Game Over Screen, Piece Preview, and Saved Piece. By separating concerns, the Strategy pattern enables more straightforward maintenance and extension of each feature without affecting others.

3. Memento Pattern (Piece Saving)

The piece saving functionality in the Tetris game utilizes a variation of the Memento design pattern. This pattern is evident in how the game captures the current Tetris piece (the originator) and stores it for later use in the SavedPiece class (the caretaker). The saved piece is then utilized in various ways, including visually displaying the next piece and allowing the player to swap the saved piece with the current one upon a specific key input. This pattern provides a mechanism to save and restore the game state at a particular moment, enhancing the gameplay experience by adding a strategic element to the game.