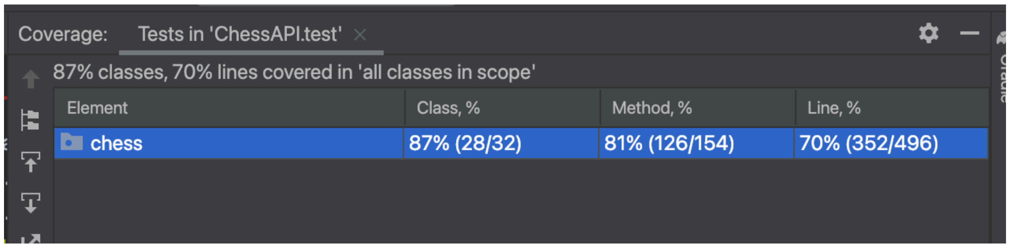
Ben Niesmertelny
Giovanni Evans

Homework 7: OOAD Interim Semester Project Report

Status Summary:

- Title:
    - Chess API Object Oriented Project
- Group Members:
    - Giovanni Evans, Ben Niesmertelny
- Work Done:
    - So far we have found a chess API that uses the JSON and Rest format, which is also free. We have added Gradle dependencies that allow us to access and call the API. Additionally, we have also included an entire class within the src class called api that holds the necessary files that allow us to use the api within our code. In terms of code that we have written, the entire game class holds the classes for the board, games, pieces, and moves. The majority of the code is done for the Board, Game, and Piece classes. We have started work on the Moves class, but we still have to implement the legal moves functions for each piece. In addition, we have written tests to account for 80% of the methods in our code. Although both of us worked on all aspects of the project, Ben worked primarily on setting up the API and implementing the game files, and Giovanni worked primarily on testing and implementing the game files.
- Issues:
    - We have not come across any glaring issues that have caused us to make big changes to the original plan.
- Test Coverage:
    - 

Coverage: Tests in 'ChessAPI.test'

87% classes, 70% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
chess	87% (28/32)	81% (126/154)	70% (352/496)

Class Diagram:

- See github repository

BDD Scenarios:

- Check and Checkmate
  - Scenario: As a player, I want to determine if my king is in check.
    - Given a specific board configuration
    - When it is the player's turn
    - Then the player should be notified if their king is currently in check.
  - Scenario: As a player, I want to determine if I have achieved checkmate.
    - Given a specific board configuration where the opponent's king is in check
    - When it is the player's turn
    - Then the player should be able to make a move that results in the opponent's king being in checkmate.
- Pawn Promotion
  - Scenario: As a player, I want to promote my pawn when it reaches the opposite end of the board.
    - Given a pawn has reached the opposite end of the board
    - When the player chooses to promote the pawn
    - Then the pawn should be replaced with a queen, rook, bishop, or knight, as per the player's choice.
- En Passant
  - Scenario: As a player, I want to perform en passant capture.
    - Given a specific board configuration where the opponent's pawn has just moved two squares forward
    - When the player's pawn is in a position to capture en passant
    - Then the player should be able to capture the opponent's pawn as if it had only moved one square forward.
- Castling
  - Scenario: As a player, I want to perform castling.
    - Given a specific board configuration where the king and rook involved in castling have not moved previously
    - When the player chooses to castle
    - Then the king should move two squares towards the rook, and the rook should be placed next to the king on the opposite side.

Plan For Completion:

- Our next steps are completing the legal moves functions for all of the pieces, implementing castling, en passant, and check/checkmate scenarios, and then ultimately getting to the point where we can run entire games through the terminal. Additionally, if time permits, we will try and find an additional existing chess API that will allow us to visualize the board at all times by using an adapter class on our end.