Thomas Maney  
Cs101 final project: Tetris  
For the final project, I created a functional Tetris game using and expanding the skeleton code Tetris and Tetromino classes. The original skeleton code, when compiled and run, had the program create a random tetrad piece, which was movable around the screen, left right and down, and could be rotated. With the board and pieces represented in an array, the first function to implement, therefore, was getting the tetrad, once it hit the bottom, to stay there permanently and create a new tetrad. This was done by implementing the function called stopMove, which checks whether the tetrad can move down, using the function in the Tetromino class. If the tetrad cannot move down, the function checks the array for the certain tetrad and marks those cells with the id, which is a number 1-7. For the tetrad to be able to move down, the array cells have to be marked with 0 values, which indicates an empty row, otherwise they would overlap other pieces.

```java
if(tetris.tetrad.canMoveDown() == false)
    for(int i=0; i<BH; i++){
        for(int j=0; j<BW; j++){
            int t = board[i][j];
            if (t > 7) {
                t = t - 7;
                board[i][j] = 0; }
            if ( t <= 7 && t>=1) board[i][j]= t;
        }
    }
```

Once the tetrad stops and is permanently placed, this function calls another function, createNewTetrad(), which was one of the functions in the skeleton code that generates a random tetrad at the top of the board and lets the user move it and maneuver in the game. Also, each time the tetrad is placed by the stopMove function, the function also calls the function called checkRow, which is used to clear lines when a row is full and is an integral part of the game. For this reason, stopMove has several input values and a return value, that it passes on to checkRow. These values include the score and level variable which is used with the scoreboard function and elsewhere in the main method.
Another aspect that had to be implemented into the code was a way for the tetrad pieces to fall constantly at a certain rate, once created. This was simply done by adding the code `tetris.tetrad.moveDown();` into the main while loop. This made the piece fall down constantly once created, which is a critical aspect of tetris.

The next function that had to be implemented for a functional tetris game was one that, when a row is full and taken up by tetrad pieces, clears the row and moves everything else above it down. This is an integral part of the tetris game and really how the game is played. The objective of the game to clear and delete as many lines as possible by maneuvering and rotating the tetrads to fill rows. This checkRow function is called each time a tetrad hits the bottom through the stopMove function. This function checks each row in the array for whether the value in the array is not equal to zero, which would signify a tetrad occupying that space.

```java
for(int i=0; i<BH; i++) {
    int p=i;
    int check=0;
    for(int j=0; j<BW; j++) {
        if(board[p][j]!=0) check=check+1;
        if(check==10)
            // This segment of the checkRow function shows the For loops used for checking the row and column of the array. The row value in each iteration is stored in the variable p, and then the j column iteration starts, which makes sure that for each row every column value is checked before the next row iteration. If a value in the array is not zero, the variable check counts up. The maximum columns and cells in each row is 10, so if the variable check gets to 10, it means that the full row, p, is filled. Using an if statement, the function uses another for loop to delete the row and move everything else down:
            for(int u=i; u>0; u--){
                for(int o=0; o<BW; o++) {
                    board[u][o]=board[u-1][o];
                    // This was initially difficult to figure out, since I thought the array values had to be set back to zero before moving everything else down. However, this for loop takes the row i from
```
the previous row iteration and for each column value, the array cell is set to the value of the array cell in the row above it. The row iteration counts down to zero, rather than up, since the top of the board is the array row 0, so these for loops set each value to the row above it, which moves everything down and deletes the filled row. Also in the if statement, a score variable counts up for each row deleted. The function draw is also called when this happens, with the score variable as an input, which deals with the scoreboard discussed later. Another thing that happens each time a row is deleted and this if statement is true, a sound is played, which goes with the row deletion on the actual game board. This is done using StdAudio and a .wav file.

The next function I had to implement was one that ended the game, called gameover. This function merely uses a for loop at row zero and checks whether a newly created tetrad can move down. If it cannot, once created and displayed at the top of the board, cannot move down this could only be that the blocks of tetrads got to high for it to move down. When this happens in tetris, the game ends, so in the function it returns a Boolean value true, which is used in the main method to break the loop of the game. When this happens, game over is printed at the top of the board, as well as in the command prompt/terminal.

```java
if(tetris.tetrad.canMoveDown()==false) {
    for(int j=0; j<BW; j++){
        int t= board[0][j];
        if(t!=0)
            return true;
    }
}
```

These three functions make up the main part of the game that had to be implemented to make it functional as a tetris game. Although each function does not contain much code, they do a lot to make the game functional and are the basic parts of the tetris game that had to be implemented and added to the skeleton code.

Another aspect that I added apart from the main functionality was a scoreboard. This is displayed at the top of the game and shows the two variables, level and score. The scoreboard
could have been displayed in a new draw window or jframe, but was more convenient and functional for the user when displayed at the top. To do this, the variables score and level are inputted into functions checkrow and stopMove. When a row is deleted, the draw class is called with the input of score and level, which displays the new score and level each time the row is deleted and the score counts up. The scoreboard uses the draw class and the code for drawing the scoreboard is in the draw function, which was in the skeleton code but modified to have the scoreboard:

```java
public void draw(int score, int level, boolean go) {
    draw.clear(Color.black);
    draw.text(BW/2, BH+1.2, "Tetris Scoreboard");
    if(go==true) {
        draw.setPenColor(Color.red);
        draw.text(BW-2, BH+.8, "GAMEOVER");
    }
    //SCOREBOARD DISPLAY
    draw.text(BW/2, BH+.7, "score=" +score);
    draw.text(BW/2, BH+.3, "level " + level);
}
```

The above code shows the draw class used in the draw function to draw the scoreboard and update it with the new score and level. The way I implemented the score was for each line, the user gains one point. This was personal preference, but it could have also been done giving other increments like 100 points for each line clear. This is a matter of user preference rather than any additional code that was needed.

Another goal that I had was to implement levels, with higher difficulty, based on the speed of the tetrads falling. This is done using a somewhat recursive method. Every time the score gets to a certain level, in this case 6 for easier demonstration, the main loop is broken and a new function is called with inputs level and speed, which counts the level up one and the speed variable down, which increases the speed.

```java
if(score==6) {
    tetris.newGame1(speed, level);
}
```
The new game function newGame1 is the main method used again, so a new draw window is opened and the new level starts. In this particular game, the speed starts at 300, and each level counts down by 100 as long as it is above 0, so there are 3 levels. At the end of these three levels, if the user has gained the required score to move on, a new draw window displays a game winning message at the end of level 3. More levels could be made by decreasing the increment of the speed. The new game function is as follow, which also shows the main method.

```java
public void newGame1(int speed, int level) {
    Tetris tetris = new Tetris();
    tetris.createNewTetrad();
    int score = 0;
    level = level + 1;
    tetris.tetrad.move();
    speed = speed - 100;
    while(true && speed>0) {
        tetris.clearBoard(tetris);
        tetris.drawBorder();
        tetris.tetrad.move();
        boolean go = tetris.gameover(tetris);
        score =tetris.stopMove(tetris, score, level, go);
        tetris.draw(score, level, go);
        tetris.tetrad.moveDown();
        //GAME WINNER SCREEN
        if(score==5 && level==3) {
            Draw Win = new Draw(200,200);
            Win.clearColor(Blue);
            Win.setPenColor(Color.orange);
            Win.setScale(0,BH);
            Win.text(BW/2,BH/2, "You won the game");
            break;
        }
        if(score==5) {
            tetris.newGame1(speed, level);
            break;  }
        if(go==true){
            System.out.println("gameover");
            break;
        }
    }
}
```

This uses some recursion, in that the input level increases and speed counts down until the base case, which is speed=0 or the while loop is broken. Although there is only one new
game function, multiple levels can be made only by changing the increment by which the speed counts down per level.

Some difficulties with this project included audio and start menu. I tried to make background music continually play but the .wav file caused a heap memory error when the game was played. The sound being played for each line cleared, however, demonstrates the ability to put sound into the game. Also a gui start menu that lets the user wait to start the game until ready, which waits until a button is pressed was going to be used, but my computer could not handle it having the main method called through a jframe and button. Also for scoring, a timer could have been used, but just using score seemed more sensible.

In conclusion, this final project was a fun learning experience that taught me a lot about and let me test my knowledge of java programming. The ability to use and declare variables is shown, through the declaration of various variables that are used including the score and level functions. Also loops are also greatly used in this project, with the checkrow and stop move functions, which show the use of nested for loops. A while loop is also demonstrated in the main method and new game function. In these while loops, the loop is continuous and only broken when the game ends or a new level is started. Break statements are used in conjunction with if statements to break the loops. In doing this project, I also showed knowledge of functions, since many functions are used, including the static main method, ones with void returns and functions with return values, including the game over function, which has a Boolean return value.

Although I did not fully write the program, but used the skeleton code to implement most of the methods, this game demonstrates the use of data types and classes. The tetris data type was used multiple times, as well as a draw type. Several of the functions implemented the separate draw or tetromino classes as well and used them in the main method of the tetris game. Overall, this tetris
game shows the knowledge of the programming java skills that we learned this semester in cs101, including recursion, functions, variables, and others, with the exception of maybe gui and linked list. I think this tetris game is a good functional game and although it is somewhat simple in its graphics and appearance, it demonstrates good knowledge of the basic computer programming skills we learned. Further improvements could include improvement of the graphics and more gui.