- What is your unit test coverage
    - We have unit tests for the back-end logic for each one of our games.
        - TicTacToe:
            - Board has 100% code coverage
            - BoardManager has 100% code coverage
            - TicTacToeTile as 100% code coverage
        - MatchingCards
            - Board 50% code coverage
            - BoardManager 28% code coverage
            - Tile: 66% line coverage
    - We have unit tests for our scoreboard and score related classes
        - Score: 76% code coverage
        - Scoreboard: 100% line coverage
        - ScoreboardMatchingCards, ScoreboardSlidingTiles, ScoreboardTicTacToe: 100% line coverage
        - ScoreMatchingCards, ScoreSlidingtiles, ScoreTicTacToe: 100% line coverage
- What are the most important classes in your program
    - The main gameplay activities for each of the games
    - The board managers for each of the games
- What design pattern did you use? What problems do each of them solve?
    - Observer
        - The observer allows us to react to changes to the board and update the
    - Composite Design Pattern
        - Game activties are made
    - Adapter
        - BoardManager to bytearrays
        - GesteureDetectGridView
    - Template
        - With abstract classes
    - Chain of Resposibility
        - DBTools -> Firebase
- How did you design your scoreboard? Where are high scores stored? How did they get displayed?
    - Each game has its own score class. This is done because each game has its own way of comparing which score is higher than the other.
    - The scores are stored on a Firebase NoSQL database.

- Each score has its own corresponding scoreboard because it improves code clearity by reducing the usage of wildcards. It also prevents having to have functions that return objects defined by wildcards.
- There are two states for the scoreboard. The game scoreboard and the user scoreboard. The data that is taken from the database differs depending on the state of the scoreboard.
- The scores are taken from the database and afterwards they are put into their corresponding scoreboards. Each scoreboard uses a method to organize the scores in a way where “higher” scores are placed near the front.
- Each score has a toString method which is used by the scoreboard to create a list of strings that represent each score. The title of the game is put before any of the scores.
- All the “printed” scoreboards are concatenated to one “printed” scoreboard which is then displayed by the scoreboard activity’s listview.
- A button is used to toggle between the two states of the scoreboard. Each press changes the data extracted from the database based on the state of the scoreboard and recalls the method with listview to display the data.