Tetris

Due Date: Midnight, October 18, 2019

Assignment

In this edition of the Tetris project, you will add several features to the supplied solution to the last project. This should produce a playable version of the game. You will note some changes to the UML class diagram as well as some stub methods in the supplied GamePanel class. Finally, note there is some Javadoc documentation supplied as well. You should consider the descriptions of these methods in the documentation to guide your implementation.

1. Add a 2D array of SmartRectangle objects for the blocks in the play area. Make sure these objects get painted to the screen.

2. Once a piece has finished falling and its components have been added to the 2D array for the play area, your program should check for full rows of SmartRectangles, remove all SmartRectangles in any full rows, and shift any SmartRectangles in rows above the recently-emptied row, down to fill the newly vacated spots.

Handling User Input

You must take input from the space bar. When the user presses the space bar, the Tetrimino which is currently falling should drop as far as it can and add its components to the GamePanel's 2D array.

Game End

Your Tetris program should check for the end of a game (when there is a component rectangle in the top-most row of the play area which is not part of a falling Tetrimino). Your program should stop responding to input from the user and stop all animation on the screen. The user should not be able to move the Tetrimino or restart the animation by pressing the pause key.

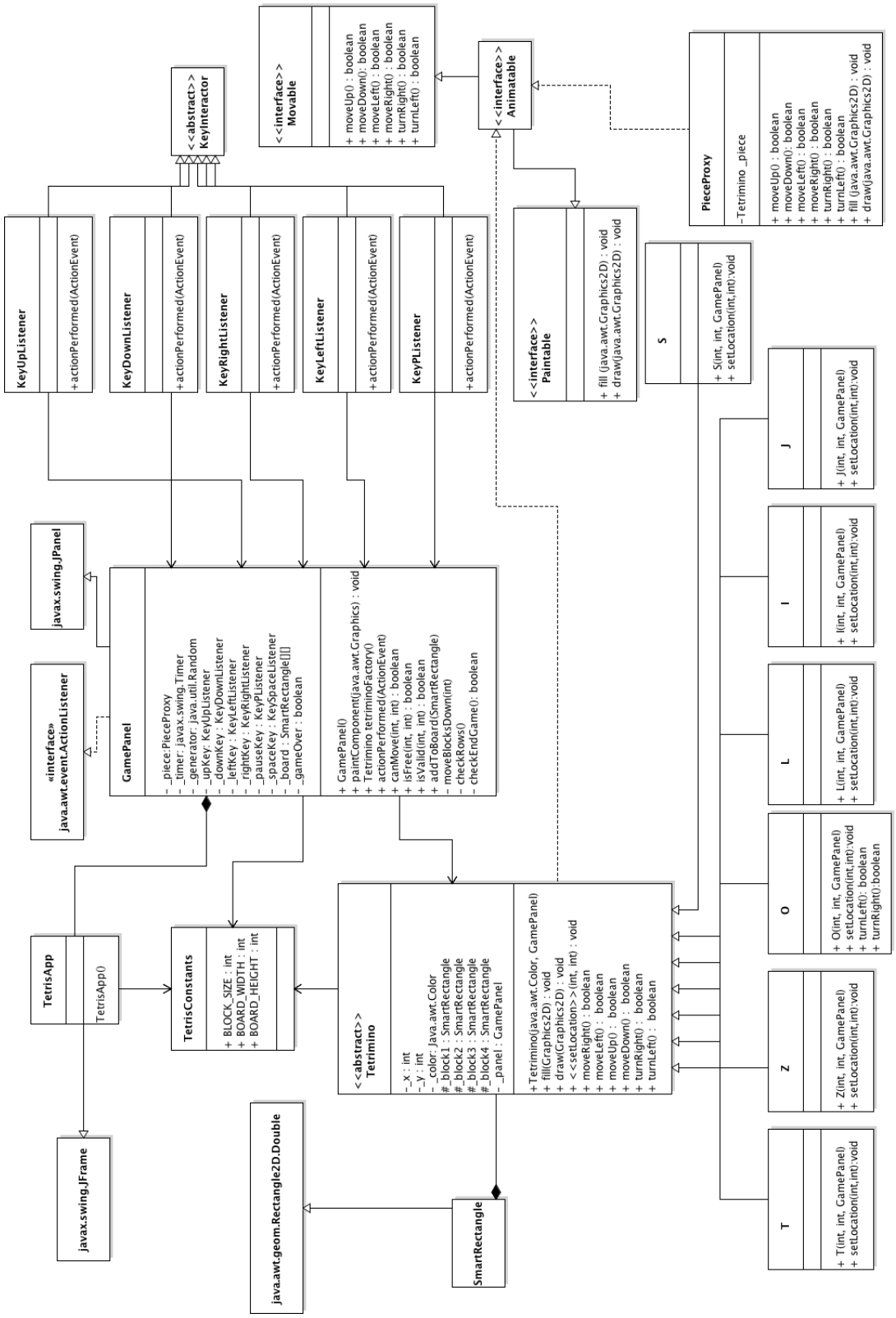

Figure 1: UML Class Diagram for Tetris Deliverable 3

Graphics and the Grid

When you look at the moving pieces, you see that they only are displayed at discrete locations. Moving is done in chunks corresponding to the rows. While you are writing your program, it may be easier to represent the position of squares with array coordinates i.e., their coordinates in the array that would be formed were the board entirely filled with squares. But you will need to be able to translate these array positions to actual positions to be able to display them correctly using fill.

Requirements

1. Tetrimino objects must stop falling and add their components to the 2D array in the GamePanel when they can not move down. Both reaching the bottom of the play area, as well as landing "on top" of one or more SmartRectangles, should initiate adding component rectangles to the GamePanel array.

2. When the user presses the space bar, the currently falling Tetrimino should fall as far as it can and add its components to the play area. Once the space bar has been pressed, the user should not have the opportunity to move or rotate the Tetrimino.

3. Your program should detect the end of the game and then stop the animation and stop responding to user input.

4. Once a piece has finished falling, the program should automatically create a new Tetrimino, animate it falling and allow it to be manipulated via the arrow keys and space bar.

5. No part of the Tetrimino should be able to be moved out of the play area by any means, either through rotation or moving left/right.

Grading

You must use the Git repository created on the UConn GitHub server by your instructor. All code must be shared via that repository. If you do not have access to the Git repository for your team:Log in to the UConn GitHub Server with your NetID and password to get an account set up. This must be done before you can be added to a project. Then, contact your instructor by email to be added to the appropriate repository as a collaborator. Once the project is complete, you must submit a copy of the Peer Assessment sheet and include everyone in your group for this project.