Igor Overchuk, Ian Scheuermann, May An van de Poll

Professor Wright

CSCI 5448-002: Object Oriented Analysis and Design

04/12/2024

Homework 7: Semester Project Interim Report

Title: Chess

Work Done

Igor: I created Game, Board and Tile classes and their tests. The Board and Tile classes should be done but may need to be extended. The Game class is a work in progress but should have most of its functionality ready to go. I also created the IChessGame interface to interact with the UI. Finally, I created the TileType and TileHighlightType enums that will be used to track game state and currently selected UI components.

May An: Created a GuiObserver, AudibleObserver, EventBus and EventType and tests for them. The GuiObserver needed some changes that Igor helped with. The EventBus follows the Singleton pattern and the observers follow the Observer pattern.

Ian: I created the piece classes and implemented the logic for all of the valid moves. There are a few more complex moves such as castling and en passant that still need to be added. I also created a factory pattern for the pieces so that they can be instantiated easily by the game at initialization. The logic for checking if a king is in check, is implemented but needs to be incorporated to valid moves to verify if moving a piece would expose your king to check still needs to be added. The Piece superclass and specific piece subclasses follow the template pattern.

Changes or Issues Encountered

We have decided not to move forward with the Strategy pattern for our pieces as that would introduce more complexity than it would resolve. Having all those extra classes would not be beneficial as we can track the state of the pieces in an instance variable.

Patterns

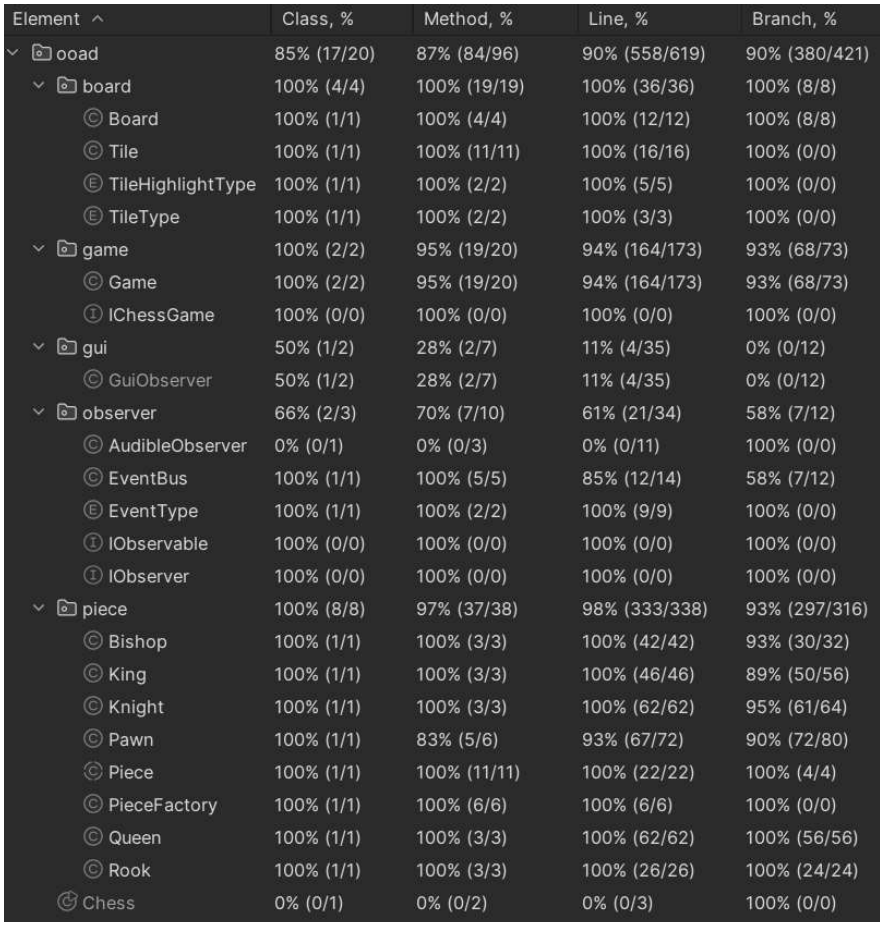
Singleton: The EventBus is implemented similarly to Arcane and is a singleton to be used across all of our classes.

Observer: The AudibleObserver is similar to Arcane and is implementing the IObserver interface that tracks the IObservable Game class to track event bus events. The GUI will also be an observer of the Game class so it knows when to update the UI.

Factory: The pieces have a PieceFactory to abstract the creation of all pieces.

Template: The abstract base class Piece is used to define basic piece behaviors and must be the base class for all pieces. The Piece class is what is used in the Game class to track the entire game.

Test Coverage Screenshot



Element	Class, %	Method, %	Line, %	Branch, %
ooad	85% (17/20)	87% (84/96)	90% (558/619)	90% (380/421)
board	100% (4/4)	100% (19/19)	100% (36/36)	100% (8/8)
Board	100% (1/1)	100% (4/4)	100% (12/12)	100% (8/8)
Tile	100% (1/1)	100% (11/11)	100% (16/16)	100% (0/0)
TileHighlightType	100% (1/1)	100% (2/2)	100% (5/5)	100% (0/0)
TileType	100% (1/1)	100% (2/2)	100% (3/3)	100% (0/0)
game	100% (2/2)	95% (19/20)	94% (164/173)	93% (68/73)
Game	100% (2/2)	95% (19/20)	94% (164/173)	93% (68/73)
IChessGame	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
gui	50% (1/2)	28% (2/7)	11% (4/35)	0% (0/12)
GuiObserver	50% (1/2)	28% (2/7)	11% (4/35)	0% (0/12)
observer	66% (2/3)	70% (7/10)	61% (21/34)	58% (7/12)
AudibleObserver	0% (0/1)	0% (0/3)	0% (0/11)	100% (0/0)
EventBus	100% (1/1)	100% (5/5)	85% (12/14)	58% (7/12)
EventType	100% (1/1)	100% (2/2)	100% (9/9)	100% (0/0)
IObservable	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
IObserver	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
piece	100% (8/8)	97% (37/38)	98% (333/338)	93% (297/316)
Bishop	100% (1/1)	100% (3/3)	100% (42/42)	93% (30/32)
King	100% (1/1)	100% (3/3)	100% (46/46)	89% (50/56)
Knight	100% (1/1)	100% (3/3)	100% (62/62)	95% (61/64)
Pawn	100% (1/1)	83% (5/6)	93% (67/72)	90% (72/80)
Piece	100% (1/1)	100% (11/11)	100% (22/22)	100% (4/4)
PieceFactory	100% (1/1)	100% (6/6)	100% (6/6)	100% (0/0)
Queen	100% (1/1)	100% (3/3)	100% (62/62)	100% (56/56)
Rook	100% (1/1)	100% (3/3)	100% (26/26)	100% (24/24)
Chess	0% (0/1)	0% (0/2)	0% (0/3)	100% (0/0)

Work Estimate

- Draw piece images on the GUI.
- Fix the bug with the pawn's first move.
- Implement more complex moves like En Passant and Castling.
- Implement more advanced status checking to make sure you can’t put yourself into check and use a piece to cover your king from a threat as well.
- Finalize some tests to make sure we have full coverage.