Double Tetris

For my final project I chose to create a program in which two players could play Tetris simultaneously. The final game displays a screen with 2 Tetris boards, labeled player one and player 2. The falling tetrad on each board is controlled by different key sets on the same keyboard. The game continues until one of the players is unable to continue playing (tetrads have reached the top of the playable board), at this point the player who is able to continue playing is the winner. In my original plan I intended to include a scoring system which would increase the speed of the falling tetrads at certain intervals however I did not have enough time to incorporate this into my final program.

The first step to creating my program was to make a working version of the original Tetris game. In order to get the Tetrads to fall on their own I created a loop which made the tetrad on the screen move down every 10 milliseconds. The step I took was to make the tetrads freeze once they touched the bottom of the game screen. I did this by creating a `freezeTetrad()` method. If `canMoveDown()` is false, I call the `stay()` method which freezes the tetrad in its current location. The second part of this method checks if the game is over, if not a new random tetrad is created at the top of the screen, continuing the game.

```java
public void freezeTetrad()
{
    int k = (int)(Math.random() * 7) + 1;
    int j = (int)(Math.random() * 7) + 1;
    if (!tetrad1.canMoveDown())
    {
        tetrad1.stay();
        if (!endGame())
        {
            tetrad1 = new Tetromino(k, board1);
        }
    }
}
```
//
if (!tetrad2.canMoveDown())
{
    tetrad2.stay();
    if(!endGame())
    {
        tetrad2 = new Tetromino(j, board2);
    }
}

The next element to create a working game of Tetris was to create a method which cleared completed rows. To do this I made a loop which checked each row of the array to see if it was full. If a completed row was found that row was deleted (array values set to 0) and all the elements in the array above the deleted row were moved down. This method gave me some trouble, when if first wrote it rows were being deleted as the tetrad was still falling instead of once the tetrad froze. Eventually I corrected this and got method to work correctly. Below is a segment of my clearRow() method, in the actual program this exact code is duplicated within the same method in order to check the second Tetris game board. The only difference is that board1 is changed to board2.

    public void clearRow()
    {
        for(int i = 0; i < BH; i++)
        {
            if(board1[i][0] > 0 & board1[i][0] < 8)
                if(board1[i][1] > 0 & board1[i][1] < 8)
                    if(board1[i][2] > 0 & board1[i][2] < 8)
                        if(board1[i][3] > 0 & board1[i][3] < 8)
                            if(board1[i][4] > 0 & board1[i][4] < 8)
                                if(board1[i][5] > 0 & board1[i][5] < 8)
                                    if(board1[i][6] > 0 & board1[i][6] < 8)
                                        if(board1[i][7] > 0 & board1[i][7] < 8)
                                            if(board1[i][8] > 0 & board1[i][8] < 8)
                                                if(board1[i][9] > 0 & board1[i][9] < 8)
                                                {
                                                    for(int k = i; k > 0; k--)
                                                    {
                                                        for(int l = 0; l < BW; l++)
                                                        {
                                                            board1[k][l] = board1[k-1][l];
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
Once I had created a working game of Tetris I went to work on creating the two player game. Creating my new game screen took a few different steps. First I doubled the width of the original Tetris screen so I would have room to draw the second game screen. I then altered the drawBorder() to draw two Tetris boards directly next to each other. I also drew text in on each board, making one Player 1’s and the other Player 2’s board.

```java
public void drawBorder()
{
    draw.setPenColor(Color.white);
    draw.line(BW, 0, BW, BH);
    draw.line(0, 0, 0, BH);
    draw.line(0, 0, BW*2, 0);
    draw.line(0, BH, BW*2, BH);
    draw.line(BW*2,0, BW*2, BH);
    draw.text(BW/2, BH-2, "PLAYER 1");
    draw.text(BW*1.5, BH-2, "PLAYER 2");
}
```

To create the second game what I did was create a second array, then insert a new tetrad into that array. One difficulty I ran into was drawing the new tetrad in the correct spot on the on the screen. It took a lot of trial and error but I eventually got it contained within its game board through mainly a process of trial and error.

```java
for (int i = 0; i < BH; i++)
    for (int j = 0; j < BW; j++)
    {
        int t = board2[i][j];
        if (t > 7) t = t - 7;
        if (t != 0)
        {
            draw.setPenColor(tileColor[t-1]);
            draw.filledSquare(j + 10.5, BH - 1 - i + 0.5, 0.48);
        }
    }
```
In order to control both tetrads from the same keyboard I wrote new code to control the second tetrad with the keys “e, s, d, f”. Basically I duplicated the key configurations for the original tetrad, changed the keys, and changed the tetrad which moved.

```java
if (c == 'e') // up
{
    tetrad1.rotate();
}
else if (c == 's') // left
{
    tetrad1.moveLeft();
}
else if (c == 'd') // down
{
    tetrad1.moveDown();
}
else if (c == 'f') // right
{
    tetrad1.moveRight();
}
```

To end the game I created a method called endGame() which returned a boolean value of true if the game was over and false if the game was continuing. The game is considered over if any element in the top row of either game’s array has a non zero value which means there is a frozen tetrad occupying that spot.

```java
public boolean endGame()
{
    int i = 0;
    for (int j = 0; j < BW; j++)
    {
        if (board1[i][j] > 0 && board1[i][j] < 8)
        {
            playerID = 2;
            return true;
        }
        else if (board2[i][j] > 0 && board2[i][j] < 8)
        {
            playerID = 1;
            return true;
        }
    }
    return false;
}
```
If endGame() returns true the drawWinner() method is called which draws a jpeg image showing that the game is over and telling which player won. Within drawWinner() the winning player is determined by the playerID set in the endGame() method.

In order to have a fully functional two player game, I duplicated the code in each method, which one segment of code controlling board1 and the other segment controlling board2. Originally I planed on having two self contained games of Tetris operation independently of each other, however I realized that it was much simpler to have one class the was able to control both Tetris boards simultaneously from which I could create a single DoubleTetris object to play the game. I am happy with how my game turned out although I ran into a few problems along the way.