CS5010 Programming Design Paradigm

Recitation 9: State Patterns in Chess

Xinyu Wang, Erdun E, Fengkai Liu, Raj Kavathekar

Reflection

In this Chess Game project, we implemented 1 parent state class and 5 state classes, it's following:

GameStartState: This state represents the game initialization.

- Functionality includes:
  - Initialize the game (partially implemented)
  - Restart the game (not implemented)

NormalPlayState: this state represents the game playing in progress, which is from the game initialization until one player is in check, and from one player resolving a check condition until the next check condition.

- Functionality includes:
  - Return PlayerTurnSwitchState: from the game initialization to before one player is in check
  - Return CheckState: transitioning from Normal Play State to Check State.

CheckState: this state represents a critical point of this game. If the check condition gets resolved, this game will transition back to Normal Play State; otherwise it will transition to CheckMate State.

- Functionality includes:
  - Return NormalPlayState: when the check condition gets resolved
  - Return CheckMate State: when the check condition didn't' get resolved

CheckMateState: this state represents the end of the game.

- Functionality includes:
  - Ending the game.

PlayerTurnSwitchState: this state represents changing the turn of two players.

- Functionality includes:
  - Switch from White to Black; Switch from Black to White

We all agree that NormalPlayState, CheckState, and CheckMate are critical to the functionality of this game based on the game definition.

Out Look

There are still some other states discussed but not implemented:

ResetState: With the current GameStartState, the game can indicate a starting point. A reset function could allow the game to restart after checkmate or game end. This was not implemented because the other states are essential to the game. Without ResetState, users can still play the game seamlessly.

AttackState: Another idea was using an AttackState to represent a player capturing an opponent's piece. However, this functionality overlaps with NormalPlayState, so we opted to handle attacks directly within the ChessGame class.

We also considered an alternative approach with states like User1State, User2State, User1CheckState, User2CheckState, User1CheckMateState, and User2CheckMateState. This setup would differentiate between User1 and User2 more explicitly. However, our current state structure is simpler and clearer, as it avoids using states to track the current player.