Design for Tetris

Application

The application class is there to define the initial steps of the game. It has a main method that initiates the controller which begins the game.

Instance Methods:

main()

Controller

The controller class is responsible for taking the input of the user and updating the output (the users interface) in the viewer class and also inform and change the model according to the input. Since the game has also an implemented timer this class will be responsible for updating the view and the model based on KeyListeners and actionPerformed. (**extends** JComponent **implements** ActionPerfomed, KeyListener)

Instance Variables:

GameLogic piece – a variable that holds the current piece

int leftWall – a variable for the left border of the board

int rightWall – a variable for the right border of the board

int topBorder – a variable for the top border of the board

int bottomBorder – a variable for the bottom border of the board

ViewBoard newBoard – a variable to hold the new board grid

boolean [][] copiedGrid – holds a row of booleans (either filled or empty positions of the grid) for when a complete row has occurred and the board has to move down

boolean [][] toRow – holds a row of booleans (either filled or empty positions of the grid)

int score – holds the score of the user by the lines that the user has filled

Instance Methods:

void run () – this returns changes after our timer has been set in motion so if no user input then the piece moves one line every unit of time till it gets to the bottom of the board or hit another piece.

void keyTyped () – returns altered piece on board (updated) after user input and after checking if the input is valid (doesn't exit the board borders)

void checkRow – returns an updated board that has moved if a row is full

void endGame – checks whether the copiedGrid has height bigger than the one of the board border thus finishes the game and starts another one

void updateScore – updates the score when the checkRow methods returns a new board

ViewBoard

This class is responsible for building a board grid of specified length and width and showing the instruction text (JLabel) and the layouts (BorderLayouts) thus it extends JPanel. Further this class is responsible for placing the pieces on the right position on the board (like the tic tac toe) it will be printing an updated view of the board every time the piece moves.

Instance Variables:

JLabel instructions – displays the titles
JPanel grid – displays the grid that will be our board
GameLogic piece – holds the information for the constructed piece

Instance Methods:

printBoard – return void a int[][] sizeBoard grid for the board

GameLogic

This class is the constructor for all of the four block tetris pieces (which are seven, T, L, S, Z, I, O, and mirroredL) constructed based on their y and x coordinates of their blocks and on their rotation. The purpose of this class is to be able to return an array of three numbers the x determining the x positions of the blocks the y determining the y values of the x position of the blocks and the z coordinate which will be relative to the x and y coordinates since it will give information for the statue of rotation the piece is currently on. This class is also responsible for methods specific to tetris piece information so if any information for the piece width, height, or orientation is required then the methods defined in this class will return them. Lastly, this class does not hold the pieces it rather is capable of constructing them.

Instance Variables:

GameLogic [][][] piece – holds the information for the constructed piece

Instance Methods:

getWidth(GameLogic piece) – returns int of total piece width

getHeight(GameLogic piece) – returns int of total piece height

getPiece(GameLogic piece) – returns GameLogic [][][] piece of the x, y, and z coordinates of the piece

paintPiece(GameLogic piece) – returns void visible colored piece

nextRotation(GameLogic piece) – returns GameLogic [][][] piece of the x, y, and z +1 coordinates of the piece which will rotate it 90 degrees clockwise

moveToLeft(GameLogic piece) – returns void visible piece moves towards the negative x direction

moveToRight(GameLogic piece) – returns void visible piece moves towards the positive x direction moveDown(GameLogic piece) – returns void visible piece moves towards the negative y direction