Checkers

Group 14:
Blake Galbavy
Austin Metz
Galen Pogoncheff
Richard Poulson

14 - Checkers

Vision:

- To create a Checkers game system that can be played both locally or remotely with a friend.

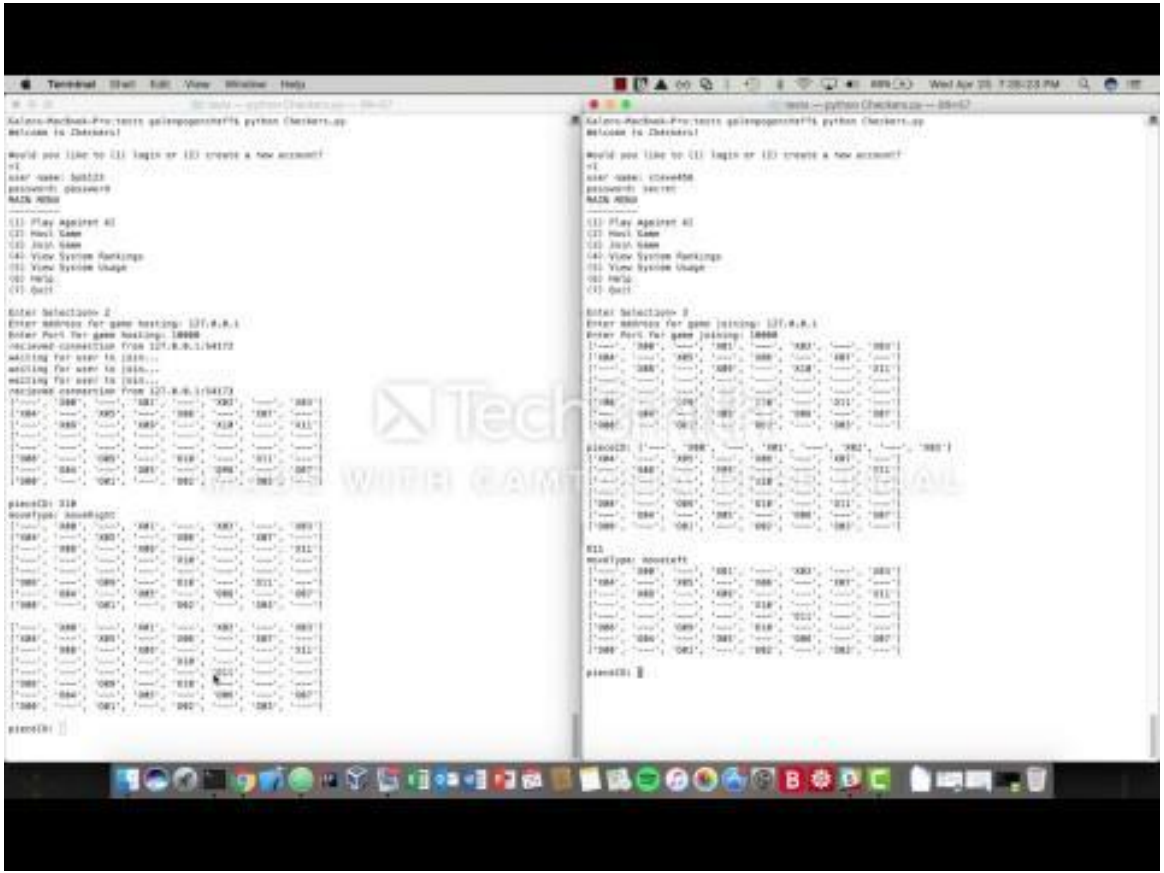
Project Description:

- ‘Checkers’ is an application that allows users to play the classic Checkers board game. Users of the application will have the ability to play checkers against other users on the network, as well as against an automated computer player (driven by artificial intelligence algorithms). User statistics, such as number of wins, number of losses, and total play time, are recorded in a database.

Use Case-03: Play Against User

- Primary Hosting User and Joining User connect
- Board is displayed for both users
- Primary User has first move and selects a checker piece
- Primary User chooses where to move the selected piece
- Board updates for both users
- Joining User selects move
- Game continues playing from the pattern above

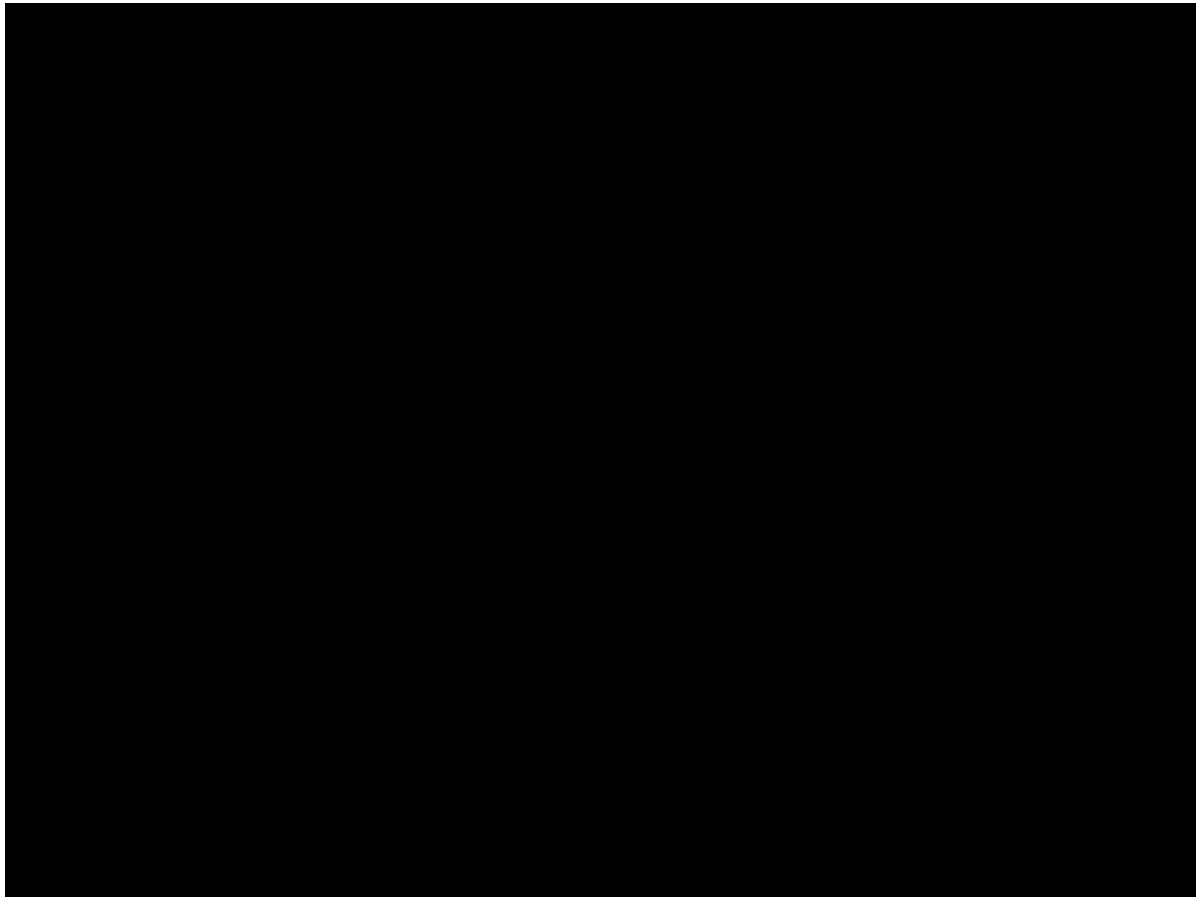
Use Case-03 Demo



Use Case-05: Play AI-Player

- After choosing to play against AI, a new instance of CheckersBoard is created and set as the current game.
- The computer player sets the heuristic function that it will use for its AI strategy, then initializes its strategy as well.
- Both the human and computer players are notified, the human player makes the first move, and both players continue until a winner is decided.

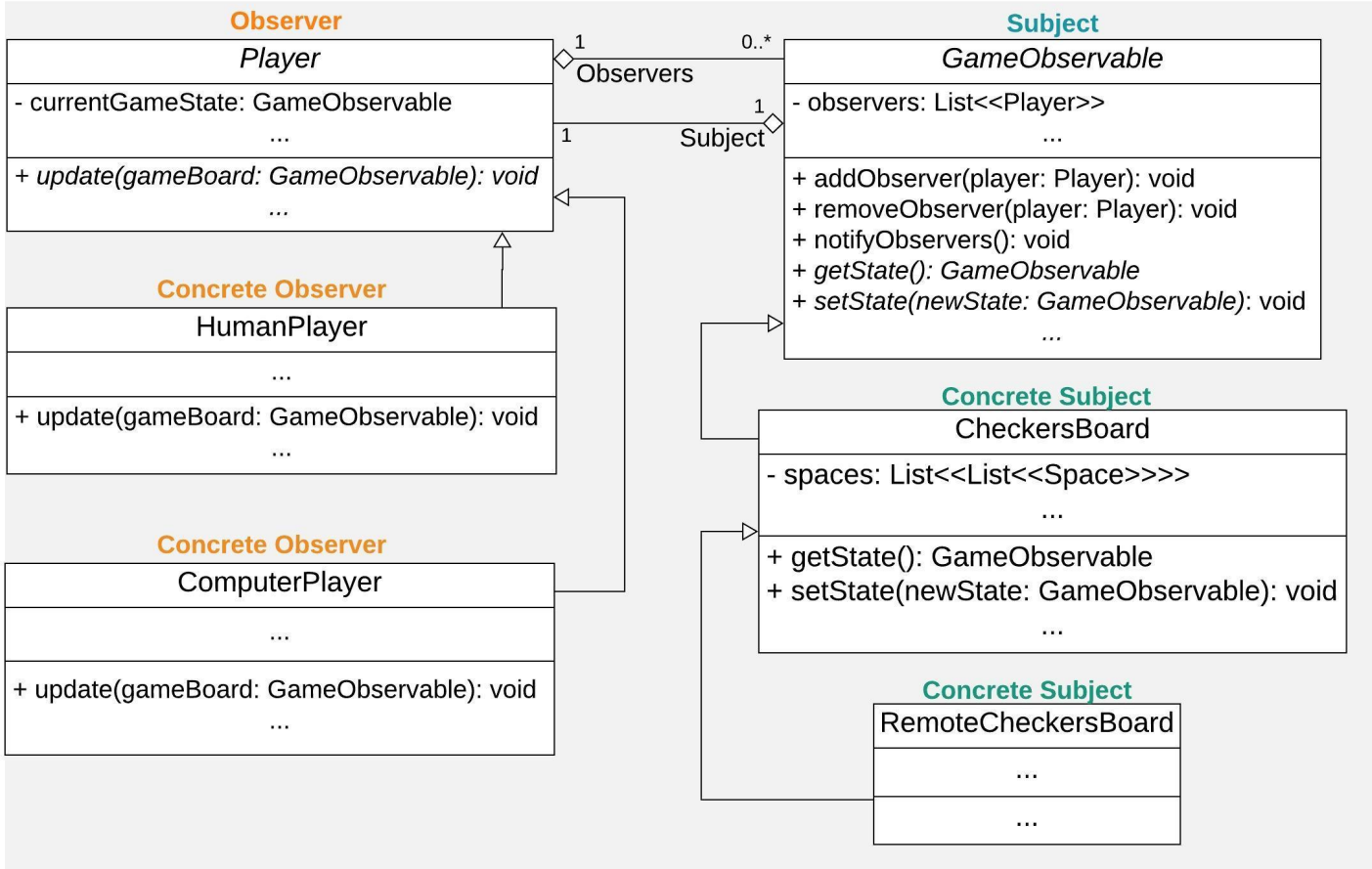
Use Case-05: Demo



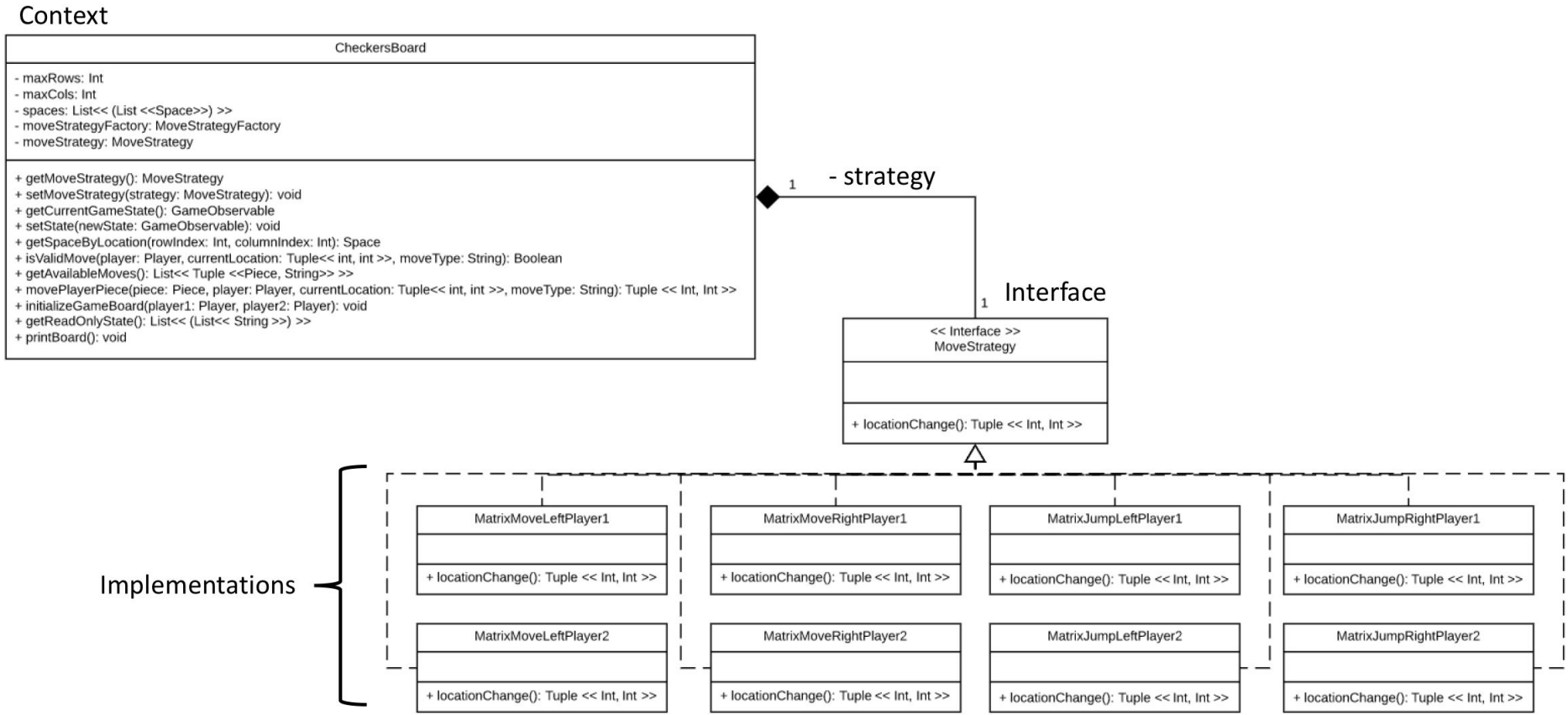
Using the Observer Pattern

- The Subject is the abstract class GameObservable, extended by the CheckersBoard and RemoteCheckersBoard classes. Player objects are the observers in the implementation.
- When a CheckersBoard or RemoteCheckersBoard is initialized, two Player objects are passed as arguments, which are then attached to the board.
- After a Player makes a move, the board notifies all attached Players, causing them to update. Updating the Player objects causes them to update their saved state of the board.
- The ComputerPlayer uses this updated game state to calculate possible moves in its following turn.

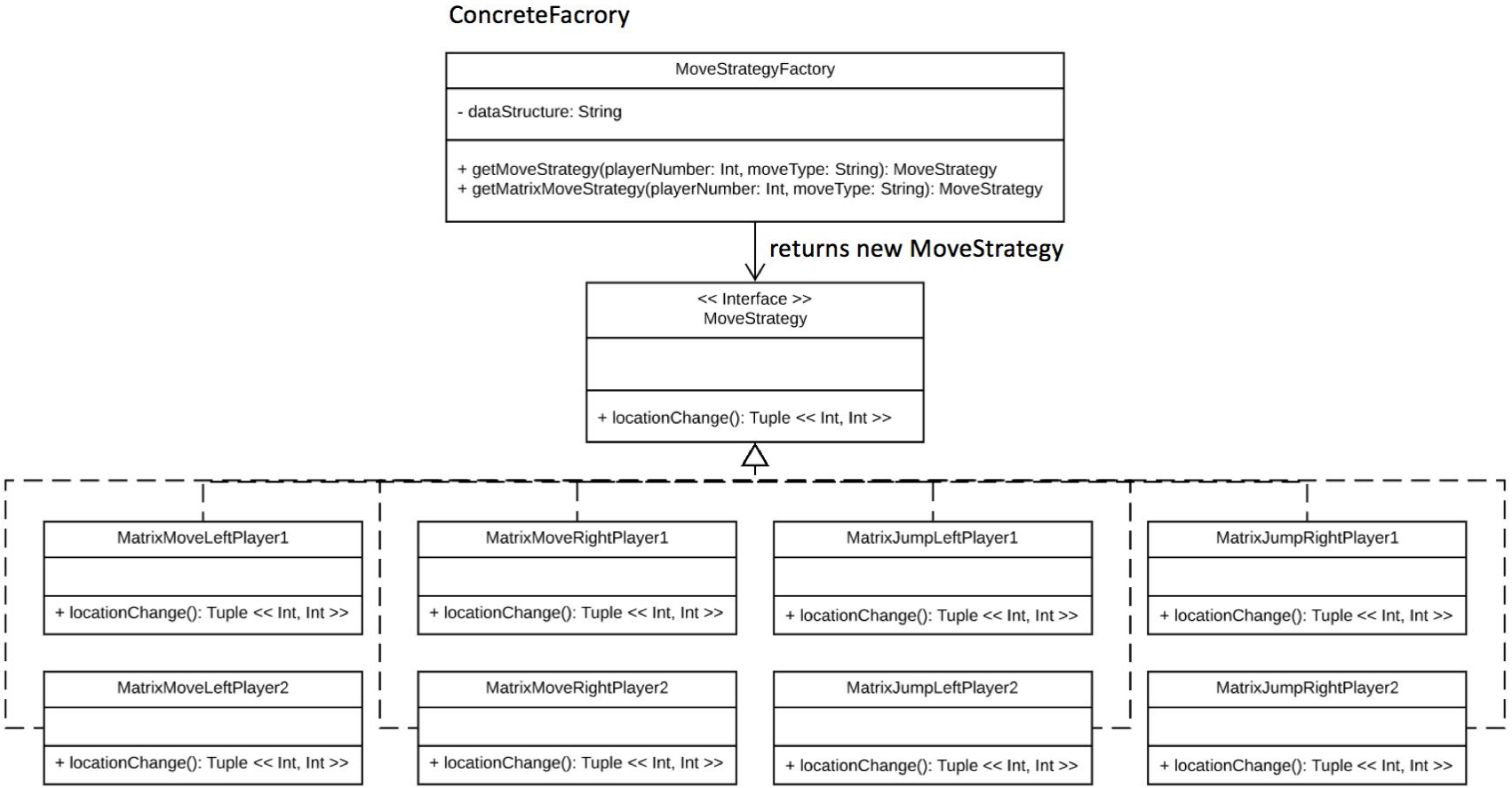
Using the Observer Pattern



Using the Strategy Pattern



Using the Factory Pattern to create Strategy Objects



Link to Project Demo

https://drive.google.com/open?id=1l2MmiAySGg3ZqhvDl5JgiLdmTpVHQBHR