Mancala Board Game

Final Report

Java Juggernaut's
James Hagan, Aidan Kaiser, Abby Skillestad, Liam Kordsmeier

Course: CPSC 224 - Software Development

I. Introduction and Project Description

We made the board game, Mancala. This is a 2 player turn-based board game played with stones/marbles. Players begin by placing a certain number of stones, prescribed for the particular game, in each of the pockets on the game board. A player may count their stones to plot the game. A turn consists of removing all stones from a pocket, and capturing based on the state of the board. The game's objective is to obtain the most stones in your bank/mancala.

II. Team Members - Bios and Project Roles

Abby Skillestad is a computer science student at Gonzaga University that is interested in software app development, game development, and hiking. Abby's skills include C/C++, Python, Java, and JavaSwing. For this project, her responsibilities included developing the Player class and Rules Screen for the GUI.

Liam Kordsmeier is a computer science student at Gonzaga University that is interested in game development and artificial intelligence. Liam's skills include, C/C++, Java, and python. For this project he was responsible for creating the main game loop and the End Screen for the GUI.

James Hagan is a computer science student at Gonzaga University that is interested in computers, model kits, and cars. His skills include C++, Java, UI/UX, and collaboration. For this project, he was in charge of the turn screen, as well as implementing the board and a single instance of a turn.

Aidan Kaiser is a computer science student at Gonzaga University that is interested in machine learning, soccer and plants. His skills include C/C++, Java and teamwork. For this project, he was in charge of the landing screen as well as implementing specific game interactions like capturing pockets and retaking turns.

III. Project Requirements

Here we will provide a description of major features that must be implemented for a viable and useful product. Major features include broad feature areas, constraints that must be met, and other major items that must be completed for the project to be considered successful.

Functional Requirements:

- Players need to move stones across board: moveStones
- Players need to score points: getScore
- Game needs to end: endGame

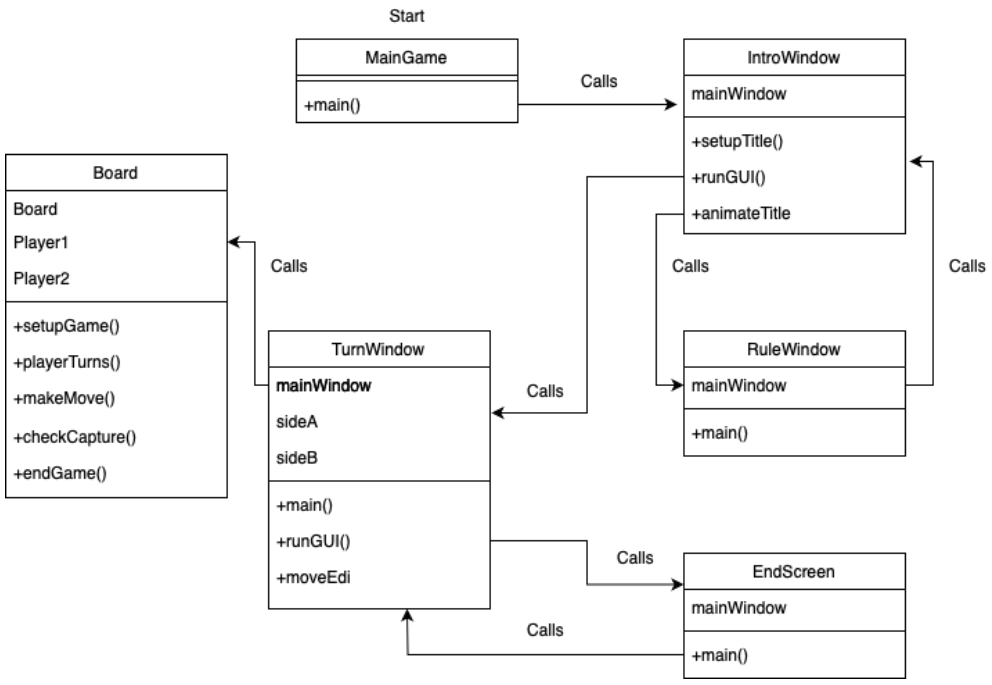
Non-Functional Requirements:

- Pockets show how many stones are in them with images
- Players have names that are changeable: getName
- Multiple Screens: RuleWindow, IntroWindow, TurnWindow, EndScreen

IV. Solution Approach

After our team became experts in the Mancala game rules and mechanics, we made a total of 11 classes. We used Object Oriented Programming for classes such as Board, Pocket, Mancala, and Player. For our games GUI, we worked with JavaSwing due to its easy startup and the fact that we originally coded in Java. With JavaSwing, we had a JFrame for each window, including each JFrame having its own class; this allowed us to easily have each JFrame lead to another JFrame.

Simplified UML Diagram:



I. Test Plan

We implemented a combination of Unit tests, as well as preset User tests to work through our project and make sure that our project worked the way we intended it to. These included

tests such as checking the EndGame, checking specific game mechanics such as PlayAgain and Capture, and testing individual Player scores.

To avoid having to debug the GUI in JavaSwing, which we were all unfamiliar with at the beginning of this project, we agreed on the plan to code the game entirely in the Terminal, which made the transition to GUI much smoother and with minimal amounts of bugs.

II. Project Implementation Description

For the GUI board, we created a shortened version of MVC, with the TurnWindow class acting as our view and controller, and using Board as our model. This helped make the transition from terminal to GUI as swift as possible. It also made our system simplified and straightforward, allowing us to quickly expand our program and find problems. (See Appendix A)
For our whole GUI system, we wanted each screen to be a separate JFrame, while still being able to move between screens. To do this, we added buttons on each screen that will lead to another screen creating a directed graph of sorts out of JFrames.(See Appendix B)

Github Repository Link:
https://github.com/GU-2024-Spring-CPSC224/final-team-project-java-juggernauts

III. Future Work

As we look towards the future, I think that we could implement a more visually complete starting board, including animation of stones from pocket to pocket. There are also a wide variety of Mancala games/rules you can play, such as “Avalanche” mancala; a dropdown menu with different game types could work for this. Game move hints could also expand the game and benefit the user experience.

IV. Glossary

Pocket: singular slot on the board where stones can go

Mancala: not just the name of the game, each player’s “bank” is a mancala.

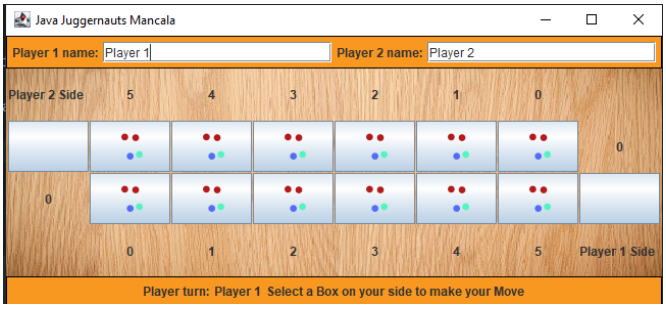
Stones: go in pockets and mancalas, move around the board.

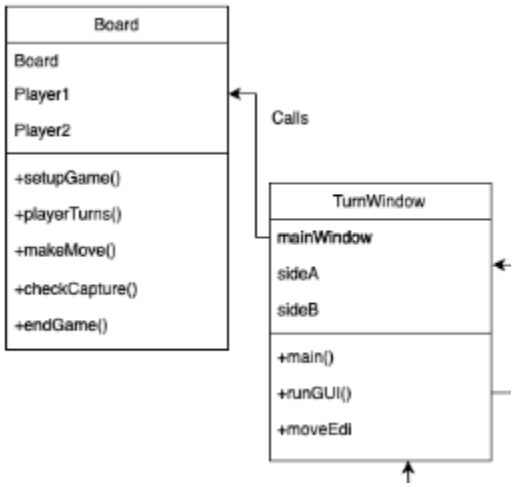
V. References

Game Rules:
Gasser, Mike. “Mancala Classic Instructions.” Matawan: Endless Games, 2015.
https://endlessgames.com/wp-content/uploads/Mancala_Instructions.pdf

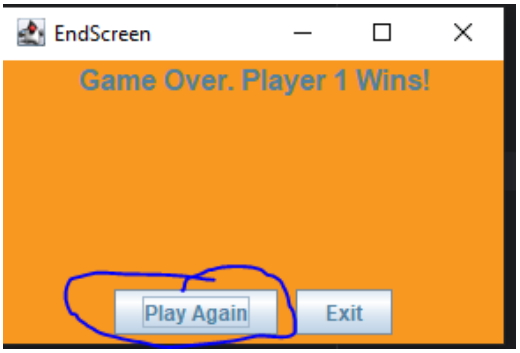
VI. Appendices

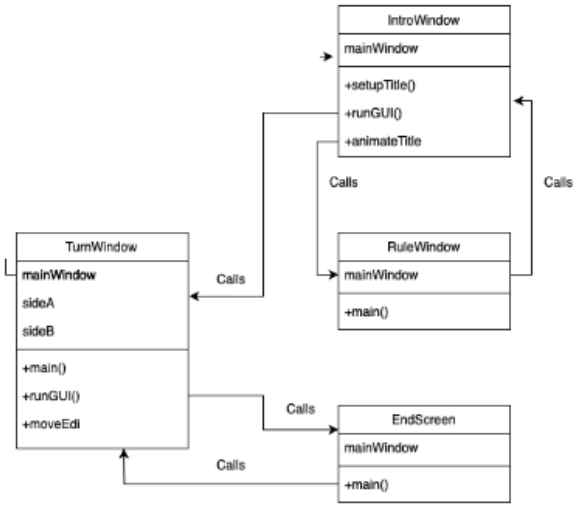
Appendix A


Java Juggernauts Mancala
Player 1 name: Player 1
Player 2 name: Player 2
Player 2 Side
5 4 3 2 1 0
0
0
0 1 2 3 4 5
Player 1 Side
Player turn: Player 1 Select a Box on your side to make your Move


Board
Board
Player1
Player2
+setupGame()
+playerTurns()
+makeMove()
+checkCapture()
+endGame()
Calls
TurnWindow
mainWindow
sideA
sideB
+main()
+runGUI()
+moveEdi

Appendix B


EndScreen
Game Over. Player 1 Wins!
Play Again
Exit


IntroWindow
mainWindow
+setupTitle()
+runGUI()
+animateTitle
Calls
Calls
TurnWindow
mainWindow
sideA
sideB
+main()
+runGUI()
+moveEdi
Calls
RuleWindow
mainWindow
+main()
Calls
EndScreen
mainWindow
+main()
Calls