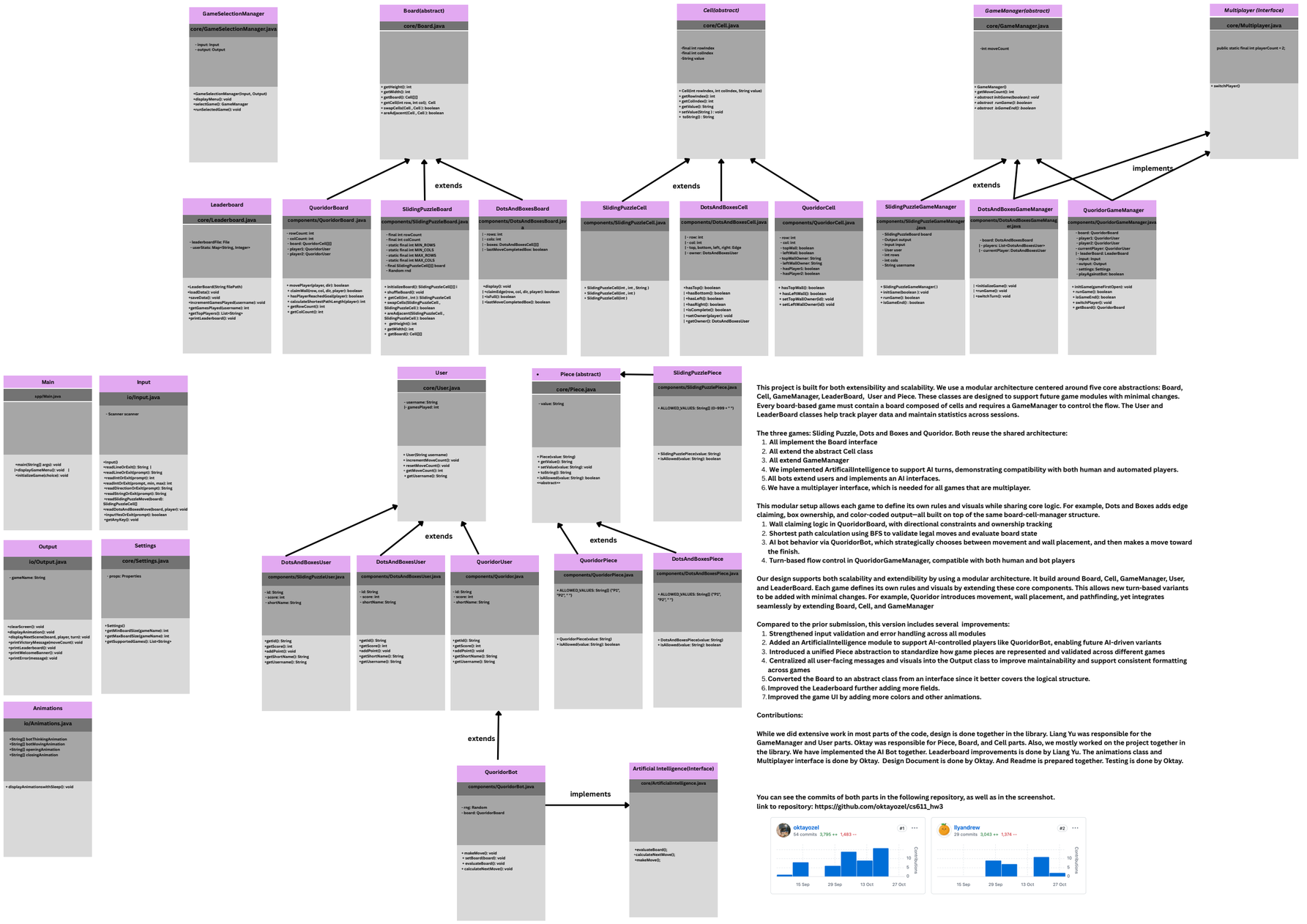
UML Diagram for Assignment 3
Oktay Ozel, Liang Yu Lin



This project is built for both extensibility and scalability. We use a modular architecture centered around five core abstractions: Board, Cell, GameManager, LeaderBoard, User and Piece. These classes are designed to support future game modules with minimal changes. Every board-based game must contain a board composed of cells and requires a GameManager to control the flow. The User and LeaderBoard classes help track player data and maintain statistics across sessions.

The three games: Sliding Puzzle, Dots and Boxes and Quoridor. Both reuse the shared architecture:

1. All implement the Board interface
2. All extend the abstract Cell class
3. All extend GameManager
4. We implemented ArtificailIntelligence to support AI turns, demonstrating compatibility with both human and automated players.
5. All bots extend users and implements an AI interfaces.
6. We have a multiplayer interface, which is needed for all games that are multiplayer.

This modular setup allows each game to define its own rules and visuals while sharing core logic. For example, Dots and Boxes adds edge claiming, box ownership, and color-coded output—all built on top of the same board-cell-manager structure.

1. Wall claiming logic in QuoridorBoard, with directional constraints and ownership tracking
2. Shortest path calculation using BFS to validate legal moves and evaluate board state
3. AI bot behavior via QuoridorBot, which strategically chooses between movement and wall placement, and then makes a move toward the finish.
4. Turn-based flow control in QuoridorGameManager, compatible with both human and bot players

Our design supports both scalability and extendibility by using a modular architecture. It build around Board, Cell, GameManager, User, and LeaderBoard. Each game defines its own rules and visuals by extending these core components. This allows new turn-based variants to be added with minimal changes. For example, Quoridor introduces movement, wall placement, and pathfinding, yet integrates seamlessly by extending Board, Cell, and GameManager

Compared to the prior submission, this version includes several improvements:

1. Strengthened input validation and error handling across all modules
2. Added an ArtificialIntelligence module to support AI-controlled players like QuoridorBot, enabling future AI-driven variants
3. Introduced a unified Piece abstraction to standardize how game pieces are represented and validated across different games
4. Centralized all user-facing messages and visuals into the Output class to improve maintainability and support consistent formatting across games
5. Converted the Board to an abstract class from an interface since it better covers the logical structure.
6. Improved the Leaderboard further adding more fields.
7. Improved the game UI by adding more colors and other animations.

Contributions:

While we did extensive work in most parts of the code, design is done together in the library. Liang Yu was responsible for the GameManager and User parts. Oktay was responsible for Piece, Board, and Cell parts. Also, we mostly worked on the project together in the library. We have implemented the AI Bot together. Leaderboard improvements is done by Liang Yu. The animations class and Multiplayer interface is done by Oktay. Design Document is done by Oktay. And Readme is prepared together. Testing is done by Oktay.

You can see the commits of both parts in the following repository, as well as in the screenshot.
link to repository: https://github.com/oktayozel/cs611_hw3