Java Tetris

In my Tetris program, I adhered to the principles of object-orientated programming by splitting up the program functionality into a variety of classes. Of the ones created, the main classes are Square, GampePiece, GamePanel, ScorePanel, and GameWindow. All classes are located within a TetrisFiles package.

The Square primary function is to keep track of and update its position. Each Square has an int array, coords, which keeps track of the Square’s coordinates in the array in which it is to be placed during a game of Tetris. Get and Set methods allow the array to be viewed and updated, respectively. The Square class also has methods to move a Square down, left, or right, making the moveDown() method in the GamePiece class more easily implemented.

The GamePiece class is abstract and acts as the superclass for each of the more specific types of GamePieces. GamePiece, though abstract, implements many non-abstract methods which are common to the GamePieces which extend the class. A GamePiece can move down, left or right, by calling the respective method of each of the four Squares that comprise a GamePiece. The GamePiece class also defines the abstract methods for rotating a GamePiece left and right, along with methods to ensure the validity of piece movement and rotation. It even has one method, adjustPiece(), to adjust a piece’s coordinates should a move/rotation result in one of the GamePiece’s Squares being outside the bounds of the board. Each specific type of Tetromino has its own subclass of GamePiece, containing a constructor specifying where the piece is to be placed on the board, which Square is the origin, and what the type of piece is.

The GamePanel class is where the pieces and game board are painted, and additionally contains both the key and mouse listeners which allow the user to move and rotate a GamePiece with the keyboard and mouse.

The ScorePanel is responsible for showing the player’s score, a message indicating the state of the system, and holding a restart button, which, upon being pressed, calls the GameWindow method to reset the board to be completely blank. Should the GameWindow indicate that the game has ended, the button will read “Play Again” instead of “Restart.”

The GameWindow extends JFrame and is made up of one GamePanel and one ScorePanel, the GamePanel located in the centre, the ScorePanel at the bottom of the window. A GameWindow has a Timer, which runs by default every 1000 ms. Each time the Timer fires, the window checks for and removes any full rows, increments the player’s score accordingly, moves the piece down, and, if needed, inserts a new random GamePiece into the board.

Finally, there is a TetrisDriver class with a one-line main function to create a GameWindow object.

Shortcomings of my solution include an overlap in code function and a bug that I was not able to fix. If, as soon as a new GamePiece appears on the screen, the player moves the GamePiece to the left edge of the screen, on occasion, the piece will stop moving down, and new pieces will spawn over each other once or twice before the program recognises the game is over. I was unable to ascertain the origin of this bug, and due to time constraints was unable to fix it.