<Linear Domination Game> Software Design Document

CS2300 Section 4 Fall 2021

<Brodie Crawford>

Project Description

This project will create a linear domination game which takes in one file full of data to run the game. The first line will have a value for the size of the board and a value that keeps track of previous turns for validity purposes. After the first line is read, the remaining lines of the input file will have pairs of two integers, meant to be used as points to draw a line on a game board. Each player alternates turns, playing lines and filling cells with the lines on the game board. Cells will be filled by choosing the midpoint of the cells closest to the line (drawn from one cell midpoint to another). If the line played is invalid, the game will not play the line and skip to the next player’s turn. If both players play an invalid perpendicular line consecutively, the game will end and output the final score. Whoever fills the most cells in a single game wins!

Approach

For this assignment, I will use many methods that each accomplish a certain task. The first thing I will do to start this project is create variables holding data for the game, including a Line class. The Line class will construct a Line object that holds an x1 value, an x2 value, a y1 value, a y2 value, and a validity flag I can use to tell if the line played was valid (without deleting invalid lines from my list in main). In main, I will create variables to hold input and output files, the game board, a corresponding array that tells me which cells were marked on what turn, a K variable, variables that keep track of the current turn number and an invalid turn (by perpendicular line), and an array list that will hold each line I read from the file for manipulation purposes. The main method will read the file with a while loop, while keeping track of turns, playing turns, checking them for validity, and exiting to end the game. At the end of the game, the board will output to the console and a file with the cells marked and the player that won the game.

After that, I will create methods to manipulate the lines by playing them on the board if they are valid, as well as displaying the board on each turn with the marked cells. As I go, I will edit the parameters and return types of the methods to make my code modularized and concise, with comments to describe what each method is doing and what the code in each method means. Then – based on the comments I wrote inside my methods – I will attempt to code each method so that it does what I want it to, using parameters passed from main to execute the method. For example, in the method to display the board, I will take the game board, the size of the game board, and the current turn number. With this, I will print the formatted board to the console window along with the player that just took the turn shown using the turn number (odd = player 1, even = player 2). Lastly, I will finish off each method by writing nonessential code such as print and assignment statements.

Detailed Design

Programming Language Used: Java

Editor: Eclipse

Bcrawfo2\_linearDominationGame modules:

- public static void main: executes all of the methods below and implements the core of the game by reading the input file and writing to the console and the output file

- public static int testValidity: Tests the validity of the line being played. Tests for the start and end cells being the same as a line in the past k turns

- public static void eraseCellCoordinatesOfEarliestKTurn: Takes the turn number and uses the turn number key to erase the correct cells from the game board

- public static void playLineOnBoard: Uses an algorithm to find the midpoints of the cells closest to the line being played, and then uses those midpoints to mark the cells on the game board and put the turn number in the turn number key (2D array of integers)

- public static char[][] createBoard: Creates the game board given a size N and initializes all cells to ‘-‘

- public static void displayboard: Displays the board to the console window with nice formatting, and prints out a statement saying which player took a turn

- public static int[] keepScore: Counts the X’s and O’s on the game board and stores the score for each player into an integer array of size 2

- public static boolean boardFull: Checks if the board is full or not

- public static void outputBoardAndWinner: Outputs the game board to the console and an output file with a statement saying which player won and by what score (or that it was a tie if applicable)

Module Sketch:

Diagram, schematic

Description automatically generated

Key Data Structures:

Line – allowed me to make several objects, each with 2 pairs of coordiates and a validity flag

2D arrays – stored the game board and the turn number tracker for the cells

Files – source of my data to run the game

ArrayLists – gave me a structure to store multiple lines in without having to define the size

Buffered writer – helped me write my game data to empty output files

Scanner – helped me access the file stream and put data into lines in my array list