Assignment 2

PROGRAMMING TECHNOLOGY

Md Rakibul Hasan

JR9RXP

Task 2

Problem:

Pebble is a two-player game, played on a board consists of n x n fields. Initially, n white and n black pebbles are placed on the board randomly. Each color belongs to only one player. The players take turns choosing one of its pebbles, and then move it horizontally or vertically. The movement also affects the neighboring pebbles in the direction (the pebble on the edge falls off). The objective of the game is to push out as many pebbles of the opponent from the board as we can, within a given number of turns (5n). A player wins, if he has more pebbles on the board at the end than his opponent. The game is draw, if they have the same number of pebbles on the board. Implement this game, and let the board size be selectable (3x3, 4x4, 6x6 → turns are 15, 20, 30). The game should recognize if it is ended, and it has to show the name of the winner in a message box (if the game is not ended with draw), and automatically begin a new game.

Description of the task:

Overview

The Pebble game is a strategic two-player board game that requires players to maneuver pebbles across a grid with the objective of pushing their opponent's pebbles off the board. The game ends after a predetermined number of turns, and the winner is determined by the number of remaining pebbles on the board.

Board Setup

The game is played on an n×n grid. Each player has a set of pebbles, one black and one white, totaling n pebbles for each color. The pebbles are placed randomly on the board at the start of the game, with each player owning one color. Game Play Players take turns to move a single pebble of their color either horizontally or vertically. A move also influences the neighboring pebbles in the chosen direction, potentially causing edge pebbles to fall off the board. The game progresses for a total of 5n turns, where n is the dimension of the board.

Objective

The primary objective is to push as many of the opponent's pebbles off the board as possible. The game concludes either when all turns are played or when a player loses all their pebbles.

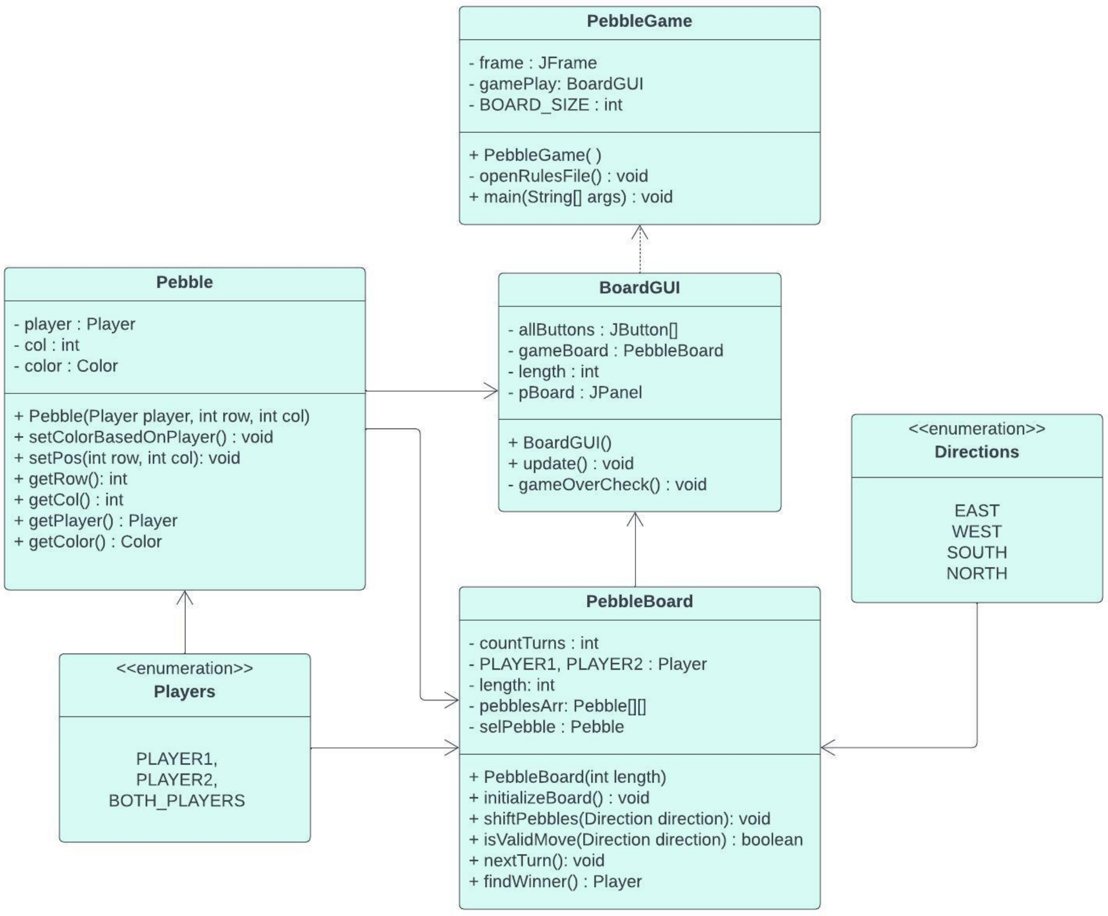
Winning Conditions

The player with the most pebbles remaining on the board at the end of the game is declared the winner. In the case of a tie, where both players have an equal number of pebbles, the game is declared a draw.

Implementation Requirements

The implementation should allow for the selection of different board sizes: 3x3, 4x4, and 6x6. The game must keep track of the number of turns and automatically end the game when the turn limit is reached. At the end of the game, the winner's name (or a draw message) should be displayed in a message box. Upon completion of a game, a new game should begin automatically.

UML Diagram:



Methods description:

A. PebbleGame.Java

PebbleGame(): Initializes the Pebble Game application with default settings and GUI components.

createMenuButton(String label, ActionListener listener): Creates a JButton for menu items, labeled as specified and linked to an ActionListener.

createSizeMenuButton(int boardSize): Generates a JButton to adjust the game's board size to the given value.

openRulesFile(): Attempts to open the game's rules file using the default desktop application.

changeBoardSize(int boardSize): Updates the game's board to a new size specified by boardSize.

main(String[] args): Main entry point for the application; initializes a new instance of PebbleGame.

B. PebbleBoard.java

Constructor- **PebbleBoard(int length):** Initializes a new PebbleBoard instance with a specified board length and sets up the initial game state.

initializeBoard(): Sets up the game board with pebbles for both players and shuffles them for added complexity.

shiftPebbles(Directions dir): Shifts the selected pebble in the specified direction if a pebble is selected.

findWinner(): Determines the winner of the game based on the count of pebbles for each player or the number of turns played.

C. Pebble.java

Constructor - Pebble(int row, int col, Players player): Creates a new Pebble instance with specified row, column, and associated player, and sets its color based on the player.

setColorBasedOnPlayer(): Sets the color of the pebble (black for Player 1, white for Player 2) based on the associated player.

setPos(int row, int col): Updates the position (row and column) of the pebble on the game board.

getRow(): Returns the row position of the pebble on the game board.

getCol(): Returns the column position of the pebble on the game board.

getPlayer(): Retrieves the player (Player 1 or Player 2) associated with the pebble.

getColor(): Gets the color of the pebble.

D. BoardGUI.Java

Constructor - BoardGUI(int length): Initializes a new BoardGUI instance for the game board with a specified length, sets up the GUI components, and attaches key listeners.

initializeButtons(): Creates and configures buttons for the game board, each representing a pebble, and adds them to the game panel.

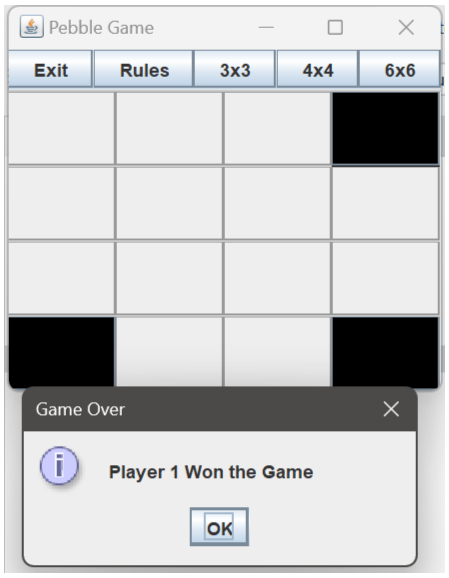
update(): Refreshes the game board UI, updating pebble positions and other visual elements based on the current game state.

Inner Class ListenToButton (implements ActionListener): Handles button click events, updating the game state based on the user's interaction with the game board buttons.

Inner Class KeyAction (implements KeyListener): Manages keyboard interactions, specifically responding to arrow key presses to shift pebbles on the board and update the game state.

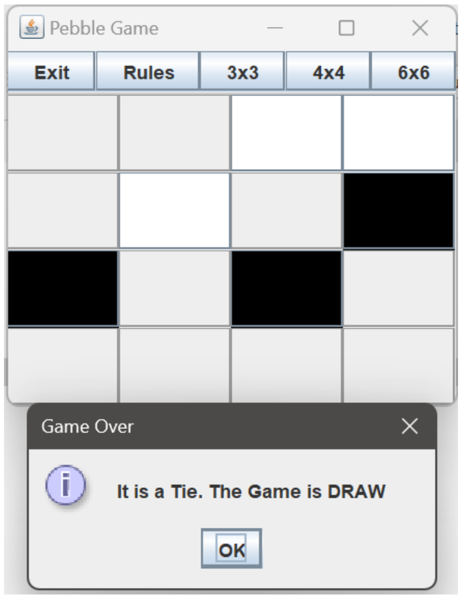
TestCases Description:

Case 1: If PLAYER1 eliminates all of PLAYER2's pebbles before running out of their moves, then PLAYER1 is declared the winner.



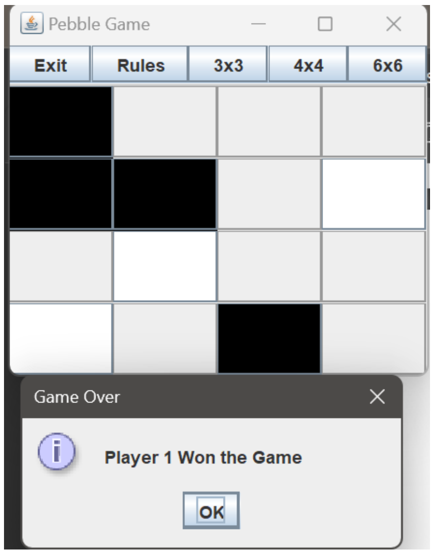
Player 1 Won the Game

Case 2: When both PLAYER1 and PLAYER2 have the same number of pebbles on the board and they have finished their attempts, it's a tie.



It is a Tie. The Game is DRAW

Case 3: If all moves have been made, and PLAYER1 has a greater number of pebbles on the board than PLAYER2, then PLAYER1 is declared the winner of the game.



Player 1 Won the Game