西安电子科技大学

Java 程序设计 课程实验报告

实验名称 自主程序开发

经济与管理 学院 2106011 班

姓名 赵红玉 学号 21069100225

同作者____________________

实验日期 2023 年 12 月 12 日

成 绩

指导教师评语： 实验内容： 实验效果： 实验报告： 考勤情况： 其他情况： 指导教师： ______年____月____日

实验报告内容基本要求及参考格式

一、实验目的及小组分工情况

二、实验所用仪器（或实验环境）

三、实验方案设计

四、实验过程记录

五、实验结果分析及实验心得

俄罗斯方块游戏设计

一、实验目的

由学生自主命题开发 Java 程序，锻炼学生分析现实需求、解决实际问题的综合编程开发能力。

二、实验所用仪器（或实验环境）

IntelliJ IDEA

三、实验方案设计

选择项目为俄罗斯方块游戏，在创建俄罗斯方块游戏时，通常会涉及三个主要的类：Shape、TetrisGame 和 TetrisWindow。
1. Shape 类代表俄罗斯方块游戏中的形状（方块），是一个抽象类。Shape 类定义形状的属性（如坐标、颜色等）和行为（如移动、旋转等），包含方法来生成随机形状、检查碰撞、旋转等等。
2. TetrisGame 类代表整个俄罗斯方块游戏的逻辑。TetrisGame 类包含游戏的状态（如当前方块、游戏区域、得分等）和游戏规则的实现，包含方法来处理用户输入、更新游戏状态、检查碰撞、计算得分等等。
3. TetrisWindow 类代表俄罗斯方块游戏的窗口界面。TetrisWindow 类是一个图形用户界面（GUI）类，用于显示游戏区域、方块、得分等，并处理用户交互，包含方法来绘制界面、处理鼠标/键盘事件、更新游戏区域的显示等。
TetrisGame 类负责游戏逻辑，包括生成随机方块、处理用户输入、更新游戏状态等。
Shape 类代表游戏中的方块，包括方块的属性和行为。
TetrisWindow 类是游戏的图形界面，负责显示游戏区域、方块和得分，并处理用户交互。

四、实验过程记录

1. 创建 Shape 类：

```
package Tetris;
import java.util.Random;
```

1.1protected enum Tetrominoe { ... }内部枚举类，用于表示方块的不同形状。

```
public class Shape {
    protected enum Tetrominoe {
        NoShape,
        Zee,
        Ess,
        Long,
        Stack,
        Square,
        El,
        Jay
    }
```

1.2pieceShape 表示方块的当前形状（使用 Tetrominoe 枚举类），coordinates 是一个二维数组，用于存储方块的坐标信息。

```
    private Tetrominoe pieceShape;
    private int[][] coordinates;
```

1.3 初始化 coordinates 数组并将方块形状设置为 NoShape

```
    public Shape() {

        coordinates = new int[4][2];
        setShape(Tetrominoe.NoShape);
    }
```

1.4 设置方块的形状。根据给定的形状，从预定义的坐标数组中复制相应的坐标值到 coordinates 数组中。

```
    void setShape(Tetrominoe shape) {

        int[][][] coordinateArray = 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}
```

```
            }
        };

        for (int i = 0; i < 4; i++) {
            System.arraycopy(coordinateArray[shape.ordinal()], 0, coordinates, 0, 4);
        }

        pieceShape = shape;
    }
```

1.5 返回方块的当前形状。

```
    Tetrominoe getShape() {
        return pieceShape;
    }
```

1.6 设置方块中指定索引的坐标的 X 或 Y 值

```
    public void setX(int shapeIndex, int xpos) {
        coordinates[shapeIndex][0] = xpos;
    }

    public void setY(int shapeIndex, int ypos) {
        coordinates[shapeIndex][1] = ypos;
    }
```

1.7 返回方块中指定索引的坐标的 X 或 Y 值。

```
    int getX(int shapeIndex) {
        return coordinates[shapeIndex][0];
    }

    int getY(int shapeIndex) {
        return coordinates[shapeIndex][1];
    }
```

1.8 随机选择一个形状，并将方块的形状设置为随机选择的形状。

```
  void setRandomShape() {
      Random randomObj = new Random();
      int x = (Math.abs(randomObj.nextInt()) % 7) + 1;

      Tetrominoe[] values = Tetrominoe.values();
```

```
        setShape(values[x]);
    }
```

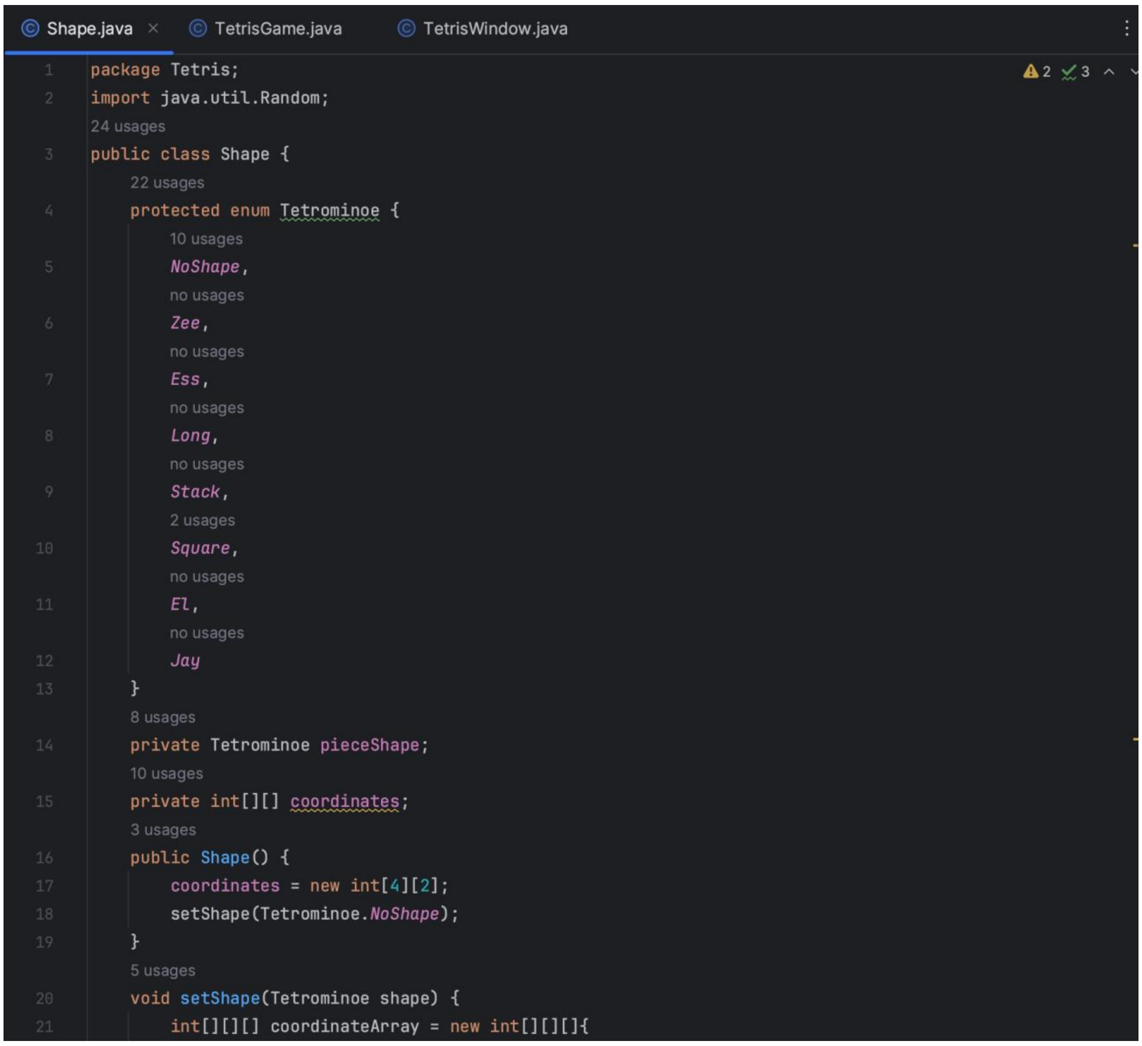
1.9minX() 方法返回方块中 X 坐标的最小值，minY() 方法返回方块中 Y 坐标的最小值。这些方法用于确定方块的边界。

```
    public int minX() {
        int m = coordinates[0][0];

        for (int i = 0; i < 4; i++) {
            m = Math.min(m, coordinates[i][0]);
        }

        return m;
    }


    int minY() {
        int m = coordinates[0][1];

        for (int i = 0; i < 4; i++) {
            m = Math.min(m, coordinates[i][1]);
        }

        return m;
    }
```

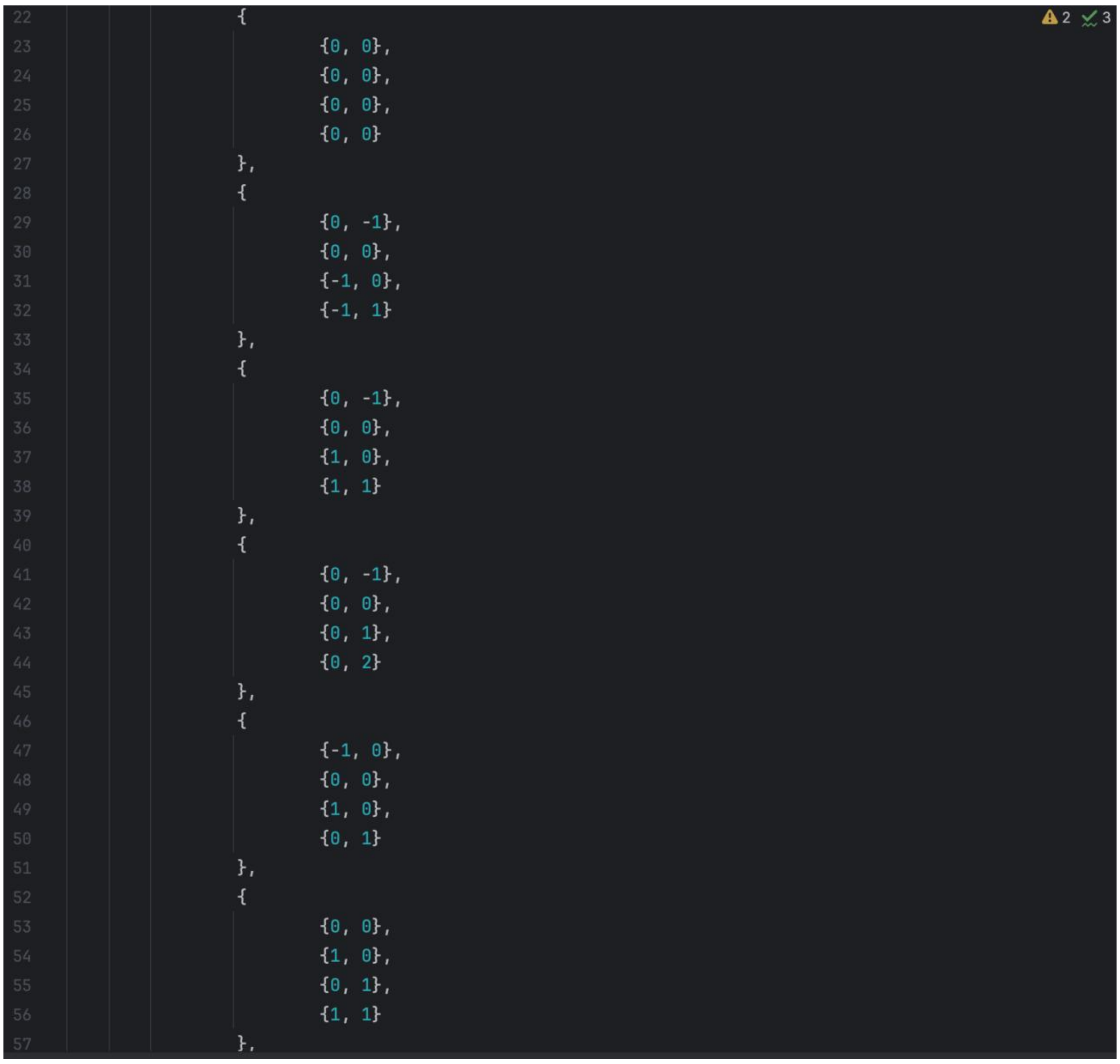
1.10rotateLeft()用于按逆时针或顺时针方向旋转方块。返回一个新的 Shape 对象，其中包含旋转后的方块。

```
      Shape rotateLeft() {
          if (pieceShape == Tetrominoe.Square) {
              return this;
          }

          Shape result = new Shape();
          result.pieceShape = pieceShape;

          for (int i = 0; i < 4; i++) {
              result.setX(i, getY(i));
              result.setY(i, -getX(i));
          }

          return result;
```

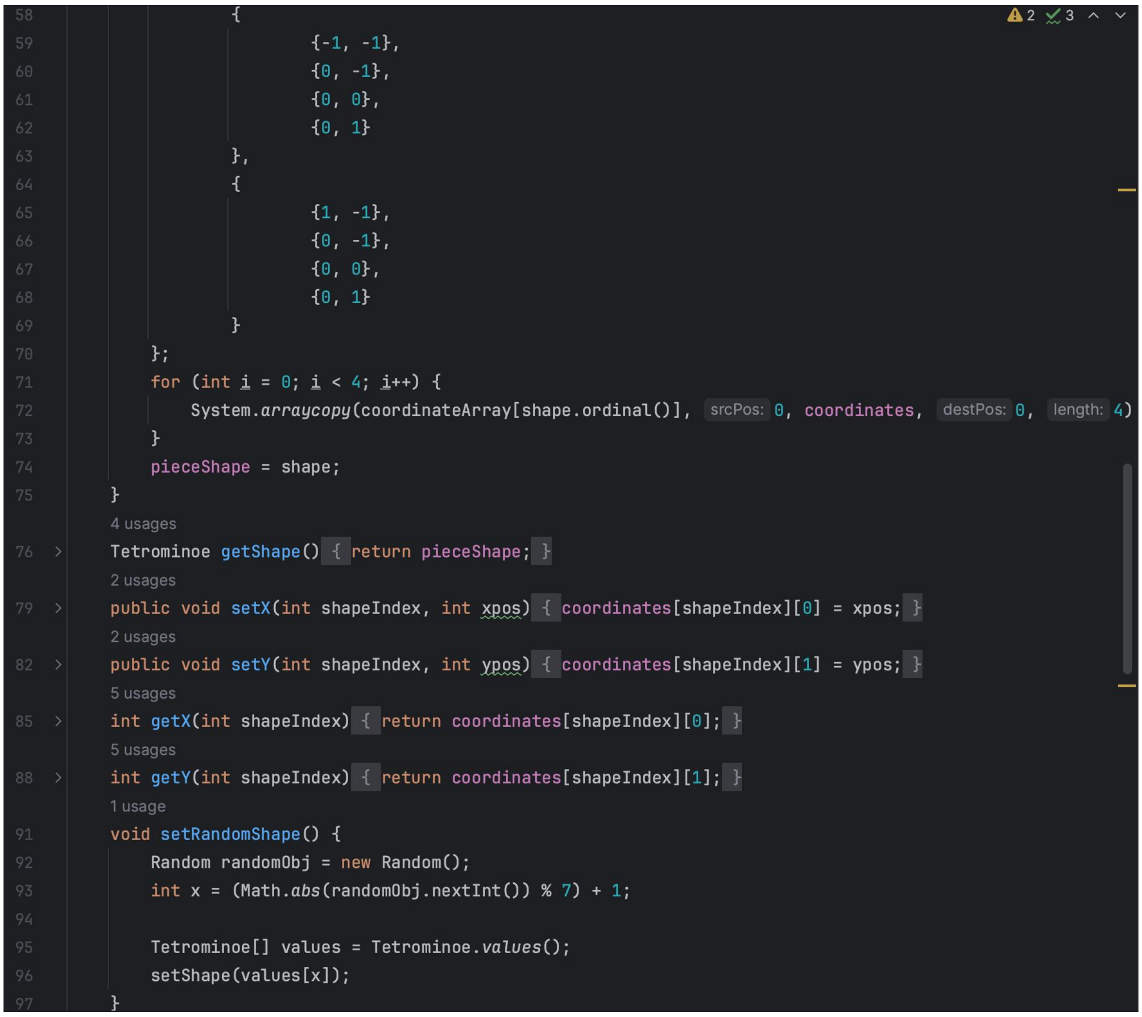
```
    }

    Shape rotateRight() {

        // A square doesn't need to be rotated, hence return
        if (pieceShape == Tetrominoe.Square) {
            return this;
        }

        Shape result = new Shape();
        result.pieceShape = pieceShape;

        for (int i = 0; i < 4; i++) {
            result.setX(i, -getY(i));
            result.setY(i, getX(i));
        }

        return result;
    }
}
```


Shape.java × TetrisGame.java TetrisWindow.java

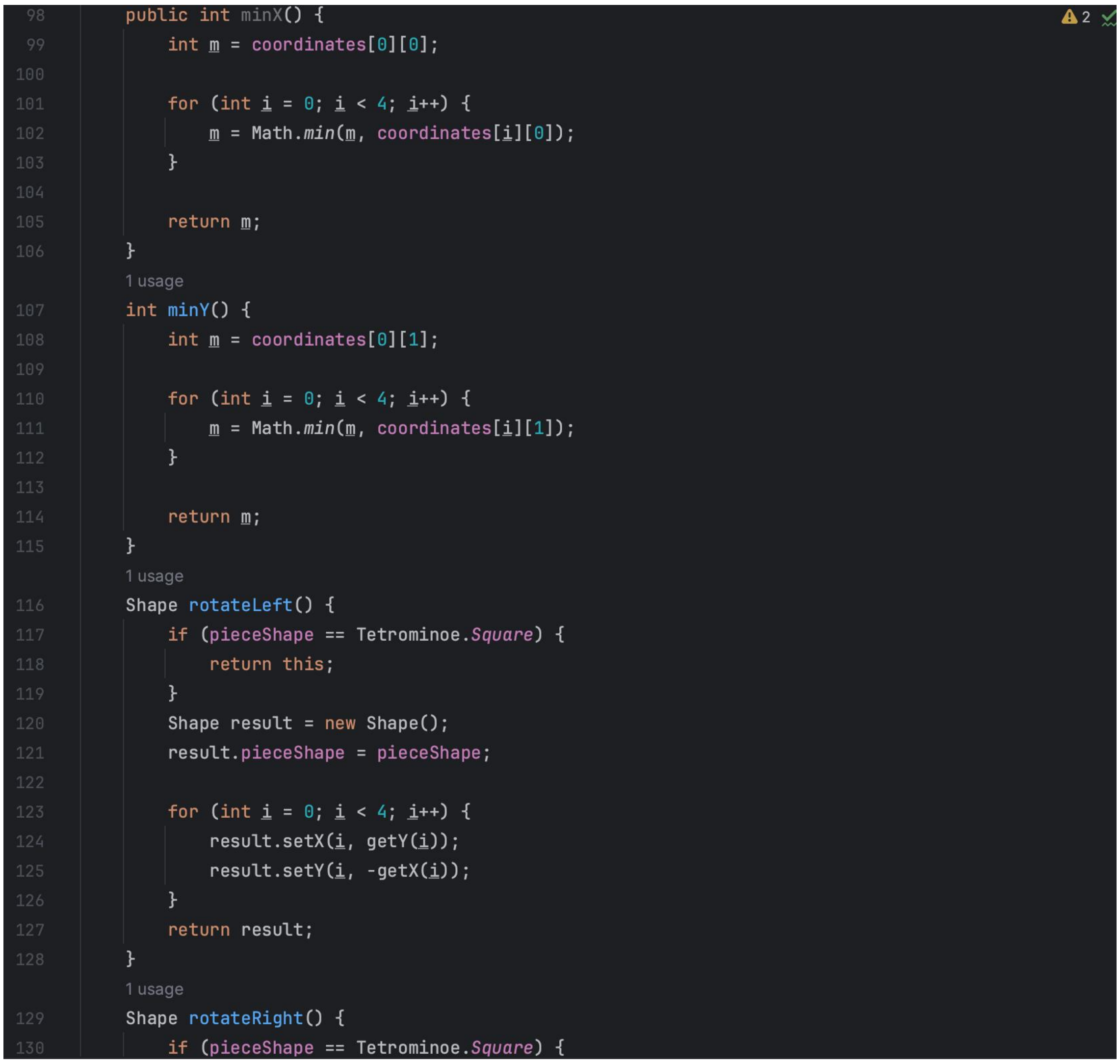
```
package Tetris;
import java.util.Random;
24 usages
public class Shape {
    22 usages
    protected enum Tetrominoe {
        10 usages
        NoShape,
        no usages
        Zee,
        no usages
        Ess,
        no usages
        Long,
        no usages
        Stack,
        2 usages
        Square,
        no usages
        El,
        no usages
        Jay
    }
    8 usages
    private Tetrominoe pieceShape;
    10 usages
    private int[][] coordinates;
    3 usages
    public Shape() {
        coordinates = new int[4][2];
        setShape(Tetrominoe.NoShape);
    }
    5 usages
    void setShape(Tetrominoe shape) {
        int[][][] coordinateArray = new int[][][]{
```



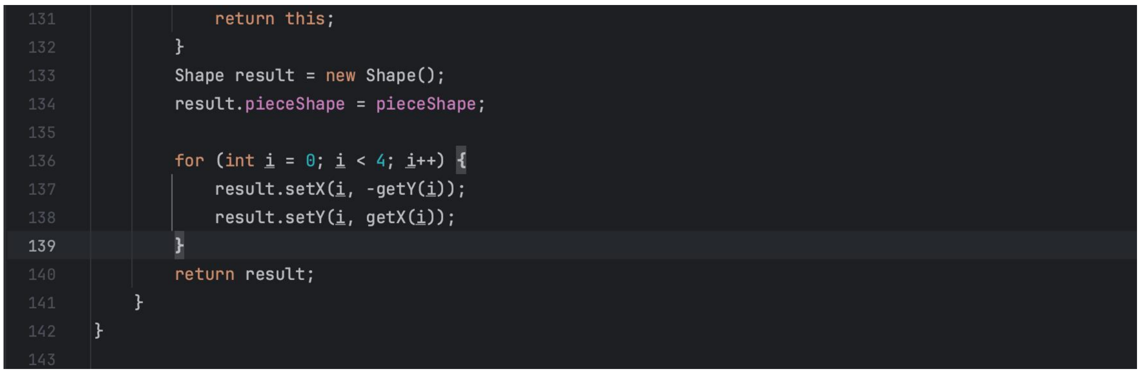
```
            {
                    {0, 0},
                    {0, 0},
                    {0, 0},
                    {0, 0}
            },
            {
                    {0, -1},
                    {0, 0},
                    {-1, 0},
                    {-1, 1}
            },
            {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



```java
                {
                        {-1, -1},
                        {0, -1},
                        {0, 0},
                        {0, 1}
                },
                {
                        {1, -1},
                        {0, -1},
                        {0, 0},
                        {0, 1}
                }
        };
        for (int i = 0; i < 4; i++) {
            System.arraycopy(coordinateArray[shape.ordinal()], srcPos: 0, coordinates, destPos: 0, length: 4)
        }
        pieceShape = shape;
    }
    4 usages
    Tetrominoe getShape() { return pieceShape; }
    2 usages
    public void setX(int shapeIndex, int xpos) { coordinates[shapeIndex][0] = xpos; }
    2 usages
    public void setY(int shapeIndex, int ypos) { coordinates[shapeIndex][1] = ypos; }
    5 usages
    int getX(int shapeIndex) { return coordinates[shapeIndex][0]; }
    5 usages
    int getY(int shapeIndex) { return coordinates[shapeIndex][1]; }
    1 usage
    void setRandomShape() {
        Random randomObj = new Random();
        int x = (Math.abs(randomObj.nextInt()) % 7) + 1;

        Tetrominoe[] values = Tetrominoe.values();
        setShape(values[x]);
    }
```



```java
    public int minX() {
        int m = coordinates[0][0];

        for (int i = 0; i < 4; i++) {
            m = Math.min(m, coordinates[i][0]);
        }

        return m;
    }
    1 usage
    int minY() {
        int m = coordinates[0][1];

        for (int i = 0; i < 4; i++) {
            m = Math.min(m, coordinates[i][1]);
        }

        return m;
    }
    1 usage
    Shape rotateLeft() {
        if (pieceShape == Tetrominoe.Square) {
            return this;
        }
        Shape result = new Shape();
        result.pieceShape = pieceShape;

        for (int i = 0; i < 4; i++) {
            result.setX(i, getY(i));
            result.setY(i, -getX(i));
        }
        return result;
    }
    1 usage
    Shape rotateRight() {
        if (pieceShape == Tetrominoe.Square) {
```



```java
            return this;
        }
        Shape result = new Shape();
        result.pieceShape = pieceShape;

        for (int i = 0; i < 4; i++) {
            result.setX(i, -getY(i));
            result.setY(i, getX(i));
        }
        return result;
    }
}
```

2.创建 TetrisGame 类：

```java
package Tetris;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
```

2.1 定义 TetrisGame 类，继承自 JPanel，表示游戏的主面板。
初始化游戏所需的变量和组件，包括游戏区域的宽度和高度、计时器、游戏状态等。

1.

```
public class TetrisGame extends JPanel {

    private final int WIDTH_OF_BOARD;
    private final int HEIGHT_OF_BOARD;

    private Timer timer;

    private boolean isFallingFinished = false;

    private boolean isPaused = false;

    private int numRemovedLines = 0;

    private int currentX = 0;
    private int currentY = 0;
    private int score = 0;

    private JLabel statusBar;
    private Shape currentShape;
    private Shape.Tetrominoe[] board;
```

2.2 提供辅助方法，如获取指定位置的方块形状、开始游戏、暂停游戏、绘制游戏界面、快速下落方块等。

1.

```
    public TetrisGame(TetrisWindow parent, int WIDTH_OF_BOARD, int
HEIGHT_OF_BOARD) {
        initBoard(parent);
        this.WIDTH_OF_BOARD = WIDTH_OF_BOARD;
        this.HEIGHT_OF_BOARD = HEIGHT_OF_BOARD;
    }

    private void initBoard(TetrisWindow parent) {
        setFocusable(true);
        statusBar = parent.getStatusBar();

        addKeyListener(new TetrisKeyAdapter());
    }

    Shape.Tetrominoe shapeAt(int x, int y) {
        return board[(y * WIDTH_OF_BOARD) + x];
    }

    void startGame() {
        currentShape = new Shape();
        board     =     new     Shape.Tetrominoe[WIDTH_OF_BOARD     *
HEIGHT_OF_BOARD];

        clearBoard();
        spawnBrick();

        timer = new Timer(300, new RepeatGameCycle());
        timer.start();
    }

    void pauseGame() {
        isPaused = !isPaused;

        if (isPaused) {
            statusBar.setText("Paused...");
        } else {
            score = numRemovedLines * 100;
            statusBar.setText(String.valueOf(score));
        }

        repaint();
    }
```

2.3 重写 paintComponent 方法，在面板上绘制游戏界面，包括已经落下的方块和当前正在下落的方块。

1.

```
        @Override
        public void paintComponent(Graphics g) {
            super.paintComponent(g);

            Dimension size = getSize();
            int   boardTop   =   (int)   size.getHeight()   -   HEIGHT_OF_BOARD   *
    squareHeight();

            for (int i = 0; i < HEIGHT_OF_BOARD; i++) {
                for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                    Shape.Tetrominoe shape = shapeAt(j, HEIGHT_OF_BOARD - i -
    1);
```

```
                if (shape != Shape.Tetrominoe.NoShape) {
                    transferColor(
                            g,
                            j * squareWidth(),
                            boardTop + i * squareHeight(),
                            shape);
                }
            }
        }

        if (getCurrentShape().getShape() != Shape.Tetrominoe.NoShape) {
            for (int i = 0; i < 4; i++) {
                int x = getCurrentX() + getCurrentShape().getX(i);
                int y = getCurrentY() - getCurrentShape().getY(i);

                transferColor(
                        g,
                        x * squareWidth(),
                        boardTop + (HEIGHT_OF_BOARD - y - 1) *
squareHeight(),
                        getCurrentShape().getShape());
            }
        }
    }

    void dropToBottom() {
        int newY = currentY;

        while (newY > 0) {
            if (!movePiece(currentShape, currentX, newY - 1)) {
                break;
            }

            newY--;
        }

        pieceDropped();
    }

    int squareWidth() {
        return (int) getSize().getWidth() / WIDTH_OF_BOARD;
    }

    int squareHeight() {
        return (int) getSize().getHeight() / HEIGHT_OF_BOARD;
    }

    void moveOneLineDown() {
        if (!movePiece(currentShape, currentX, currentY - 1)) {
            pieceDropped();
        }
    }

private void clearBoard() {
    for (int i = 0; i < HEIGHT_OF_BOARD * WIDTH_OF_BOARD; i++) {
        board[i] = Shape.Tetrominoe.NoShape;
    }
}

public Shape getCurrentShape() {
    return currentShape;
}

public int getCurrentX() {
    return currentX;
}

public int getCurrentY() {
    return currentY;
}

private void pieceDropped() {
    for (int i = 0; i < 4; i++) {
        int x = currentX + currentShape.getX(i);
        int y = currentY - currentShape.getY(i);
        board[(y * WIDTH_OF_BOARD) + x] = currentShape.getShape();
    }

    removeFullLines();

    if (!isFallingFinished) {
        spawnBrick();
    }
}

private void spawnBrick() {
    currentShape.setRandomShape();
    currentX = (WIDTH_OF_BOARD / 2) + 1;
```

```
            currentY = HEIGHT_OF_BOARD - 1 + currentShape.minY();

            if (!movePiece(currentShape, currentX, currentY)) {
                currentShape.setShape(Shape.Tetrominoe.NoShape);
                timer.stop();

                String msg = String.format("Game over! Score: %d", numRemovedLines *
100);
                statusBar.setText(msg);
            }
        }
```

2.4 实现移动方块、清除满行、生成新方块等游戏逻辑。

```
    public boolean movePiece(Shape newPiece, int newX, int newY) {
        for (int i = 0; i < 4; i++) {
            int x = newX + newPiece.getX(i);
            int y = newY - newPiece.getY(i);

            if (x < 0 || x >= WIDTH_OF_BOARD || y < 0 || y >= HEIGHT_OF_BOARD) {
                return false;
            }

            if (shapeAt(x, y) != Shape.Tetrominoe.NoShape) {
                return false;
            }
        }

        currentShape = newPiece;
        currentX = newX;
        currentY = newY;

        repaint();

        return true;
    }

    private void removeFullLines() {
        int numFullLines = 0;

        for (int i = HEIGHT_OF_BOARD - 1; i >= 0; i--) {
            boolean isLineFull = true;

            for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                if (shapeAt(j, i) == Shape.Tetrominoe.NoShape) {
                    isLineFull = false;
                    break;
                }
            }

            if (isLineFull) {
                numFullLines++;

                for (int k = i; k < HEIGHT_OF_BOARD - 1; k++) {
                    for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                        board[(k * WIDTH_OF_BOARD) + j] = shapeAt(j, k + 1);
                    }
                }
            }
        }

        if (numFullLines > 0) {
            numRemovedLines += numFullLines;

            statusBar.setText(String.valueOf(numRemovedLines * 100));
            isFallingFinished = true;
            currentShape.setShape(Shape.Tetrominoe.NoShape);
        }
    }
```

2.5 在指定位置绘制方块，并设置颜色。

```
    void transferColor(Graphics g, int x, int y, Shape.Tetrominoe shape) {

                Color[] colors = {
                new Color(0, 0, 0),
                new Color(239, 35, 35),
                new Color(17, 76, 217),
                new Color(112, 211, 238),
                new Color(23, 212, 40),
                new Color(243, 230, 87),
                new Color(33, 13, 132),
                new Color(197, 67, 234)
        };

        Color color = colors[shape.ordinal()];

        g.setColor(color);
        g.fillRect(
                x + 1,
                y + 1,
                squareWidth() - 2,
                squareHeight() - 2
        );

        g.setColor(color.brighter());
        g.drawLine(
                x,
                y + squareHeight() - 1,
                x,
                y);
        g.drawLine(
                x,
                y,
                x + squareWidth() - 1,
                y);

        g.setColor(color.darker());
        g.drawLine(
                x + 1,
                y + squareHeight() - 1,
                x + squareWidth() - 1,
                y + squareHeight() - 1);
        g.drawLine(
                x + squareWidth() - 1,
                y + squareHeight() - 1,
                x + squareWidth() - 1,
                y + 1);
    }
```

2.6 实现游戏循环的监听器，用于更新游戏状态和重绘面板。

```
    private class RepeatGameCycle implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            updateBoard();
            repaint();
        }
    }
```

2.7 更新游戏状态，包括方块自动下落和生成新方块。

```
    private void updateBoard() {
        if (isPaused) {
            return;
        }

        if (isFallingFinished) {
            isFallingFinished = false;
            spawnBrick();
        } else {
            moveOneLineDown();
        }
    }
```
2.8 开始新游戏。

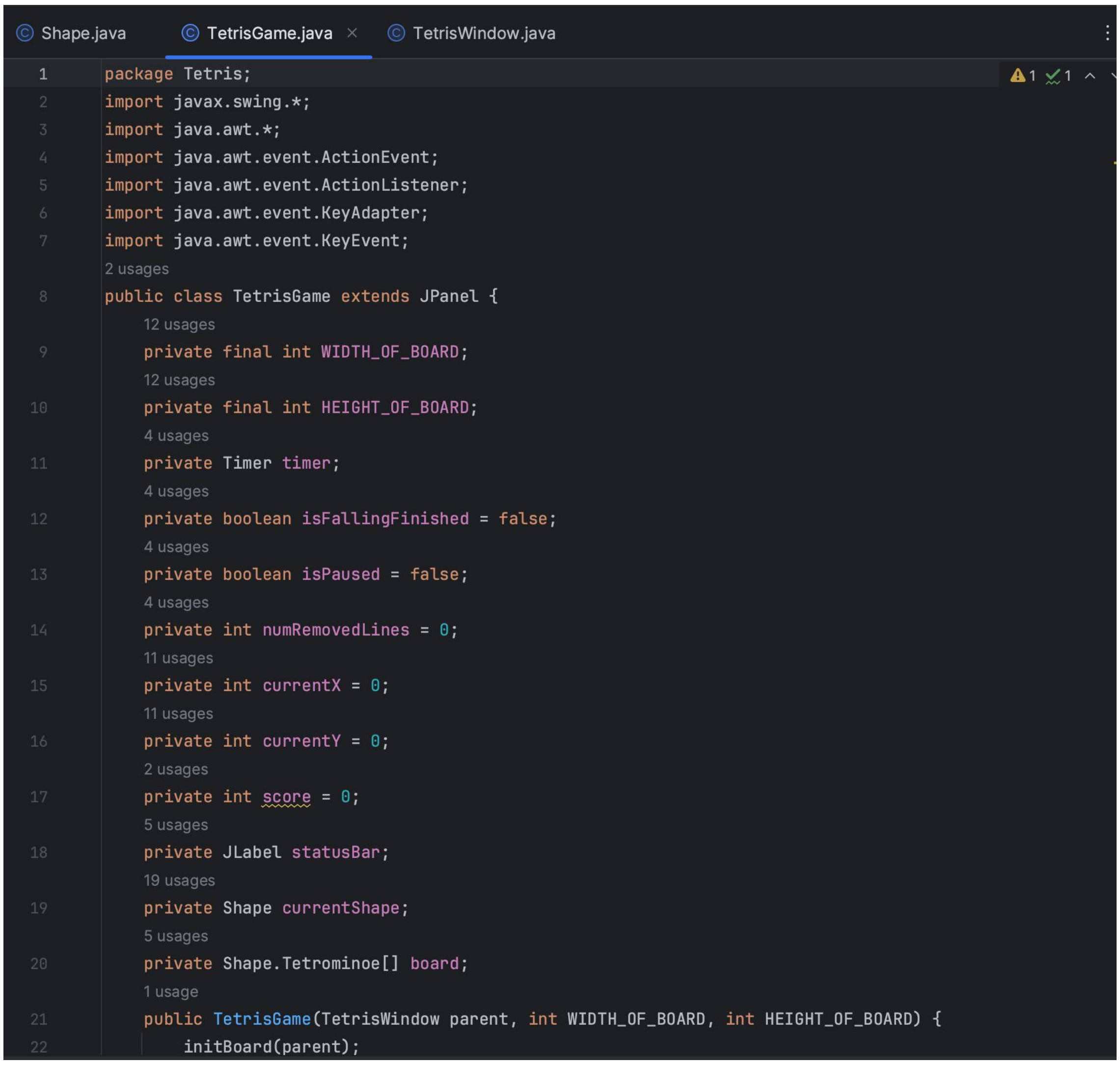
```
    void newGame() {
        currentShape.setShape(Shape.Tetrominoe.NoShape);
        timer.stop();

        startGame();
    }
```

2.9 将键盘事件转换为游戏操作。

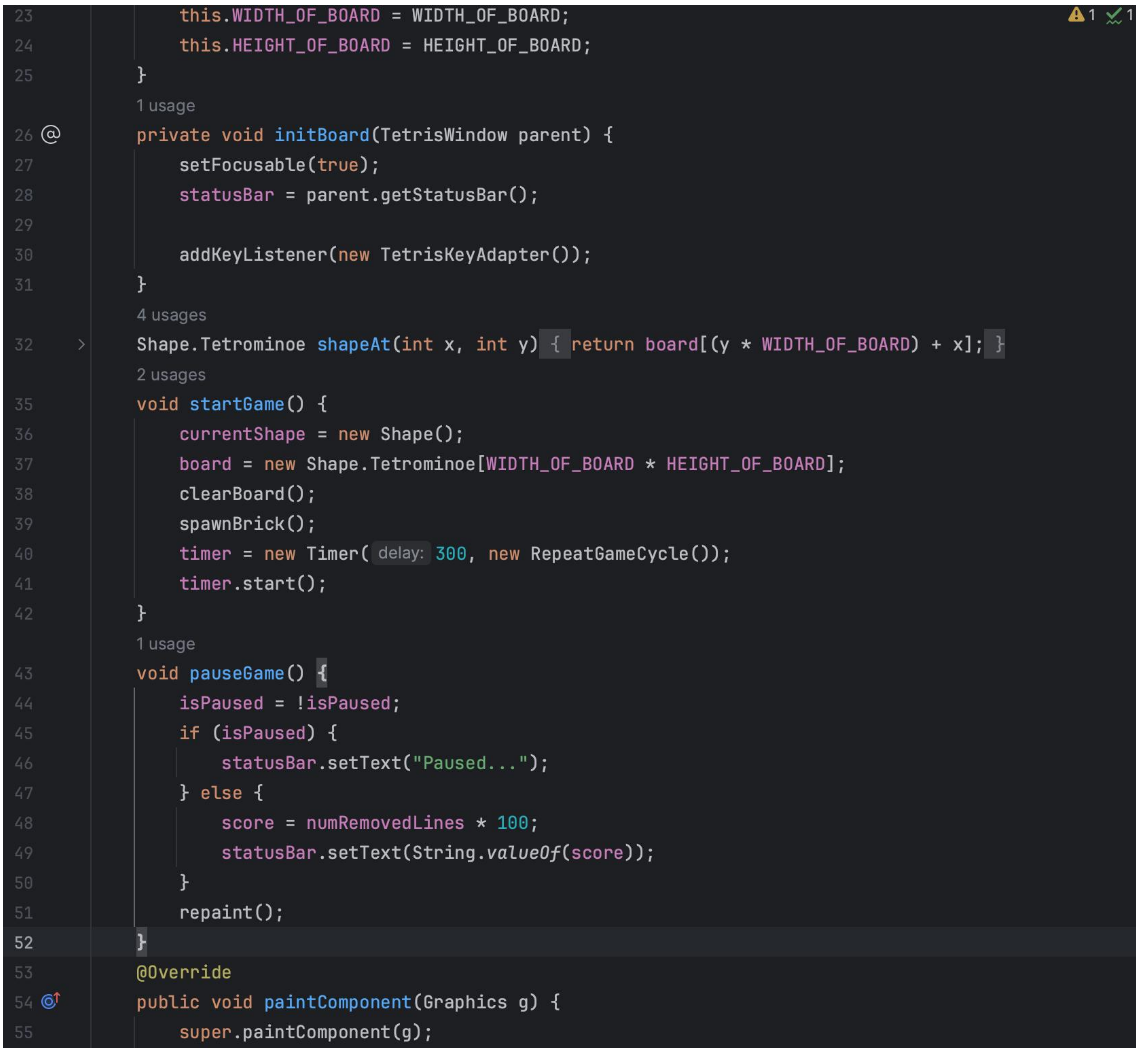
```
      private void translateKey(KeyEvent event) {
          if (currentShape.getShape() == Shape.Tetrominoe.NoShape) {
              return;
          }

          int eventKeyCode = event.getKeyCode();

          switch (eventKeyCode) {
              case KeyEvent.VK_SPACE -> pauseGame();
              case KeyEvent.VK_LEFT -> movePiece(currentShape, currentX - 1,
  currentY);
              case KeyEvent.VK_RIGHT -> movePiece(currentShape, currentX + 1,
  currentY);
              case KeyEvent.VK_DOWN -> movePiece(currentShape.rotateRight(),
  currentX, currentY);
              case KeyEvent.VK_UP -> movePiece(currentShape.rotateLeft(), currentX,
  currentY);
              case KeyEvent.VK_D -> dropToBottom();
              case KeyEvent.VK_F -> moveOneLineDown();
              case KeyEvent.VK_N -> newGame();
          }
      }
```

2.10 键盘适配器，用于监听键盘按键事件。

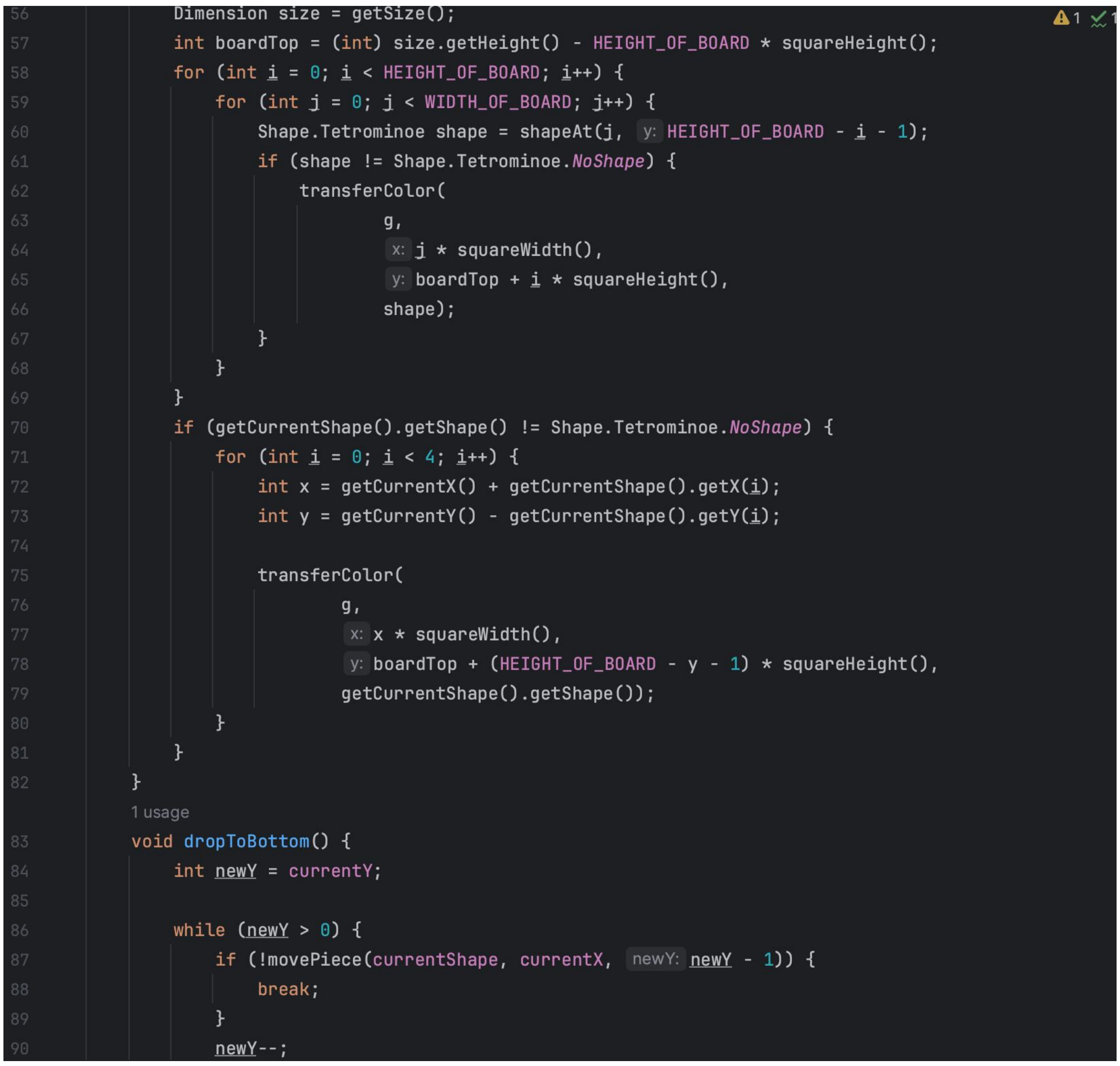
```
        class TetrisKeyAdapter extends KeyAdapter {

                @Override
                public void keyPressed(KeyEvent event) {
                        translateKey(event);
                }
        }
}
```



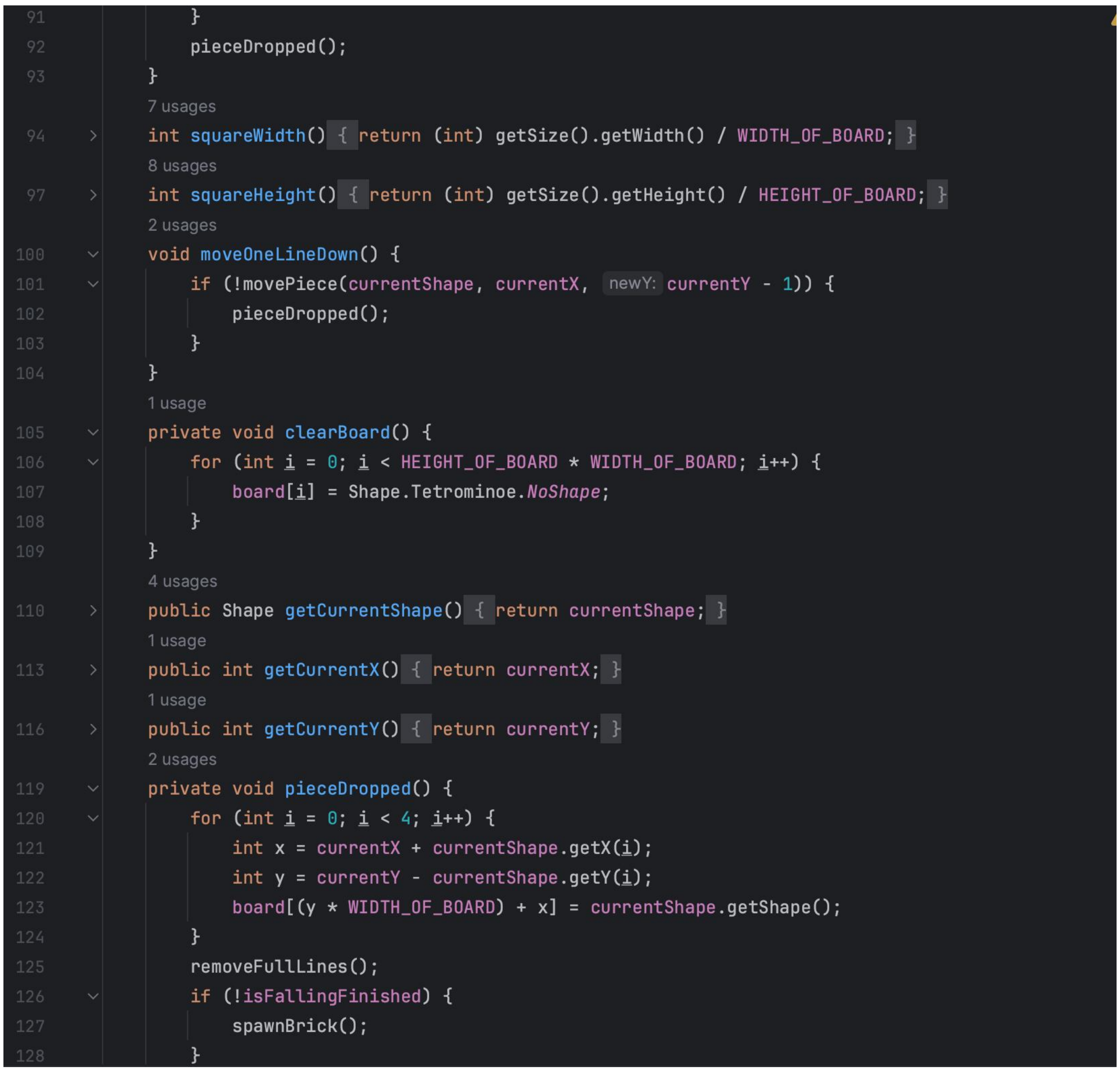
```
package Tetris;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
2 usages
public class TetrisGame extends JPanel {
    12 usages
    private final int WIDTH_OF_BOARD;
    12 usages
    private final int HEIGHT_OF_BOARD;
    4 usages
    private Timer timer;
    4 usages
    private boolean isFallingFinished = false;
    4 usages
    private boolean isPaused = false;
    4 usages
    private int numRemovedLines = 0;
    11 usages
    private int currentX = 0;
    11 usages
    private int currentY = 0;
    2 usages
    private int score = 0;
    5 usages
    private JLabel statusBar;
    19 usages
    private Shape currentShape;
    5 usages
    private Shape.Tetrominoe[] board;
    1 usage
    public TetrisGame(TetrisWindow parent, int WIDTH_OF_BOARD, int HEIGHT_OF_BOARD) {
        initBoard(parent);
```



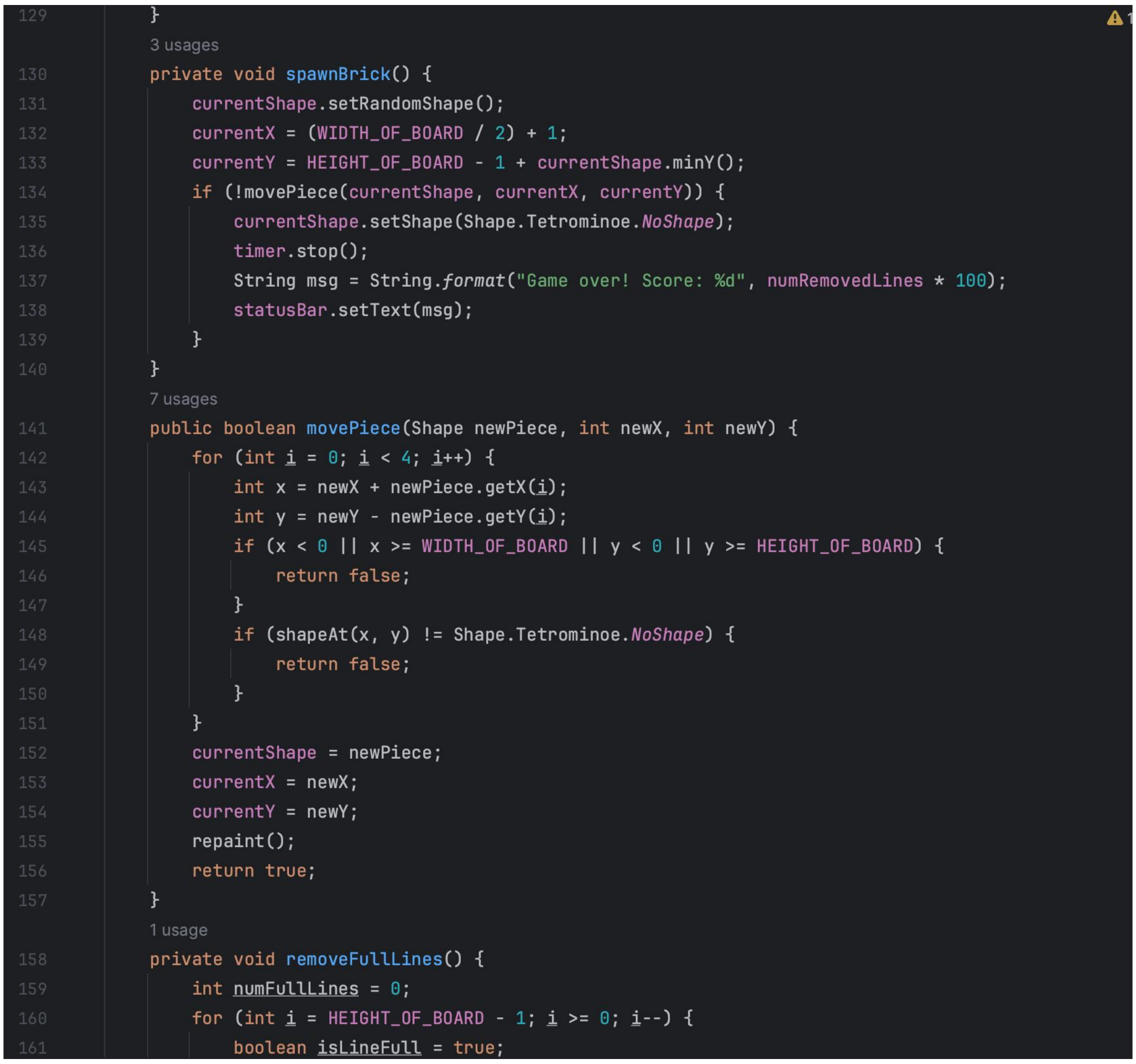
```
        this.WIDTH_OF_BOARD = WIDTH_OF_BOARD;
        this.HEIGHT_OF_BOARD = HEIGHT_OF_BOARD;
    }
    1 usage
    private void initBoard(TetrisWindow parent) {
        setFocusable(true);
        statusBar = parent.getStatusBar();

        addKeyListener(new TetrisKeyAdapter());
    }
    4 usages
    Shape.Tetrominoe shapeAt(int x, int y) { return board[(y * WIDTH_OF_BOARD) + x]; }
    2 usages
    void startGame() {
        currentShape = new Shape();
        board = new Shape.Tetrominoe[WIDTH_OF_BOARD * HEIGHT_OF_BOARD];
        clearBoard();
        spawnBrick();
        timer = new Timer( delay: 300, new RepeatGameCycle());
        timer.start();
    }
    1 usage
    void pauseGame() {
        isPaused = !isPaused;
        if (isPaused) {
            statusBar.setText("Paused...");
        } else {
            score = numRemovedLines * 100;
            statusBar.setText(String.valueOf(score));
        }
        repaint();
    }
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
```



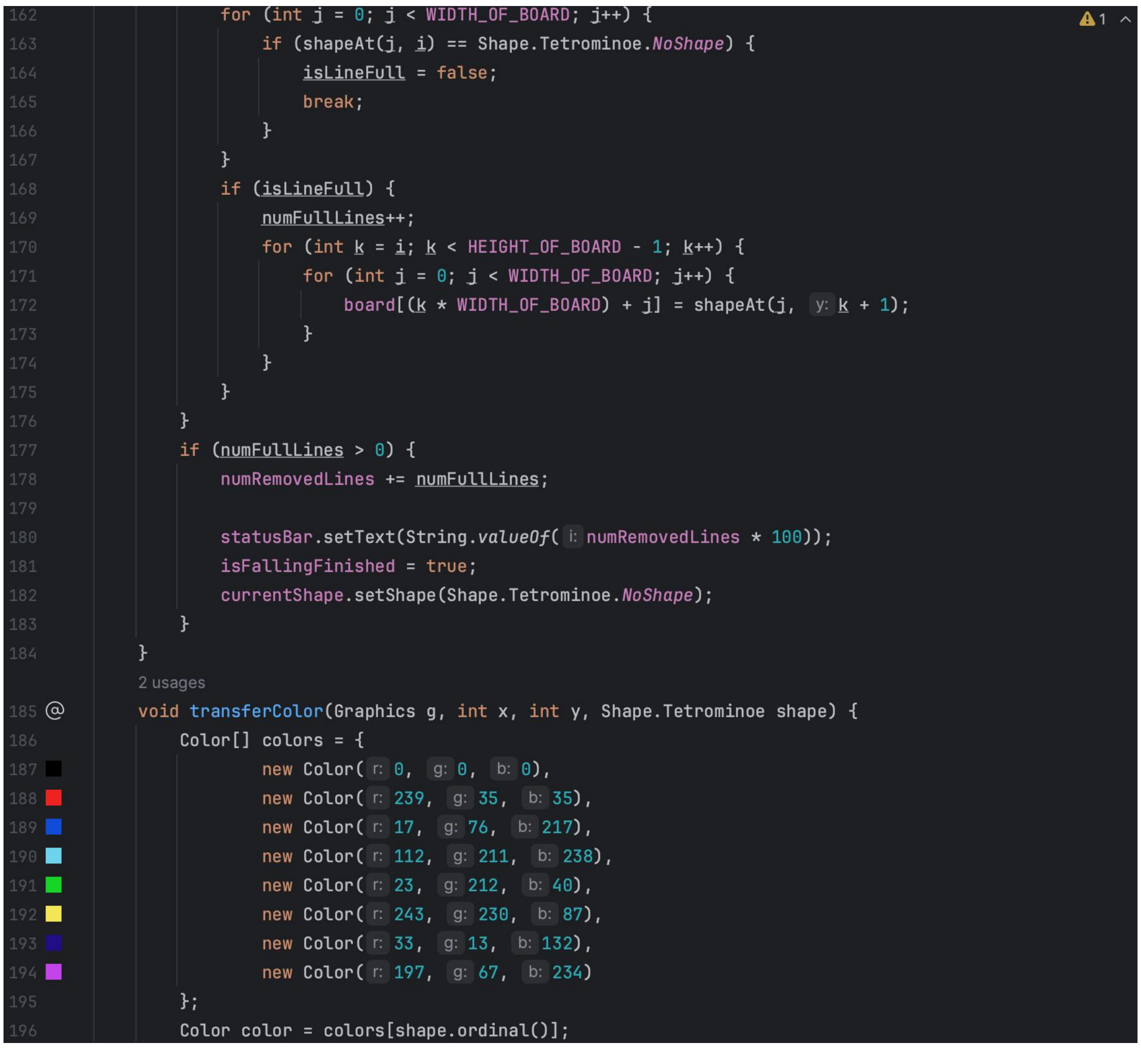
```
        Dimension size = getSize();
        int boardTop = (int) size.getHeight() - HEIGHT_OF_BOARD * squareHeight();
        for (int i = 0; i < HEIGHT_OF_BOARD; i++) {
            for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                Shape.Tetrominoe shape = shapeAt(j, y: HEIGHT_OF_BOARD - i - 1);
                if (shape != Shape.Tetrominoe.NoShape) {
                    transferColor(
                            g,
                            x: j * squareWidth(),
                            y: boardTop + i * squareHeight(),
                            shape);
                }
            }
        }
        if (getCurrentShape().getShape() != Shape.Tetrominoe.NoShape) {
            for (int i = 0; i < 4; i++) {
                int x = getCurrentX() + getCurrentShape().getX(i);
                int y = getCurrentY() - getCurrentShape().getY(i);

                transferColor(
                        g,
                        x: x * squareWidth(),
                        y: boardTop + (HEIGHT_OF_BOARD - y - 1) * squareHeight(),
                        getCurrentShape().getShape());
            }
        }
    }
    1 usage
    void dropToBottom() {
        int newY = currentY;

        while (newY > 0) {
            if (!movePiece(currentShape, currentX, newY: newY - 1)) {
                break;
            }
            newY--;
```



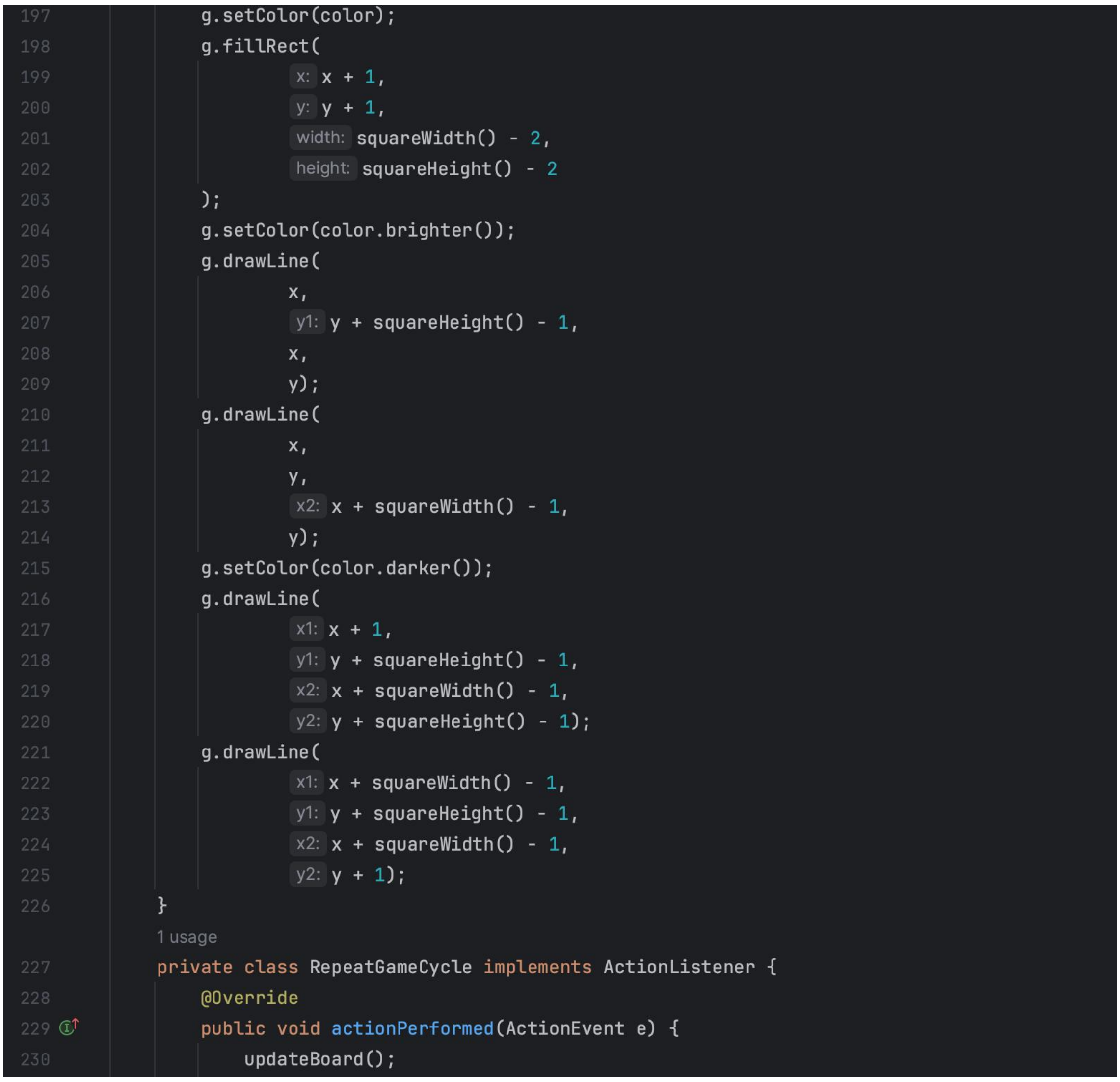
```
        }
        pieceDropped();
    }
    7 usages
    int squareWidth() { return (int) getSize().getWidth() / WIDTH_OF_BOARD; }
    8 usages
    int squareHeight() { return (int) getSize().getHeight() / HEIGHT_OF_BOARD; }
    2 usages
    void moveOneLineDown() {
        if (!movePiece(currentShape, currentX,  newY: currentY - 1)) {
            pieceDropped();
        }
    }
    1 usage
    private void clearBoard() {
        for (int i = 0; i < HEIGHT_OF_BOARD * WIDTH_OF_BOARD; i++) {
            board[i] = Shape.Tetrominoe.NoShape;
        }
    }
    4 usages
    public Shape getCurrentShape() { return currentShape; }
    1 usage
    public int getCurrentX() { return currentX; }
    1 usage
    public int getCurrentY() { return currentY; }
    2 usages
    private void pieceDropped() {
        for (int i = 0; i < 4; i++) {
            int x = currentX + currentShape.getX(i);
            int y = currentY - currentShape.getY(i);
            board[(y * WIDTH_OF_BOARD) + x] = currentShape.getShape();
        }
        removeFullLines();
        if (!isFallingFinished) {
            spawnBrick();
        }
```



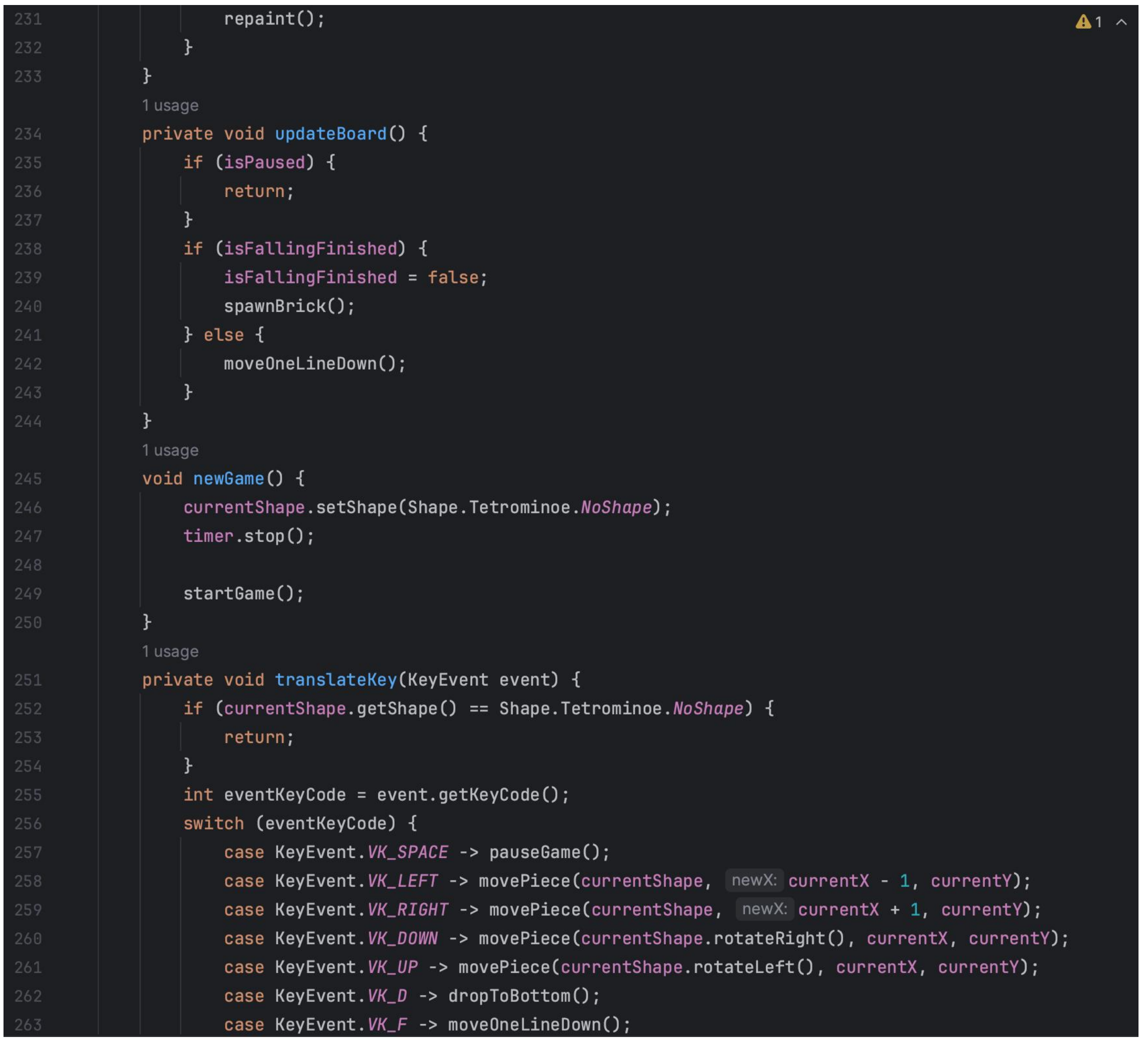
```
    }
    3 usages
    private void spawnBrick() {
        currentShape.setRandomShape();
        currentX = (WIDTH_OF_BOARD / 2) + 1;
        currentY = HEIGHT_OF_BOARD - 1 + currentShape.minY();
        if (!movePiece(currentShape, currentX, currentY)) {
            currentShape.setShape(Shape.Tetrominoe.NoShape);
            timer.stop();
            String msg = String.format("Game over! Score: %d", numRemovedLines * 100);
            statusBar.setText(msg);
        }
    }
    7 usages
    public boolean movePiece(Shape newPiece, int newX, int newY) {
        for (int i = 0; i < 4; i++) {
            int x = newX + newPiece.getX(i);
            int y = newY - newPiece.getY(i);
            if (x < 0 || x >= WIDTH_OF_BOARD || y < 0 || y >= HEIGHT_OF_BOARD) {
                return false;
            }
            if (shapeAt(x, y) != Shape.Tetrominoe.NoShape) {
                return false;
            }
        }
        currentShape = newPiece;
        currentX = newX;
        currentY = newY;
        repaint();
        return true;
    }
    1 usage
    private void removeFullLines() {
        int numFullLines = 0;
        for (int i = HEIGHT_OF_BOARD - 1; i >= 0; i--) {
            boolean isLineFull = true;
```



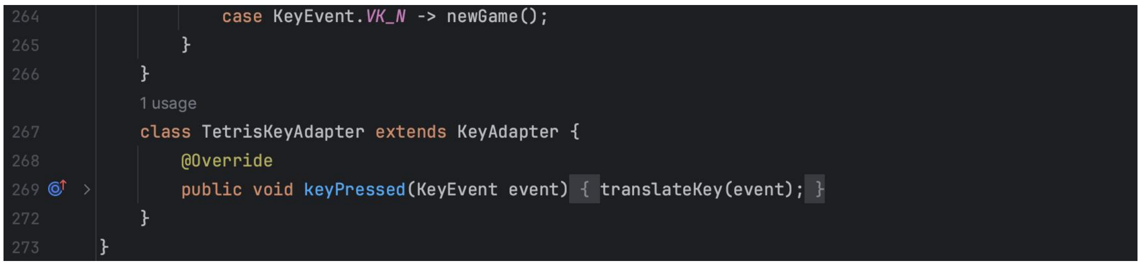
```
            for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                if (shapeAt(j, i) == Shape.Tetrominoe.NoShape) {
                    isLineFull = false;
                    break;
                }
            }
            if (isLineFull) {
                numFullLines++;
                for (int k = i; k < HEIGHT_OF_BOARD - 1; k++) {
                    for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                        board[(k * WIDTH_OF_BOARD) + j] = shapeAt(j, y: k + 1);
                    }
                }
            }
        }
        if (numFullLines > 0) {
            numRemovedLines += numFullLines;

            statusBar.setText(String.valueOf( i: numRemovedLines * 100));
            isFallingFinished = true;
            currentShape.setShape(Shape.Tetrominoe.NoShape);
        }
    }
    2 usages
    void transferColor(Graphics g, int x, int y, Shape.Tetrominoe shape) {
        Color[] colors = {
                new Color( r: 0,  g: 0,  b: 0),
                new Color( r: 239,  g: 35,  b: 35),
                new Color( r: 17,  g: 76,  b: 217),
                new Color( r: 112,  g: 211,  b: 238),
                new Color( r: 23,  g: 212,  b: 40),
                new Color( r: 243,  g: 230,  b: 87),
                new Color( r: 33,  g: 13,  b: 132),
                new Color( r: 197,  g: 67,  b: 234)
        };
        Color color = colors[shape.ordinal()];
```



```java
        g.setColor(color);
        g.fillRect(
                x: x + 1,
                y: y + 1,
                width: squareWidth() - 2,
                height: squareHeight() - 2
        );
        g.setColor(color.brighter());
        g.drawLine(
                x,
                y1: y + squareHeight() - 1,
                x,
                y);
        g.drawLine(
                x,
                y,
                x2: x + squareWidth() - 1,
                y);
        g.setColor(color.darker());
        g.drawLine(
                x1: x + 1,
                y1: y + squareHeight() - 1,
                x2: x + squareWidth() - 1,
                y2: y + squareHeight() - 1);
        g.drawLine(
                x1: x + squareWidth() - 1,
                y1: y + squareHeight() - 1,
                x2: x + squareWidth() - 1,
                y2: y + 1);
    }
    1 usage
    private class RepeatGameCycle implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            updateBoard();
```



```java
            repaint();
        }
    }
    1 usage
    private void updateBoard() {
        if (isPaused) {
            return;
        }
        if (isFallingFinished) {
            isFallingFinished = false;
            spawnBrick();
        } else {
            moveOneLineDown();
        }
    }
    1 usage
    void newGame() {
        currentShape.setShape(Shape.Tetrominoe.NoShape);
        timer.stop();

        startGame();
    }
    1 usage
    private void translateKey(KeyEvent event) {
        if (currentShape.getShape() == Shape.Tetrominoe.NoShape) {
            return;
        }
        int eventKeyCode = event.getKeyCode();
        switch (eventKeyCode) {
            case KeyEvent.VK_SPACE -> pauseGame();
            case KeyEvent.VK_LEFT -> movePiece(currentShape,  newX: currentX - 1, currentY);
            case KeyEvent.VK_RIGHT -> movePiece(currentShape,  newX: currentX + 1, currentY);
            case KeyEvent.VK_DOWN -> movePiece(currentShape.rotateRight(), currentX, currentY);
            case KeyEvent.VK_UP -> movePiece(currentShape.rotateLeft(), currentX, currentY);
            case KeyEvent.VK_D -> dropToBottom();
            case KeyEvent.VK_F -> moveOneLineDown();
```



```java
            case KeyEvent.VK_N -> newGame();
        }
    }
    1 usage
    class TetrisKeyAdapter extends KeyAdapter {
        @Override
        public void keyPressed(KeyEvent event) { translateKey(event); }
    }
}
```

3.创建 TetrisWindow 类：

package Tetris;

import javax.swing.*;
import java.awt.*;
import java.util.Scanner;

3.1 声明一个标签用于显示游戏状态栏。

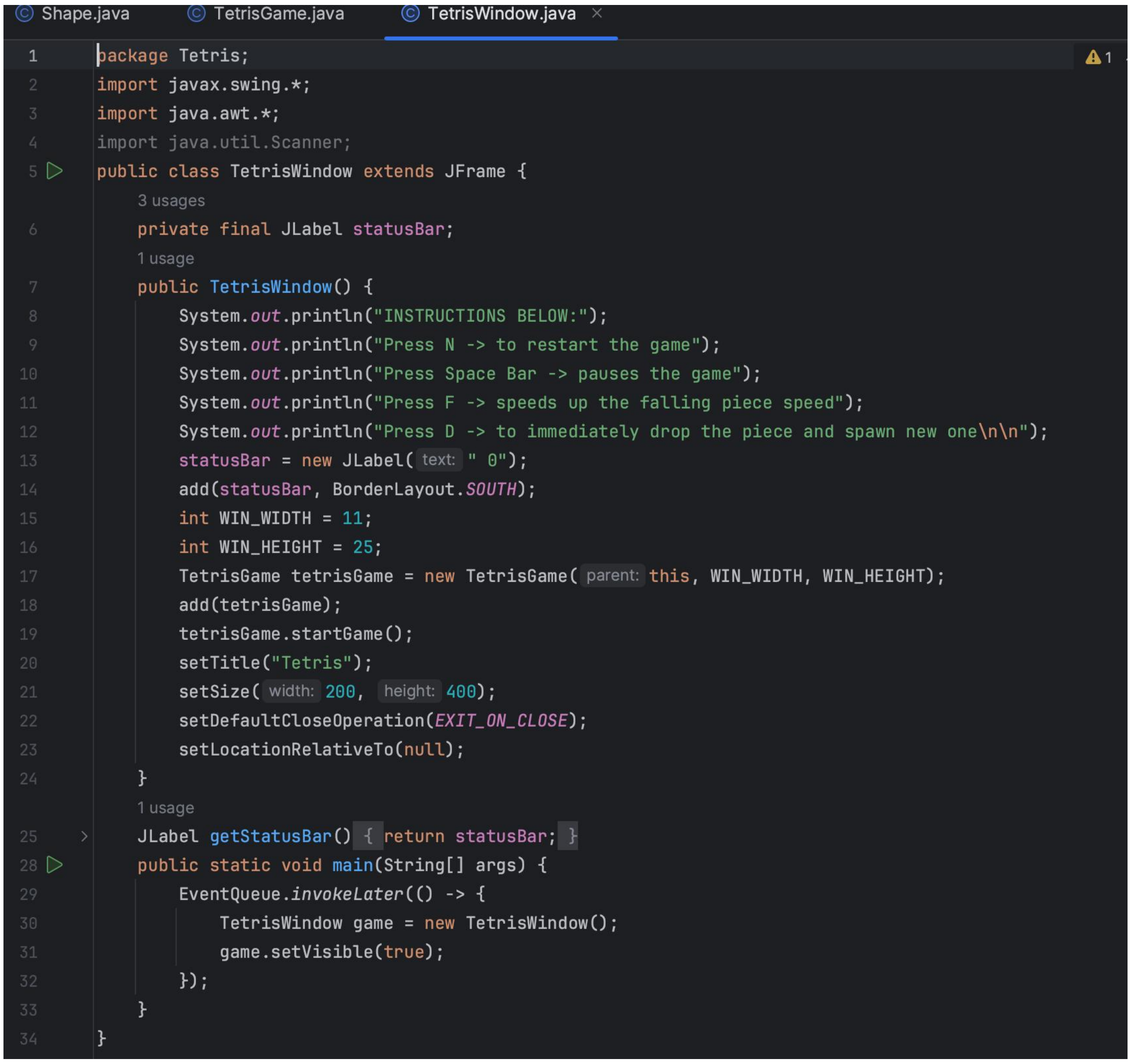
public class TetrisWindow extends JFrame {

```
    private final JLabel statusBar;
```
3.2 初始化游戏窗口，设置游戏说明，并创建 TetrisGame 的实例。

1.
```
    public TetrisWindow() {
        System.out.println("INSTRUCTIONS BELOW:");
        System.out.println("Press N -> to restart the game");
        System.out.println("Press Space Bar -> pauses the game");
        System.out.println("Press F -> speeds up the falling piece speed");
        System.out.println("Press D -> to immediately drop the piece and
spawn new one\n\n");

        statusBar = new JLabel(" 0");
        add(statusBar, BorderLayout.SOUTH);

        int WIN_WIDTH = 11;
        int WIN_HEIGHT = 25;

        TetrisGame tetrisGame = new TetrisGame(this, WIN_WIDTH,
WIN_HEIGHT);
        add(tetrisGame);
        tetrisGame.startGame();

        setTitle("Tetris");
        setSize(200, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }
```

3.3 创建 statusBar 标签并添加到窗口的底部。

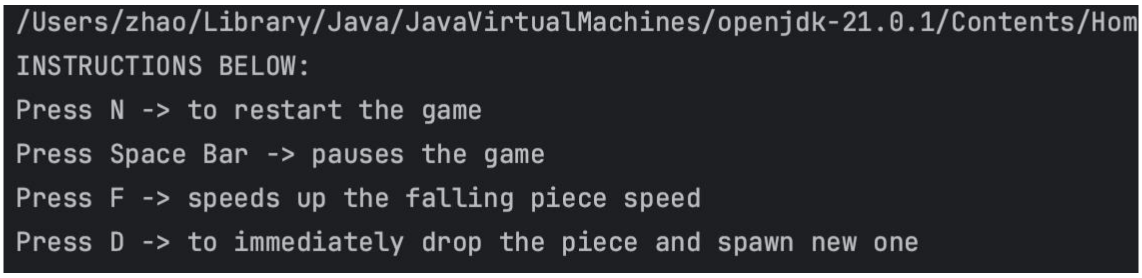
```
    JLabel getStatusBar() {
        return statusBar;
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            TetrisWindow game = new TetrisWindow();
            game.setVisible(true);
        });
    }
}
```



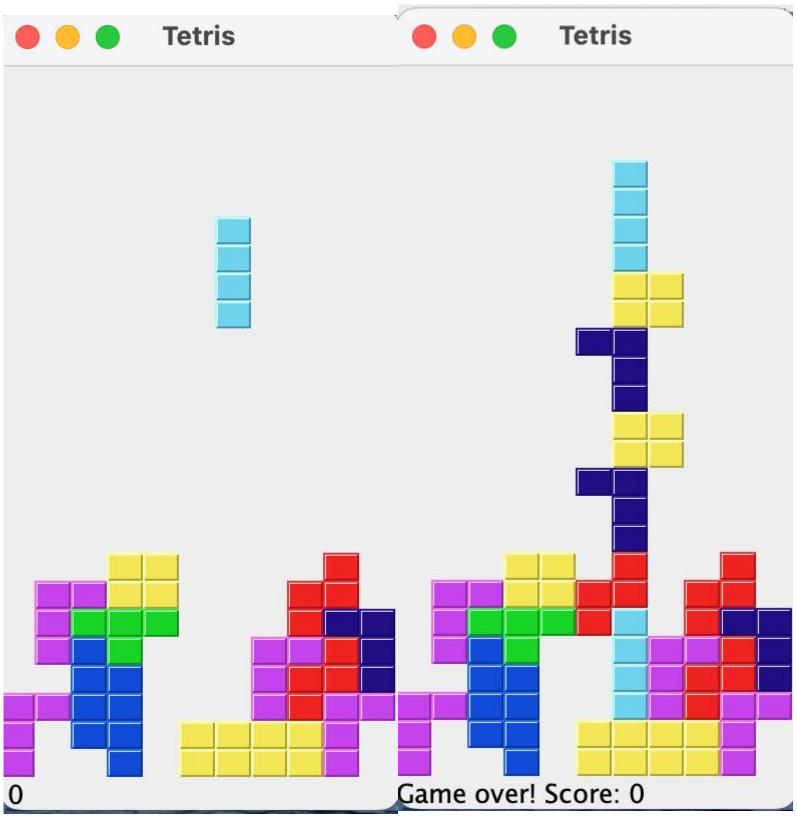
```java
package Tetris;
import javax.swing.*;
import java.awt.*;
import java.util.Scanner;
public class TetrisWindow extends JFrame {
    3 usages
    private final JLabel statusBar;
    1 usage
    public TetrisWindow() {
        System.out.println("INSTRUCTIONS BELOW:");
        System.out.println("Press N -> to restart the game");
        System.out.println("Press Space Bar -> pauses the game");
        System.out.println("Press F -> speeds up the falling piece speed");
        System.out.println("Press D -> to immediately drop the piece and spawn new one\n\n");
        statusBar = new JLabel( text: " 0");
        add(statusBar, BorderLayout.SOUTH);
        int WIN_WIDTH = 11;
        int WIN_HEIGHT = 25;
        TetrisGame tetrisGame = new TetrisGame( parent: this, WIN_WIDTH, WIN_HEIGHT);
        add(tetrisGame);
        tetrisGame.startGame();
        setTitle("Tetris");
        setSize( width: 200,  height: 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }
    1 usage
    JLabel getStatusBar() { return statusBar; }
    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            TetrisWindow game = new TetrisWindow();
            game.setVisible(true);
        });
    }
}
```

五、实验结果分析及实验心得

1. 实验结果



```
/Users/zhao/Library/Java/JavaVirtualMachines/openjdk-21.0.1/Contents/Hom
INSTRUCTIONS BELOW:
Press N -> to restart the game
Press Space Bar -> pauses the game
Press F -> speeds up the falling piece speed
Press D -> to immediately drop the piece and spawn new one
```



2. 结果分析

游戏窗口的标题是 "Tetris"，大小为 200x400 像素，并且具有关闭操作。游戏窗口中包含一个游戏面板，用于显示俄罗斯方块的游戏画面和玩家操作。在游戏窗口的底部有一个状态栏，用于显示游戏的状态和得分。游戏支持重新开始游戏、暂停游戏、加速下落和快速下落等操作。通过键盘按键 N 可以重新开始游戏，空格键可以暂停游戏，按键 F 可以加速方块下落速度，按键 D 可以立即将方块下落并生成新的方块。

3.实验心得

通过编写代码，我对俄罗斯方块游戏的实现有了更深入的了解。了解了如何将一个复杂的问题分解为更小的模块，代码中使用了面向对象的设计思想，将游戏的各个组件和功能模块进行了封装和分离，使代码具有可维护性和可扩展性。

我也学会了如何处理图形界面的交互和更新，并根据用户的操作来改变游戏的状态。代码中使用了 Java 的图形用户界面（GUI）库 Swing 来创建游戏窗口和绘制游戏画面。通过使用 Swing，可以实现跨平台的图形界面，并方便地与用户进行交互。

实验还提醒我在编程过程中要注重良好的代码风格和结构。我注意到代码的可读性和可维护性对于开发和调试过程至关重要。通过使用合适的命名、注释和代码组织，我能够更轻松地理解和修改代码。

总的来说，这个实验是一次很有意义的编程练习。通过编写俄罗斯方块游戏代码，我不仅加深了对面向对象编程和图形用户界面的理解，还提高了我的问题解决能力和编码技巧。我相信这些经验将对我今后的编程工作和学习有所帮助。