Vietnamese National University HCMC

International University

** **

**REPORT PROJECT**

**\_ Tetris \_**

**Group 24:**

Đỗ Đông Quân ITITIU19043

Đào Huỳnh Thiên Long ITITIU19026

Võ Huỳnh Huy Thịnh ITITIU19055

**Course: OBJECT – ORIENTED PROGRAMMING**

**2020 – 2021**

Contents

1. [Introduction 2](#_bookmark0)
2. [Property of Tetris Game 3](#_bookmark1)
   1. [Goal 3](#_bookmark2)
   2. [Rule 3](#_bookmark3)
3. [Literature 3](#_bookmark4)
   1. [Class 3](#_bookmark5)
   2. [Object 4](#_bookmark6)

[Object Declaration 4](#_bookmark7)

* 1. [Other functions 5](#_bookmark8)
     1. [Star game 5](#_bookmark9)
     2. [Game cycle 5](#_bookmark10)
     3. [Draw Square 5](#_bookmark11)
     4. [Set Color 6](#_bookmark12)
     5. [Key Pressed 6](#_bookmark15)
     6. [Mouse Pressed 6](#_bookmark16)
     7. [Set Random Shape 7](#_bookmark17)

1. [UML Diagram of Tetris Game 8](#_bookmark18)
2. [Methodology 9](#_bookmark19)
3. [Main Class – Function – Explain 11](#_bookmark20)
4. [Shape Class 11](#_bookmark21)

[Check collision Algorithm 12](#_bookmark22)

[Checkline Algorithm 14](#_bookmark23)

[Rotate Algorithm 15](#_bookmark24)

1. [Board Class 16](#_bookmark25)

[Move Shape Algorithm 17](#_bookmark26)

1. [Result – Limited – Conclusion 18](#_bookmark27)
2. [Result 18](#_bookmark28)
3. [Limited 19](#_bookmark29)
4. [Conclusion 19](#_bookmark30)

# Introduction

Nowadays, with the speed up rapidly of Software Technology industry, the high level in programming skill is more necessary. So, the classic Procedural-Oriented Programming method cannot be responsed completely for all requirements. That leads the invention of a new method follow the principal ***Alan – Kay*** named **“Object – Oriented Programming”** to overcome this case.

This project was designed based on Object – Oriented Programming method by Java language. So this has solved some problems that was occur while building by common Procedural – Oriented Method:

* + - The code is clear, easy to understand and concise.
    - The project is a unified logic system by many related - Classes combine together.
    - Each Class has many Methods which take on different behaviors of own class.
    - The ability of reusing the resources.

The purpose of this project is to design a basic game that was built on foundation of object – oriented. Our group decided to modify a game named “Tetris” follow the principle of this method. This game will clearly show the object – oriented property, the combination of classes, between class and object. Tetris is a game require the user to arrange different shapes of cubes in order to there is no empty in a line. Although Tetris has appeared for a long time, our Tetris has been developed with new design, speed of the game, not require the configuration of the computer and specific is this game is build based on the Object – Oriented Method.

Moreover, besides the demand we have mentioned above, we also want to create this project to practice all the techniques we have acquired during this course such as working with Class, Object and how to implement the problem by OOP method logically, etc.. Furthermore, we event want to improve ourselves by the application of new algorithms to make our software run more effectively. Although we are fully aware of the risk and some difficulty when we utilize some knowledge outside the university syllabus, we still want to challenge ourselves to get acquainted with the real pressure in the workplace in the future. To put it another way, the bottom line here is that we desire to make a step further on our way to ultimately become an advanced and professional software developer.

## Property of Tetris Game

## Goal

The aim in Tetris is simple. You will have random blocks go down from the top of the screen. You can move the blocks around, either left to right and/or you can rotate them become 4 different forms. The blocks fall at the center rate, but you can make them fall faster if you are sure of your positioning. Your objective is to get all blocks to fill all empty space in a line at the bottom of the screen; whenever you do this, you will find that blocks vanish, and you get awarded point for each line.

## Rule

Tetris has very simple rules: you can only move the pieces in specific ways; your game is over if your pieces reach the top of the screen; and you can only remove pieces from the screen by filling all the blank space in a line.

## Literature

## Class

In Object – Oriented Programming, a **class** is a blueprint for creating **objects** (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods)

The user – defined objects are created using ***class*** keyword. The class is a blueprint that defines a nature of a future object. An **instance** is a specific object created from a particular class. Classes are used to create and manage new objects and support **inheritance** – a key ingredient in object – oriented programming and a mechanism of reusing code.

### Class Declaration

**public class** < class name > **{**

<list of instance variables>

<list of methods>

**}**

## Object

An Object is nothing but a self – contained component which consists of methods and properties to make a particular type of data useful. Object determines the behavior of the class. When you send a message to an object, you are asking the object to invoke or execute one of its method.

From a programming point of view, an object can be a data structure, a variable or a function. It has a memory location allocated. The object is designed as class hierarchies.

### Object Declaration

ClassName ReferenceVariable = **new** ClassName ();

## Other functions

* 1. **Start game**

var game = new Tetris();

* 1. **Game cycle**

private class GameCycle implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

doGameCycle();

}

}

private void doGameCycle() {

update();

repaint();

}

* 1. **Draw Square**

private void drawSquare(Graphics g, int x, int y, Tetrominoe shape) {

…..

}

* 1. **Set Color**

Color colors[] = {new Color(204, 102, 102),

* 1. **Key Pressed**

public class TAdapter extends KeyAdapter {

…….

@Override

public void keyPressed (KeyEvent e) {

…….

ase KeyEvent.VK\_P -> pause();

case KeyEvent.VK\_LEFT -> tryMove(curPiece, curX - 1,curY);

case KeyEvent.VK\_RIGHT -> tryMove(curPiece, curX+1, curY);

case KeyEvent.VK\_DOWN -> tryMove(curPiece.rotateRight(), curX, curY);

case KeyEvent.VK\_UP -> tryMove(curPiece.rotateLeft(), curX, curY);

case KeyEvent.VK\_SPACE -> dropDown();

case KeyEvent.VK\_D -> oneLineDown();

* 1. **Set Random Shape**

void setRandomShape() {

var r = new Random();

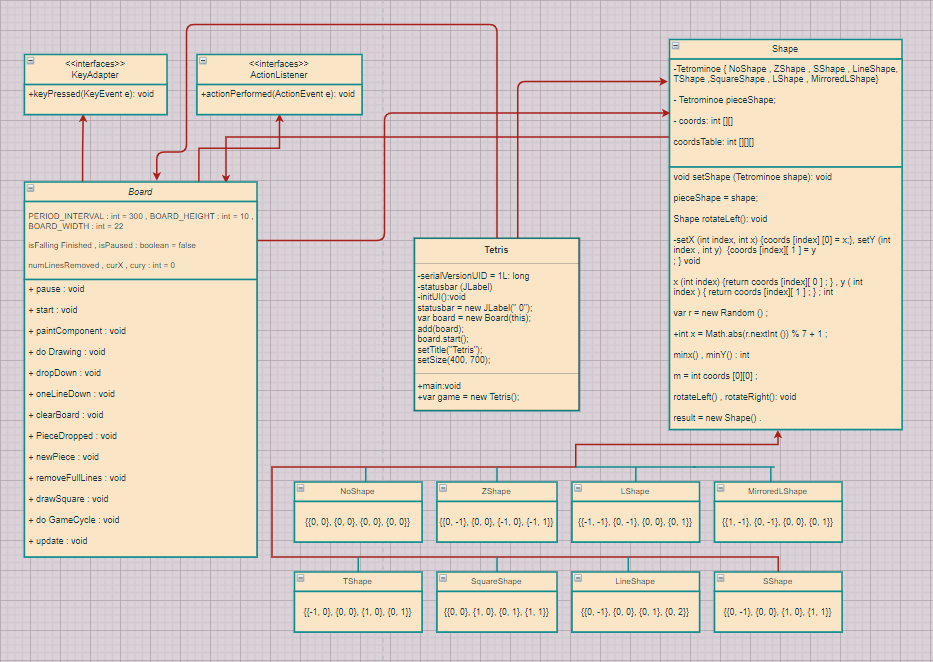
int x = Math.abs(r.nextInt()) % 7 + 1;

Tetrominoe[] values = Tetrominoe.values();

setShape(values[x]);

}

# UML Diagram of Tetris Game



* 1. **Methodology**

This project is combined by 3 classes. There are Board, Shape, Tetris. We build the game based on object – oriented method so we focus on Shape class and Board class – 2 main classes of this project.

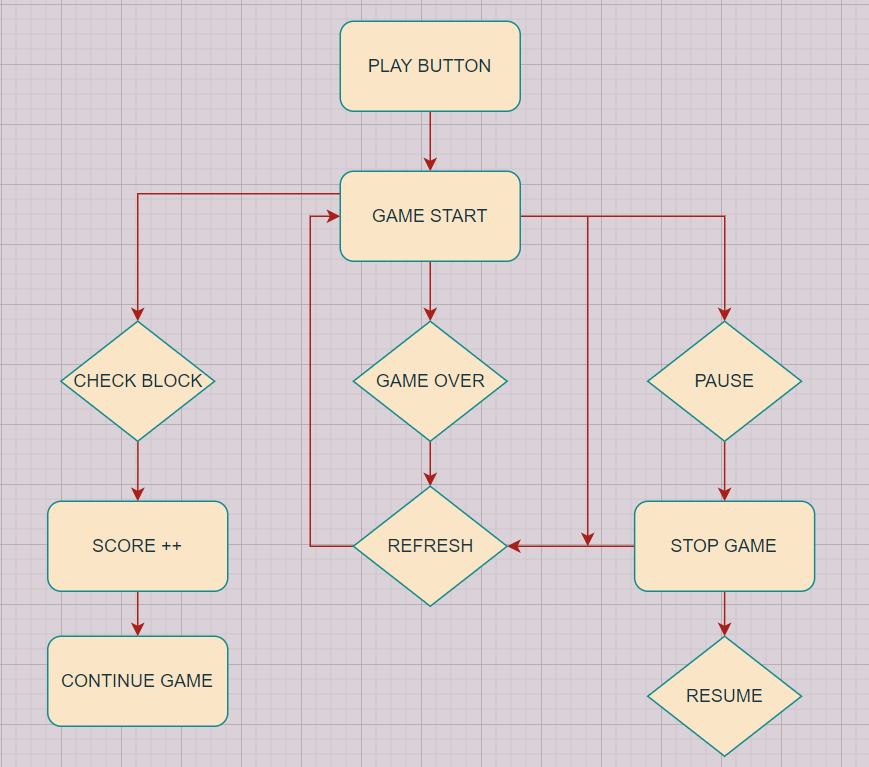
First, we create class Shape their properties and behaviors, and there are 7 objects - shapes that implements from Shape class. (Z, J, O, T, L, I, S). Each object was inheritance from parent Shape and can be reused the behaviors. The Shape can be updated, rotated, collided, create new random shape, …

Next, we built class Board inheritance from Shape Class and Interface Class to make up the Tetris Game, control the blocks, the board class does most of work:

* Control a Tetris board, start game, stop game, random the next shape.
* Increase the score if there is full row on the board.
* Support for the common operation that a client “player” module needs to build a GUI version of the game: add pieces to the board, let pieces gradually fall downward, detect various conditions about the board.

After that, we built the Tetris class to connect the user with the game. Tetris class is inheritance from Interface to control the button. When the user presses the button “F11” on keyboard is appeared to start the game.

Finally, we connect all the main class to a logically system program a main function in Tetris class to run the game.

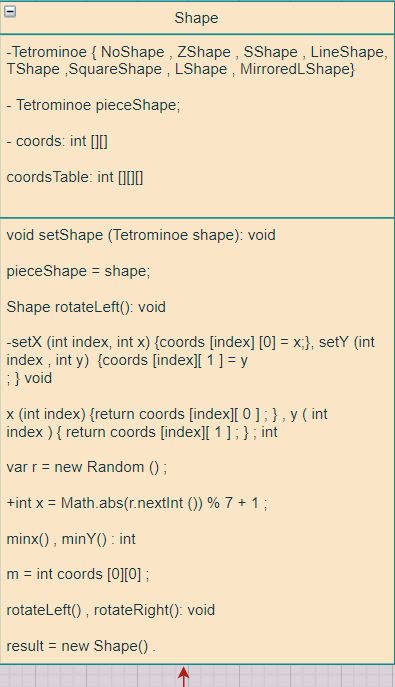


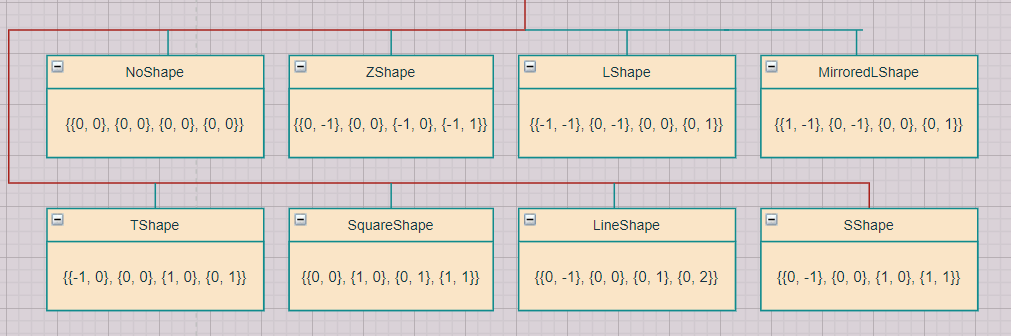
***Algorithm of the Tetris Game***

**INTERNATIONAL UNIVERSITY OBJECT – ORIENTED PROGRAMMING**

# Main Class – Function – Explain

1. **Shape Class**





The Tetrominoe enum holds seven Tetris shape names and the empty shape called NoShape.

protected enum Tetrominoe {

NoShape, ZShape, SShape, LineShape,

TShape, SquareShape, LShape, MirroredLShape

}

void setShape(Tetrominoe shape) {

int[][][] coordsTable = new int[][][]{

{{0, 0}, {0, 0}, {0, 0}, {0, 0}},

{{0, -1}, {0, 0}, {-1, 0}, {-1, 1}},

{{0, -1}, {0, 0}, {1, 0}, {1, 1}},

{{0, -1}, {0, 0}, {0, 1}, {0, 2}},

{{-1, 0}, {0, 0}, {1, 0}, {0, 1}},

{{0, 0}, {1, 0}, {0, 1}, {1, 1}},

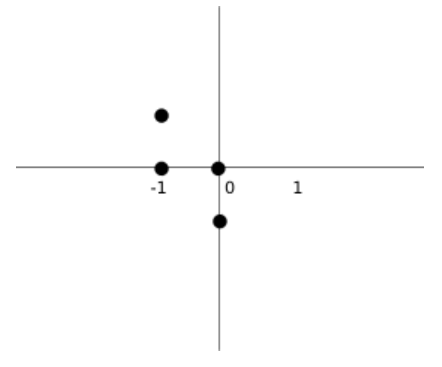
{{-1, -1}, {0, -1}, {0, 0}, {0, 1}},

{{1, -1}, {0, -1}, {0, 0}, {0, 1}}

};

The coordsTable array holds all possible coordinate values of the Tetris pieces. This is a template from which all pieces take their coordiate values.

The following image will help understand the coordinate values a bit more. The coords array saves the coordinates of the Tetris piece. For example, numbers (-1, 1), (-1, 0), (0, 0), and (0, -1) represent a rotated S-shape. The following diagram illustrates the shape.



void setRandomShape() {

var r = new Random();

int x = Math.abs(r.nextInt()) % 7 + 1;

Tetrominoe[] values = Tetrominoe.values();

setShape(values[x]);

}

In this random part, we will program shapes so that shapes can fall randomly without following any rules or arrangement.

Shape rotateLeft() {

if (pieceShape == Tetrominoe.SquareShape) {

return this;

}

var result = new Shape();

result.pieceShape = pieceShape;

for (int i = 0; i < 4; i++) {

result.setX(i, y(i));

result.setY(i, -x(i));

}

return result;

}

This code rotates a piece to the left. The square does not have to be rotated. That's why we simply return the reference to the current object. Looking at the previous image will help to understand the rotation.

# Board Class

# 

In class board, we programmed the operation way of Tetris game. For example, draw a game board, how to fall and locate the shapes, pause, resume game, scoring ...

We have four constants. The BOARD\_WIDTH and BOARD\_HEIGHT define the size of the board. The PERIOD\_INTERVAL constant defines the speed of the game.

The isFallingFinished determines if the Tetris shape has finished falling and we then need to create a new shape. The isStarted is used to check if the game has started. Likewise, the isPaused is used to check if the game is paused. The numLinesRemoved counts the number of lines that we have removed so far. The curX and curY determine the actual position of the falling Tetris shape.

private void pause() {

isPaused = !isPaused;

if (isPaused) {

statusbar.setText("paused");

} else {

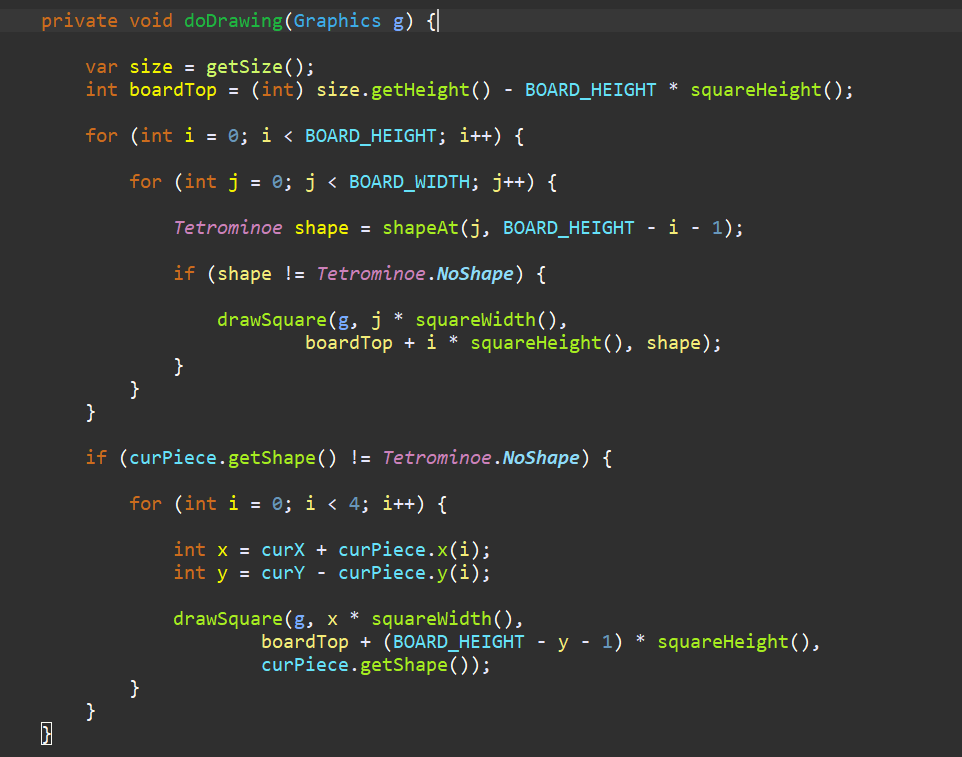
statusbar.setText(String.valueOf(numLinesRemoved));

}

repaint();

}

The pause() method pauses or resumes the game. When the game is paused, we display the paused message in the statusbar.



Inside the doDrawing() method, we draw all objects on the board. The painting has two steps.

In the first step we paint all the shapes or remains of the shapes that have been dropped to the bottom of the board. All the squares are remembered in the board array. We access it using the shapeAt() method.

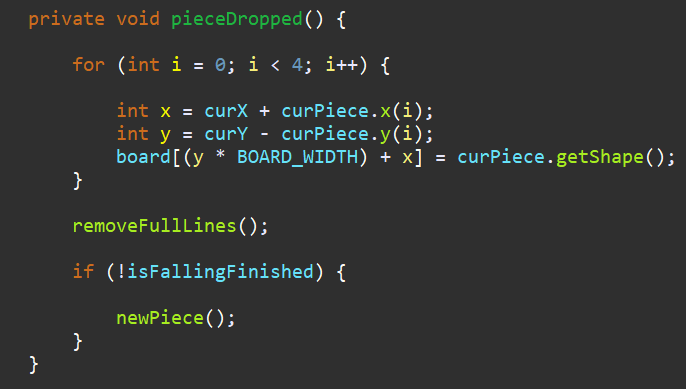
In the second step, we paint the actual falling piece.



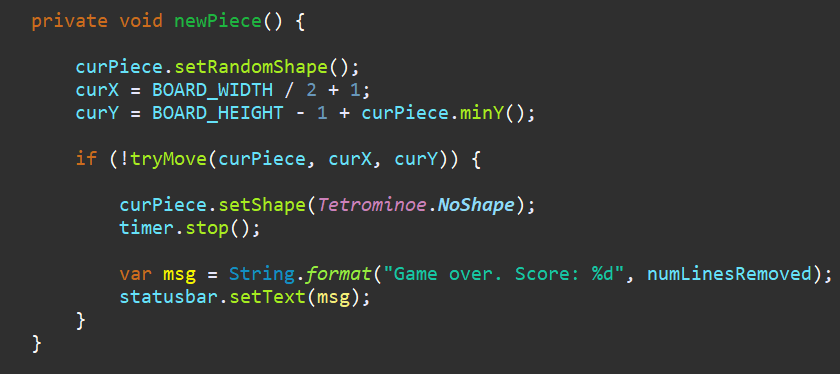
If we press the Space key, the piece is dropped to the bottom. We simply try to drop the piece one line down until it reaches the bottom or the top of another fallen Tetris piece. When the Tetris piece finishes falling, the pieceDropped() is called.

In the oneLineDown() method we try to move the falling piece down one line until it is fully dropped.

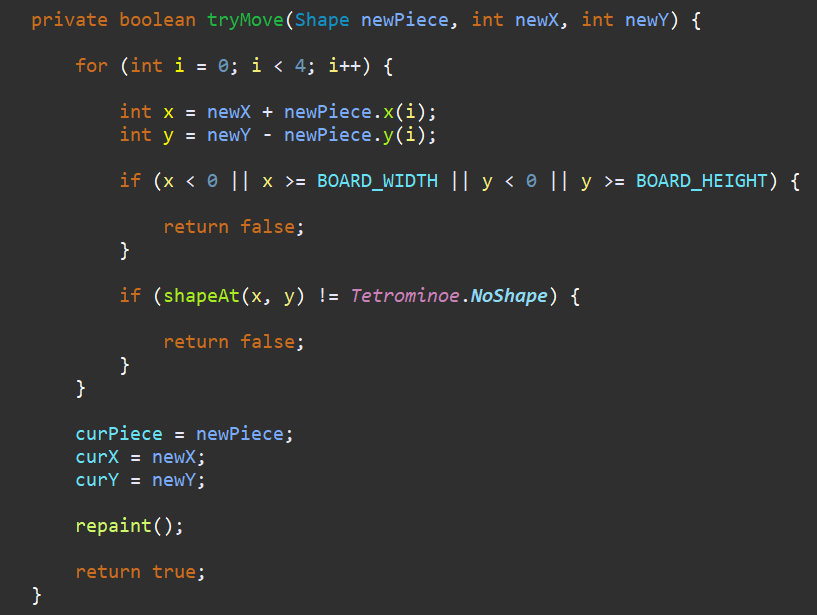
The clearBoard() method fills the board with empty Tetrominoe.NoShape. This is later used at collision detection.



The pieceDropped() method puts the falling piece into the board array. Once again, the board holds all the squares of the pieces and remains of the pieces that has finished falling. When the piece has finished falling, it is time to check if we can remove some lines off the board. This is the job of the removeFullLines() method. Then we create a new piece, or more precisely, we try to create a new piece.



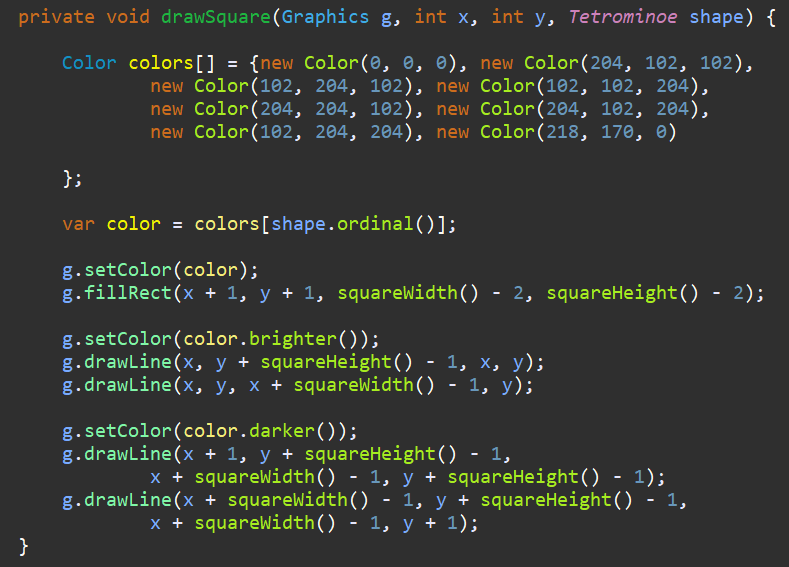
The newPiece() method creates a new Tetris piece. The piece gets a new random shape. Then we compute the initial curX and curY values. If we cannot move to the initial positions, the game is over—we top out. The timer is stopped and we display Game over string containing the score on the statusbar.



The tryMove() method tries to move the Tetris piece. The method returns false if it has reached the board boundaries or it is adjacent to the already fallen Tetris pieces.

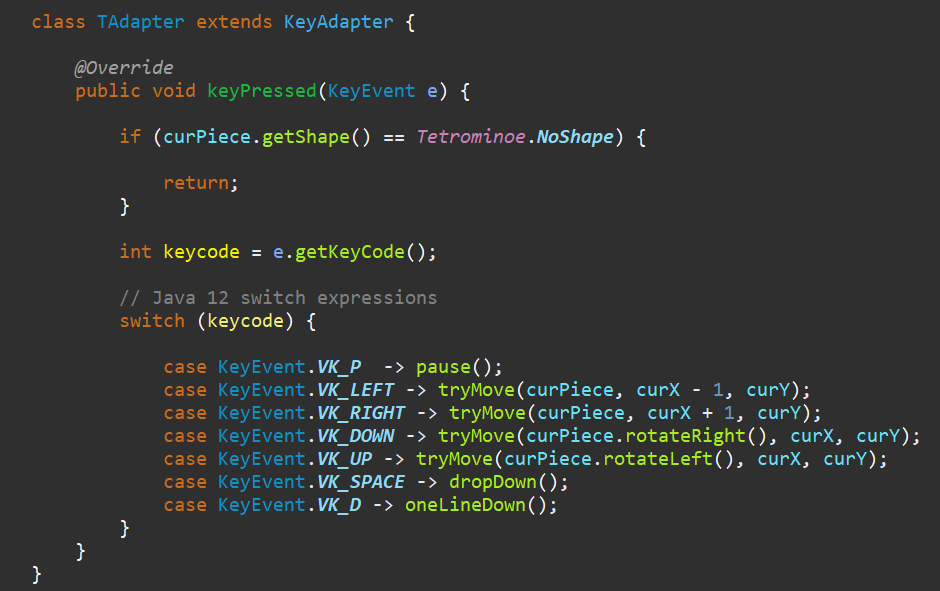


Inside the removeFullLines() method we check if there is any full row among all rows in the board. If there is at least one full line, it is removed. After finding a full line we increase the counter. We move all the lines above the full row one line down. This way we destroy the full line. Notice, that in our Tetris game, we use so called naive gravity. This means that the squares may be left floating above empty gaps.



Every Tetris piece has four squares. Each of the squares is drawn with the drawSquare() method. Tetris pieces have different colours. The left and top sides of a square are drawn with a brighter color. Similarly, the bottom and right sides are drawn with darker colours. This is to simulate a 3D edge.

The update() represents one step of the game. The falling piece goes one line down or a new piece is created if the previous one has finished falling.



The game is controlled with cursor keys. We check for key events in the KeyAdapter.

We get the key code with getKeyCode() method.

With Java 12 switch expressions, we bind key events to methods. For instance, with the Space key we drop down the falling tetris piece.

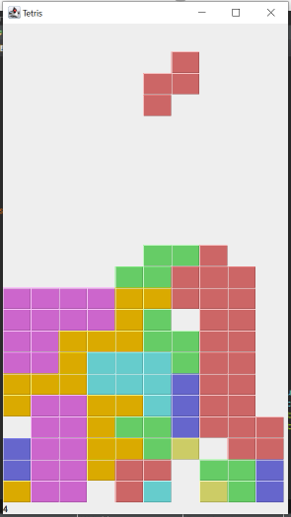
# Result – Limited – Conclusion

## Result

Based on principle of OOP Method, our project has been completely with basic rules and properties of Tetris game. The combination of classes and objects in system is relative logically. We also do successfully the display to connect the user with program, it can be control by other input devices like: mouse, keyboard.

Moreover, we will not show which piece the next piece is for players to develop their ability to brainstorm, solve problems.

CLOSE



SCORE

## Limited

Beside the success of buil the game with basic rules, our project still has many case that cannot be solved:

* + Do not have Save Game, Save Score, input the Name for the user.
  + This game just for 1 player, still have not modified for 2 players.
  + There is only 1 level. We have not yet built more upgrade levels.

## Conclusion

Tetris Game that was built by object – oriented method is easier and logically than traditional – method. This clearly shows polymorphism, inheritance, encapsulation, data abstraction of OOP and the relationship between Shape, Board or other shapes together is linked tightly and systematically. Besides that, learning more knowledge out of the limited of this course is one of the important things to do while performing this project.