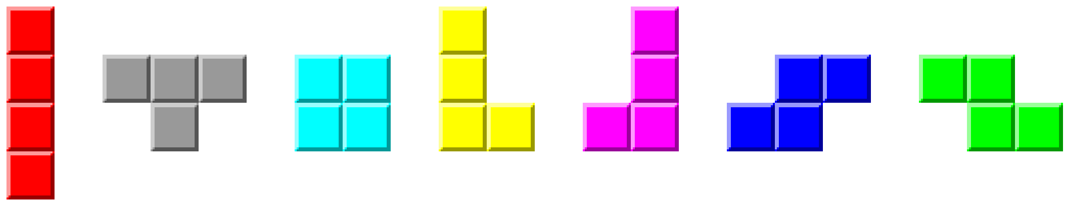
Programming Project 5 – Tetris

Due: Thursday April 28, 2016 at 11:59 pm

For this project, you are to write a Java application to play Tetris using the Java Swing Libraries. This project will be done in groups of 2 or 3 students. Students can determine their own groups. Note that is copyrighted and legal action has been taken against websites hosting Tetris-like games.

A discussion of Tetris can be found at http://en.wikipedia.org/wiki/Tetris. Tetris is a real time game that continuously drops pieces (called Tetrominoes) on to a pile. The user is allowed to rotate and move these pieces left and/or right to fill up rows in the pile. When a row is completely filled, that row disappears and all pieces on top of that row fall down. Over time, the tetrominoes fall faster and the game is over when the pile reaches to the top of the playing area.

There are 7 different Tetrominoes. Each one is made up of 4 blocks. The 7 Tetrominoes are referred to as **I**, **T**, **O**, **L**, **J**, **S** and **Z**. These seven Tetrominoes are shown below:



Your program must create a base class of Tetromino. Each of the seven different types of Tetrominoes must be implemented using a derived class that inherits from the base Tetromino class. Your program must also use Design Patterns in at least two spots. A readme.txt file must be submitted with your program CLEARLY explaining the Design Patterns being used including the implementation details (i.e. samples of code from your program).

The playing area is a grid of 20 rows and 10 columns. When a tetromino is dropped, it is added to the playing area at the top most row and will descend until it cannot go any farther down. The descent rate is controlled by a timer. Every time the timer "goes off", the tetromino will move down one row. This will continue until the time when the timer goes off and the tetromino can't go to the next row because it is either at the bottom row or it is on top of another tetromino. At this time, the tetromino will become locked in place and a new tetromino will be dropped. When the timer first places a tetromino on the bottom row or on top of another tetromino, the tetromino may be moved left or right (assuming there is room of it to move) or maybe even be able to rotate. Your program should randomly select the next tetromino from all seven different kinds.

At every 10th line cleared, the “level” of the game increases. The speed at which each tetromino falls increases and the score for each line cleared increases. This means that you should shorten the timer's delay interval. There are many formulas as to the “proper” delay time. A simple one to use is that closely reflects one implementation is a follows. This delay stops decreasing at level 24.

Delay = (50 – (currentLevel * 2)) / 60 seconds

Your program will need controls to move the tetromino left or right, rotate it left or right, drop the tetromino and pause the game. These controls are to respond to keyboard presses. Examples of key event programs can be found on the Java Example Page. These controls must not need any modifier keys (such as Shift, Control, Alt) to invoke the control. The exact keys used will be left to the programmer. You are also to add buttons to the GUI that will also activate these controls.

A few notes on the movements of the tetromino.

- When the tetromino moves, it cannot overlap with another tetromino in the playing area nor move outside of the playing area. If a move were to cause such an action, ignore the move.
- When a tetromino is rotated, it moves 90 degrees. Thus when a tetromino is rotated 4 times, it is to end up in the same position as when it started (assuming that is doesn't drop down a row while it is being rotated). Don't have the tetromino spin off to the right, left, up or down if you continue to rotate it. The anchor point for each rotation, should be somewhere to the center of tetromino.
- When a tetromino moves, the delay to move to the next line is reset. So the move to the next line only occurs when the player stops spinning the tetromino or moving the tetromino left or right. Thus a player can effectively pause the game by continuously spinning the tetromino.
- When a tetromino is dropped, we will implement a "hard drop" which will instantly drop the tetromino directly down as far as it can go without overlapping another tetromino and lock it in place (so it can not move or spin).
- **For 5 pts extra credit, you can add a second drop key that will do a "soft drop".** A soft drop will quickly move the tetromino down line by line (without waiting for the delay). The user can stop moving the tetromino down at any time and will still be able to spin or move the tetromino after doing a soft drop. The soft drop must allow the user to only move the tetromino down to a desired location. Exact implementation of the soft drop is left to the development team.

In addition to the playing area and the control buttons, your program is to display the following information:

- The player's current score.

  A player's score is increased every time a row is filled in and cleared. If a player fills in and clears multiple lines simultaneously, the more point are scored. The point values are as follows:

One line cleared:	40 points X current level
Two lines cleared:	100 points X current level
Three lines cleared:	300 points X current level
Four lines cleared:	1200 points X current level

  Note: Four lines can only be cleared with the **I** tetromino placed vertically. Your program should only check for filled lines when the current tetromino is locked into place. Some games give some small number of points every time a tetromino locks into place. You may do this if you so desire.

- The next tetromino that will be dropped.
- The number of lines cleared
- The current level of the game
- The number of seconds elapsed since the game started

Your program will need a menu that will perform the following operations.

- Quit the program

  This menu button is in addition the quitting your program by clicking on the X on the right side of the title bar.

- Display an About box

  The About box can be a simple dialog box that gives the name of the program's author, when the program was written and why ("The X-th programming assignment for CS 342").

- Provide help on how to use your program.

  This can be a dialog box that describes the basics of your program. This should include a description of all menu, button and keyboard operations.

- Start/Restart the game.

  This should clear any remnants of a previous game and begin play on a new game. You may wish to include a button and keyboard press to perform this function.

The Playing Area

The exact implementation of the playing area is up to you to determine. Here are some thoughts that I have.

The playing area could be made of a grid of 20x10 labels. I have (at least) two images. One image to show an empty position in the grid. The other image to show an occupied position in the grid. A position may be occupied by the current falling tetromino or was part of a previously fallen tetromino. As the current tetromino falls, moves, rotates, etc., the images contained in the labels in the grid change to reflect the falling, moving, rotating tetromino. Each tetromino will take up 4 different positions in the grid. Here are some programs that I was using to mess around with this idea.

- LabelTest.Java
- LabelTest2.Java
- LabelTest3.Java

That uses these image files.

- bluesquare.jpg

- blacksquare.jpg
- whitesquare.jpg

Here are a list of images that could be used for this program.

Another was is similar to the previous idea, but to draw in a JPanel instead of using JLabels and images. This idea doesn't use images, but requires that the programmer keep track of the coordinate positions of tetrominoes on the screen and to draw/redraw them when needed. Personally, I found that drawing, clearing and re-drawing the tetrominoes with a JPanel to be more difficult.

Gravity

There are two ways to handle the falling of blocks that are on top of row when it gets cleared. One way is to just move all of the blocks down by the number of rows that were cleared. The other way is to have the blocks fall until they land on top of another block. The Wikipedia page on Tetris talks about both of these options. They call the first "naive gravity" and the second "flood fill". The first one is easier for the programmer, while the second makes the game easier for the player. You may use which ever you wish.

However for **10 pts extra credit**, you can implement both forms of gravity and have a menu option that allows the user to switch between the two forms of gravity.

Programming Style

Your program must be written in good programming style. This includes (but is not limited to):

- Use of classes,
- Data hiding,
- Meaningful identifier names,
- A file header comment at the beginning of each source code file,
- A method header comment at the beginning of the method,
- Proper use of blank lines,
- Proper use of indentation,
- Explanatory "value-added" in-line comments, etc.

You are not allowed to share code with any other person outside of your group/team (inside this class or not). You may discuss the project with other groups; however, you may not show any code you write to another group nor may you look at any other group's written code.

Project Submission

You are to submit this project using GitHub.