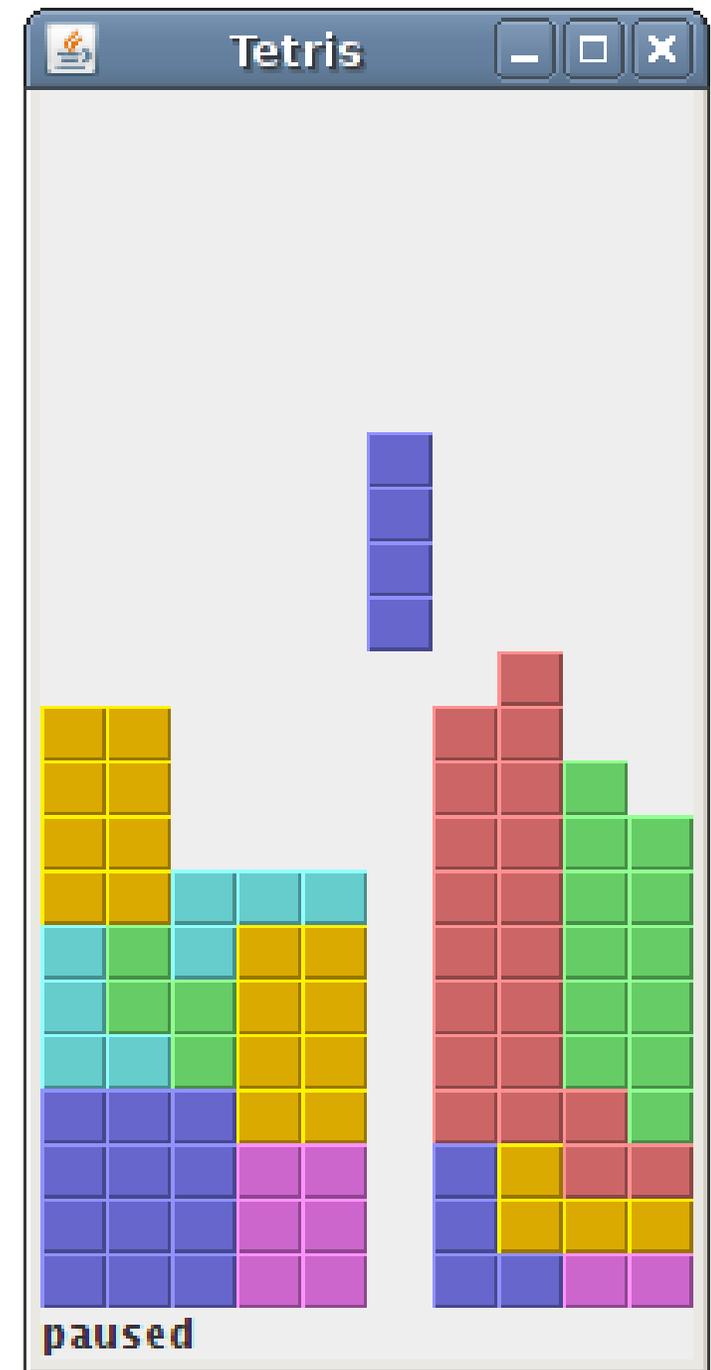


OOP Homework 3 - Tetris

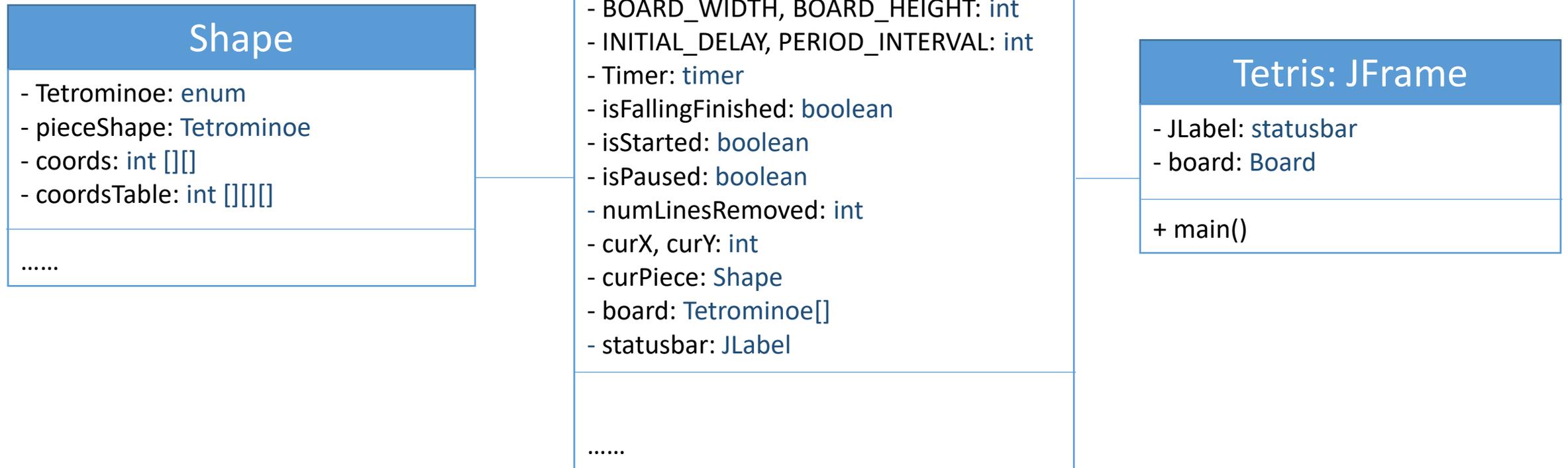
Kuan-Ting Lai
2021/6/16

Developing a Tetris Game

- zetcode.com/tutorials/javagamestutorial/tetris/
- Using Java Swing
- ↑ ↓: Rotate
- ← →: Move left/right
- Space: drop immediately
- d: drop faster



Class Diagram

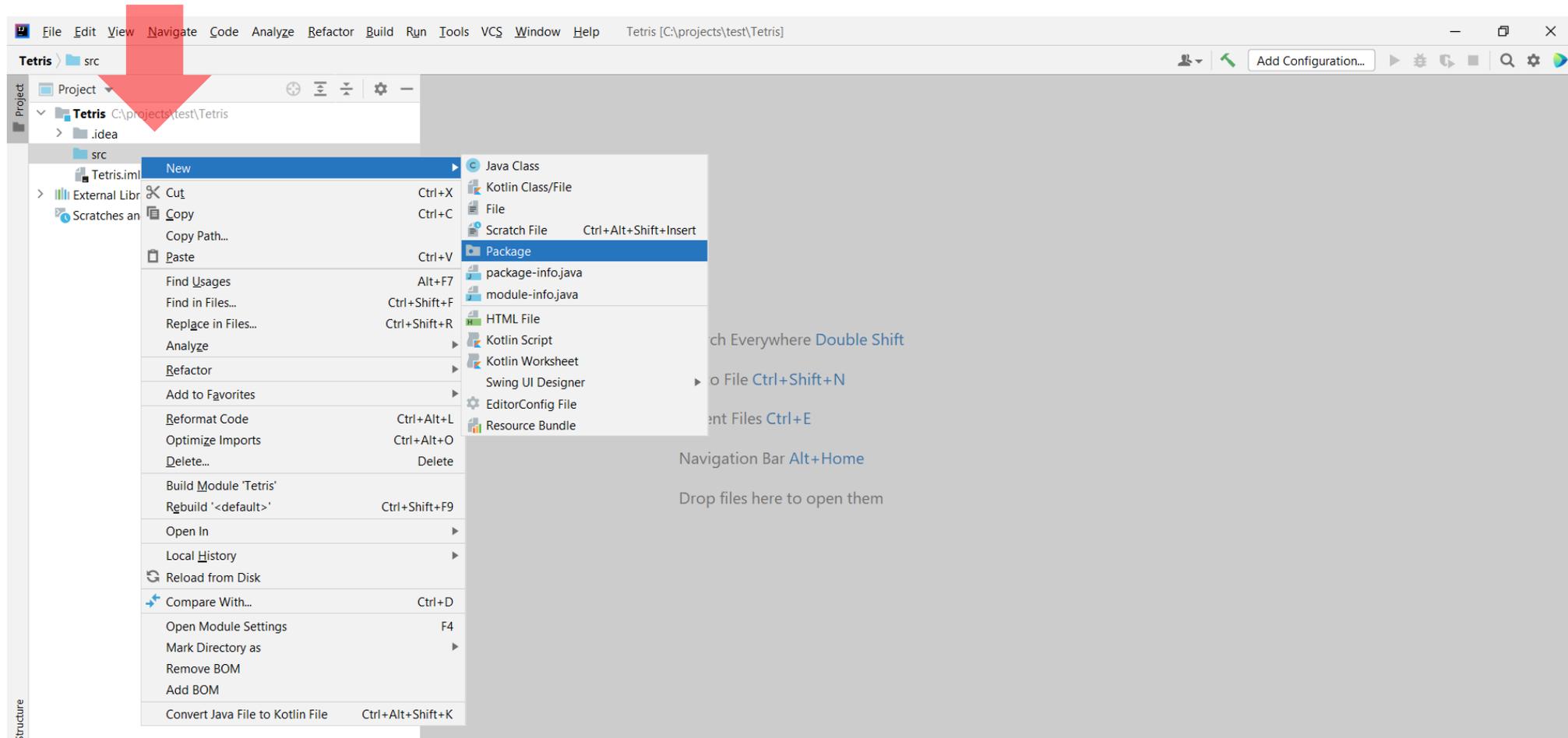


Building Tetris using IntelliJ

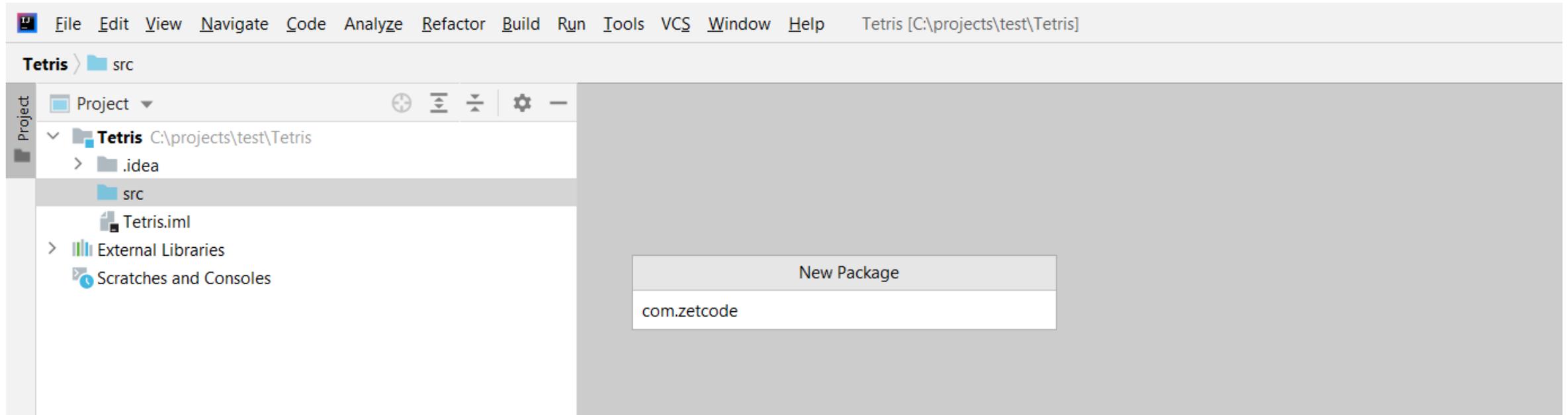
- Create a project “Tetris”
- Add a package “com.zetcode”
- Add three files in the package:
 - “Shape.java”, “Board.java”, “Tetris.java”

Create New Project & New Package

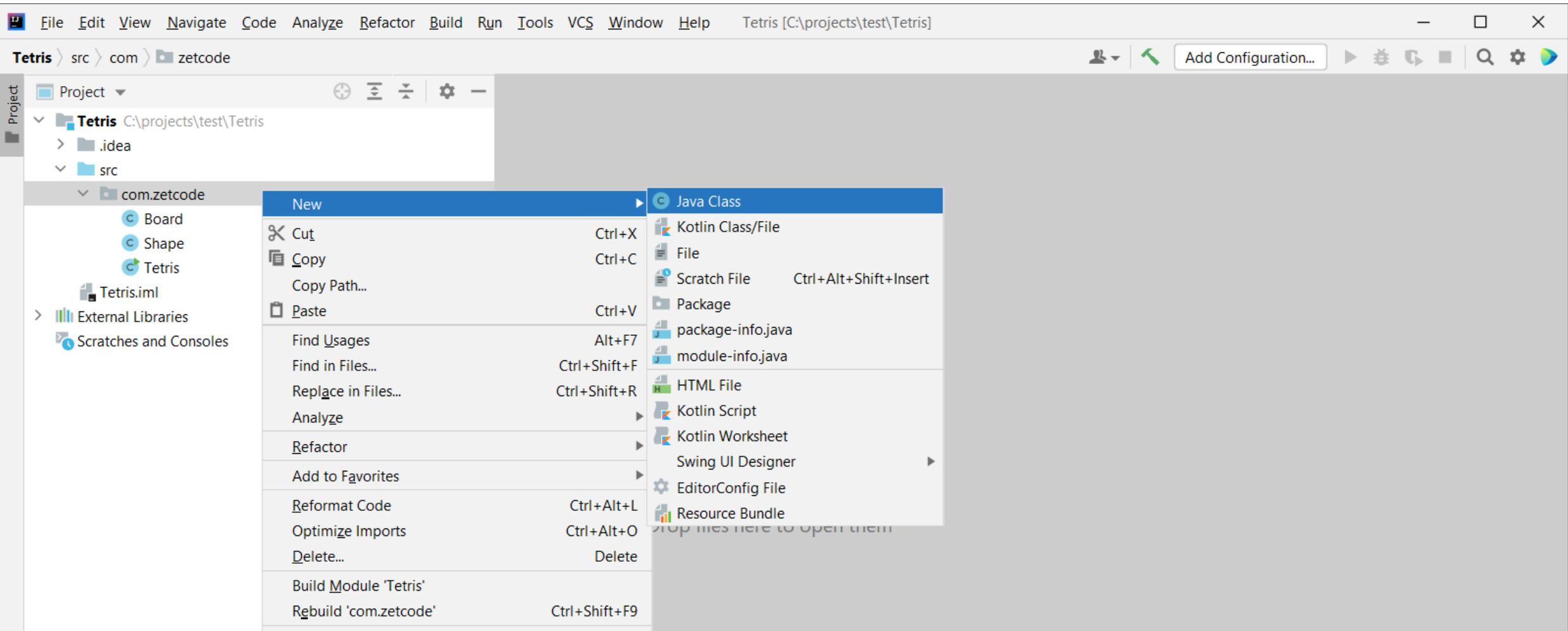
1. File -> New Project -> Next -> Next -> Project name: Tetris
2. Right click on src -> New -> Package



New Package name: com.zetcode



Add 3 files: Board.java, Shape.java & Tetris.java



Shape.java

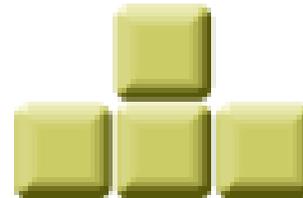
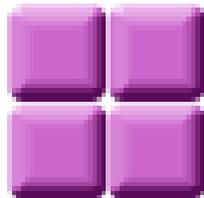
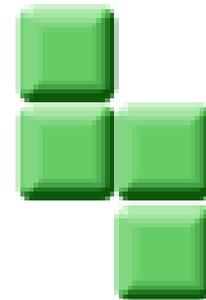
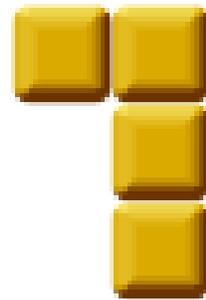
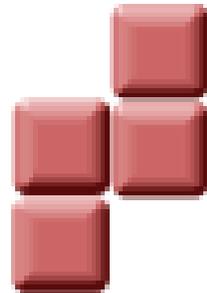
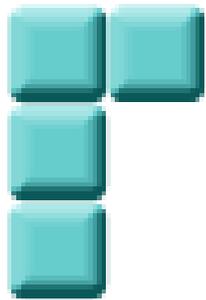
Shape

- Tetrominoe: enum
- pieceShape: Tetrominoe
- coords: int [][]
- coordsTable: int [][][]

- + Shape()
- # setShape(shape: Tetrominoe)
- + getShape() : Tetrominoe
- + setRandomShape()
- + x(index: int) : int
- + y(index: int) : int
- + minX() : int
- + minY() : int
- + rotateLeft() : Shape
- + rotateRight() : Shape
- initShape()
- setX(index: int, x: int)
- setY(index: int, y: int)

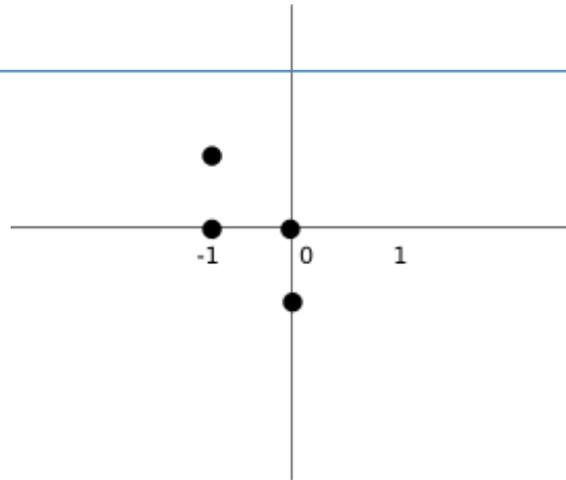
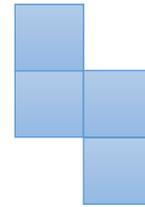
Tetrominoe

```
protected enum Tetrominoe { NoShape, ZShape, SShape, LineShape,  
    TShape, SquareShape, LShape, MirroredLShape };
```



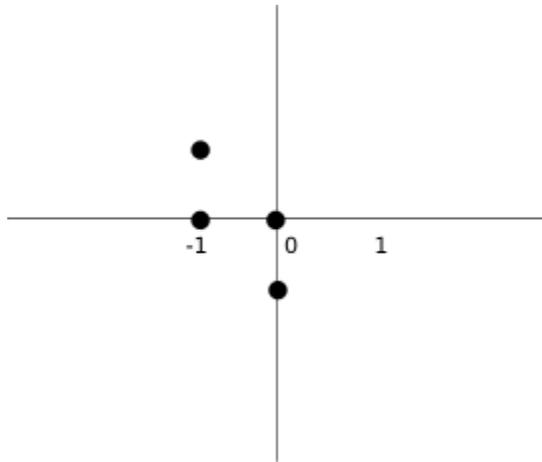
coordsTable for Tetrominoe

```
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 } }  
};
```



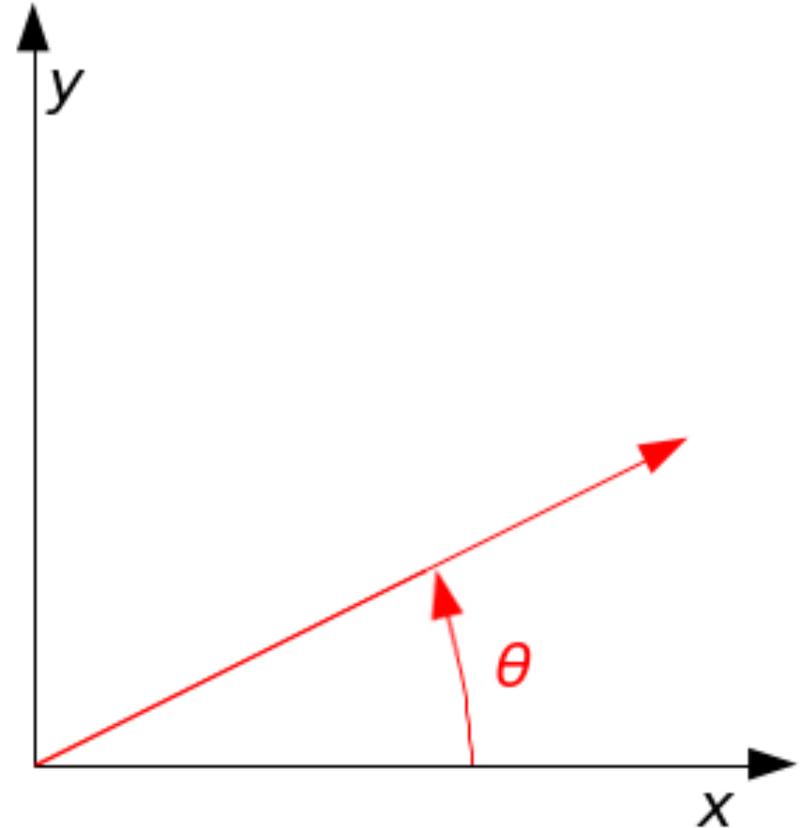
Get Shapes' Coordinates

```
for (int i = 0; i < 4 ; i++) {  
    for (int j = 0; j < 2; ++j) {  
        coords[i][j] = coordsTable[shape.ordinal()][i][j];  
    }  
}
```



Rotation Matrix

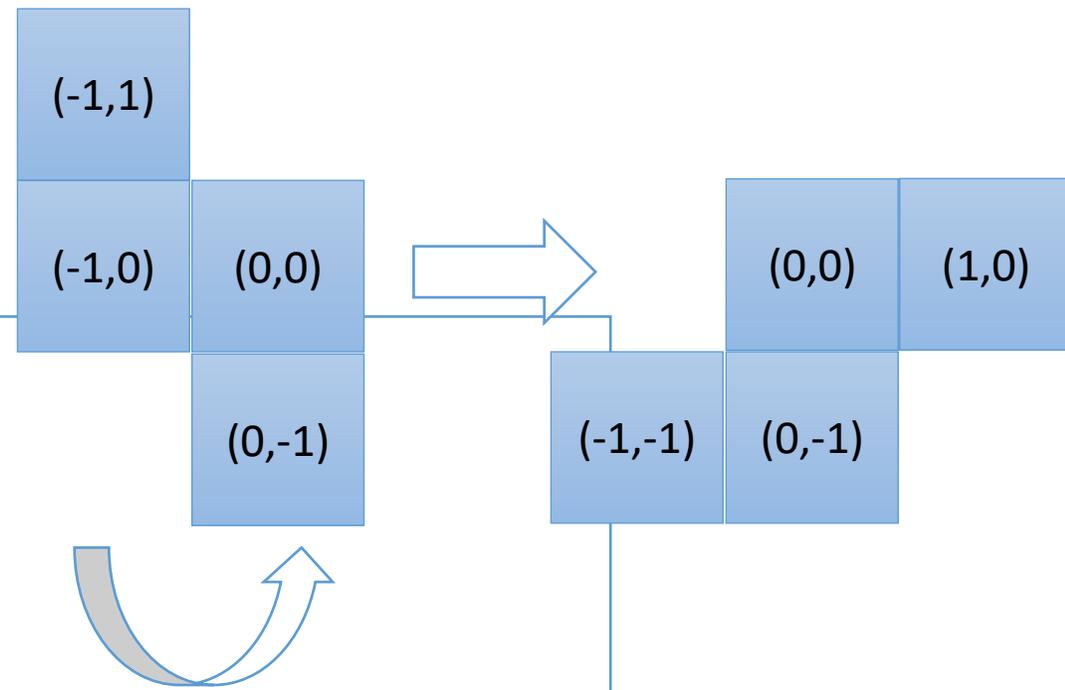
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Rotate Right ($x' = -y$, $y' = x$)

```
public Shape rotateRight()
{
    if (pieceShape == Tetrominoe.SquareShape)
        return this;
    Shape 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;
}

private void setX(int index, int x) { coords[index][0] = x; }
private void setY(int index, int y) { coords[index][1] = y; }
public int x(int index) { return coords[index][0]; }
public int y(int index) { return coords[index][1]; }
```



Board.java

Board: JPanel

```
- Timer: timer
- isFallingFinished, isStarted, isPaused : boolean
- numLinesRemoved: int
- curX, curY: int
- curPiece: Shape
- board: Tetrominoe[]

+ Board()
+ paintComponent(g: Graphics)
- squareWidth()
- squareHeight()
- shapeAt(x: int, y: int)
- start()
- pause()
- doDrawing(g: Graphics)
- dropDown()
- oneLineDown()
- clearBoard()
- pieceDropped()
- newPiece()
- removeFullLines()
- doGameCycle()
- update()
- tryMove(newPiece: Shape, newX: int, newY: int): boolean
- drawSquare(g: Graphics, x: int, y: int, shape: Tetrominoe)
```

Initializing Board

```
private void initBoard(Tetris parent) {  
  
    setFocusable(true);  
    timer = new Timer();  
    timer.scheduleAtFixedRate(new ScheduleTask(),  
        INITIAL_DELAY, PERIOD_INTERVAL);  
  
    curPiece = new Shape();  
  
    statusBar = parent.getStatusBar();  
    board = new Tetrominoe[BOARD_WIDTH * BOARD_HEIGHT];  
    addKeyListener(new TAdapter());  
    clearBoard();  
}
```

Updating Game

- Inheriting TimerTask

```
private void doGameCycle() {
    update();
    repaint();
}
private void update() {
    if (isPaused) {
        return;
    }
    if (isFallingFinished) {
        isFallingFinished = false;
        newPiece();
    } else {
        oneLineDown();
    }
}
private class ScheduleTask extends TimerTask {
    @Override
    public void run() {
        doGameCycle();
    }
}
```

Start & Pause

```
public void start() {
    isStarted = true;
    clearBoard();
    newPiece();
}

private void pause() {
    if (!isStarted) {
        return;
    }
    isPaused = !isPaused;
    if (isPaused) {
        statusBar.setText("paused");
    } else {
        statusBar.setText(String.valueOf(numLinesRemoved));
    }
}
```

Two-stage Drawing

```
private void doDrawing(Graphics g) {
    Dimension size = getSize();
    int boardTop = (int) size.getHeight() - BOARD_HEIGHT * squareHeight();
    for (int i = 0; i < BOARD_HEIGHT; ++i) {
        for (int j = 0; j < BOARD_WIDTH; ++j) {
            Tetrominoe shape = shapeAt(j, BOARD_HEIGHT - i - 1);
            if (shape != Tetrominoe.NoShape) {
                drawSquare(g, 0 + j * squareWidth(),
                    boardTop + i * squareHeight(), shape);
            }
        }
    }
    if (curPiece.getShape() != Tetrominoe.NoShape) {
        for (int i = 0; i < 4; ++i) {
            int x = curX + curPiece.x(i);
            int y = curY - curPiece.y(i);
            drawSquare(g, 0 + x * squareWidth(),
                boardTop + (BOARD_HEIGHT - y - 1) * squareHeight(),
                curPiece.getShape());
        }
    }
}
```

drawSquare()

```
private void drawSquare(Graphics g, int x, int y, Tetrominoe shape) {
    Color colors[] = {
        new Color(0, 0, 0), new Color(204, 102, 102),
        new Color(102, 204, 102), new Color(102, 102, 204),
        new Color(204, 204, 102), new Color(204, 102, 204),
        new Color(102, 204, 204), new Color(218, 170, 0)
    };

    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);
}
```

tryMove()

```
private boolean tryMove(Shape newPiece, int newX, int newY) {  
  
    for (int i = 0; i < 4; ++i) {  
        int x = newX + newPiece.x(i);  
        int y = newY - newPiece.y(i);  
  
        if (x < 0 || x >= BOARD_WIDTH || y < 0 || y >= BOARD_HEIGHT) {  
            return false;  
        }  
        if (shapeAt(x, y) != Tetrominoe.NoShape) {  
            return false;  
        }  
    }  
  
    curPiece = newPiece;  
    curX = newX;  
    curY = newY;  
  
    repaint();  
    return true;  
}
```

```
private class TAdapter extends KeyAdapter {
    @Override
    public void keyPressed(KeyEvent e) {
        System.out.println("key pressed");
        if (!isStarted || curPiece.getShape() == Tetrominoe.NoShape) {
            return;
        }
        int keycode = e.getKeyCode();
        if (keycode == KeyEvent.VK_P) {
            pause();
            return;
        }
        if (isPaused) {
            return;
        }
        switch (keycode) {
            case KeyEvent.VK_LEFT: tryMove(curPiece, curX - 1, curY); break;
            case KeyEvent.VK_RIGHT: tryMove(curPiece, curX + 1, curY); break;
            case KeyEvent.VK_DOWN: tryMove(curPiece.rotateRight(), curX, curY); break;
            case KeyEvent.VK_UP: tryMove(curPiece.rotateLeft(), curX, curY); break;
            case KeyEvent.VK_SPACE: dropDown(); break;
            case KeyEvent.VK_D: oneLineDown(); break;
        }
    }
}
```

Dropping Tetrominoe

```
private void dropDown() {
    int newY = curY;
    while (newY > 0) {
        if (!tryMove(curPiece, curX, newY - 1)) {
            break;
        }
        --newY;
    }
    pieceDropped();
}
```

```
private void oneLineDown() {
    if (!tryMove(curPiece, curX, curY - 1)) {
        pieceDropped();
    }
}
```

pieceDropped()

```
private void pieceDropped() {  
  
    for (int i = 0; i < 4; ++i) {  
  
        int x = curX + curPiece.x(i);  
        int y = curY - curPiece.y(i);  
        board[(y * BOARD_WIDTH) + x] = curPiece.getShape();  
    }  
  
    removeFullLines();  
  
    if (!isFallingFinished) {  
        newPiece();  
    }  
}
```

newPiece()

```
private void newPiece() {  
  
    curPiece.setRandomShape();  
    curX = BOARD_WIDTH / 2 + 1;  
    curY = BOARD_HEIGHT - 1 + curPiece.minY();  
  
    if (!tryMove(curPiece, curX, curY)) {  
  
        curPiece.setShape(Tetrominoe.NoShape);  
        timer.cancel();  
        isStarted = false;  
        statusBar.setText("Game over");  
    }  
}
```

Tetris.java

Tetris: JFrame

- JLabel: statusBar
- board: Board

+ Tetris()
+ main()
- initUI()
+ JLabel getStatusBar()

Tetris.java

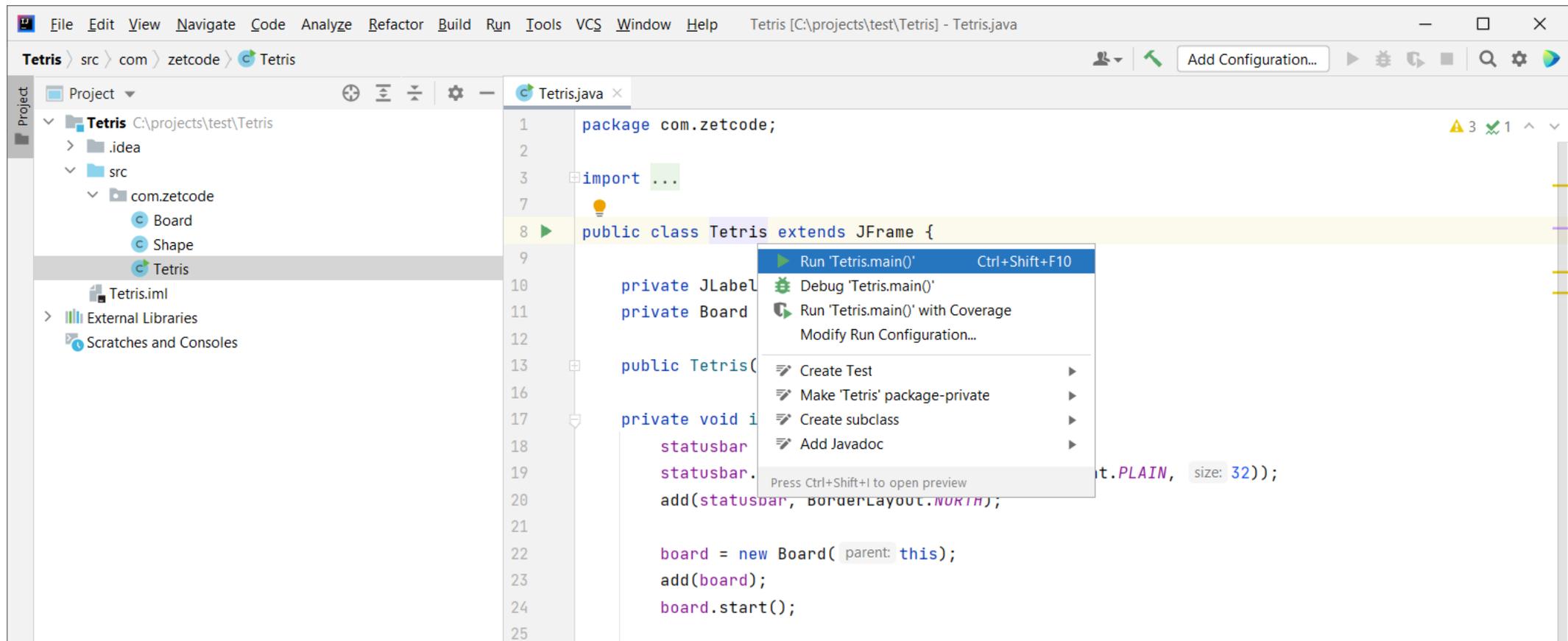
```
public class Tetris extends JFrame {
    private JLabel statusbar;
    private Board board;
    public Tetris() { initUI(); }
    private void initUI() {
        statusbar = new JLabel(" 0");
        statusbar.setFont(new Font("Serif", Font.PLAIN, 32));
        add(statusbar, BorderLayout.NORTH);

        board = new Board(this);
        add(board);
        board.start();

        setTitle("Tetris");
        setSize(400, 800);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }
    public JLabel getStatusBar() { return statusbar; }
    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            Tetris game = new Tetris();
            game.setVisible(true);
        });
    }
}
```

Run Tetris (ALT + enter)

- ALT + enter -> Run Tetris.main()



Tetris > src > com > zetcode > Tetris > rotate

Project

- Tetris
 - src
 - com.zetcode
 - board
 - shape
 - Tetris

Tetris.iml

External Libraries

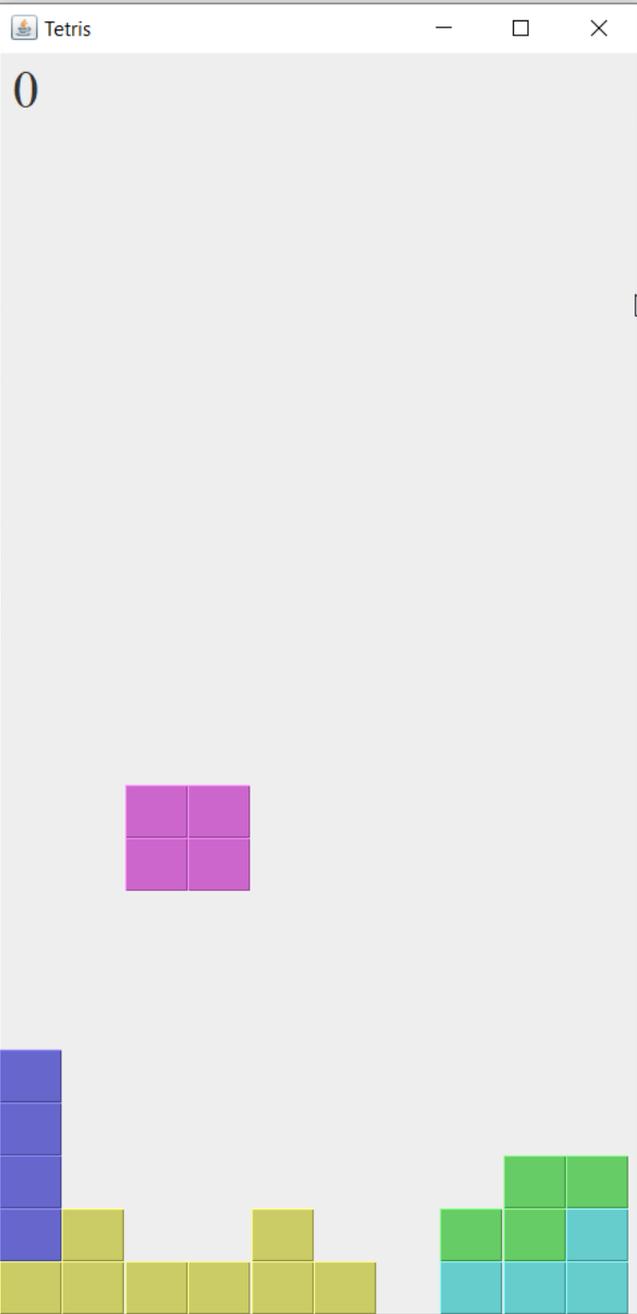
Scratches and Consoles

```

53 public boolean mov
54
55
56
57
58 public boolean rot
59
60 Shape piece;
61 if (clockwise)
62     piece = bo
63 else
64     piece = bo
65 return board.t
66
67
68 public static void
69 EventQueue.inv
70 Tetris gam
71 game.setVi
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Running the Code



```

Piece(), newX: board.curX + x_val, board.curY); }

```

Run: Tetris

- key pressed

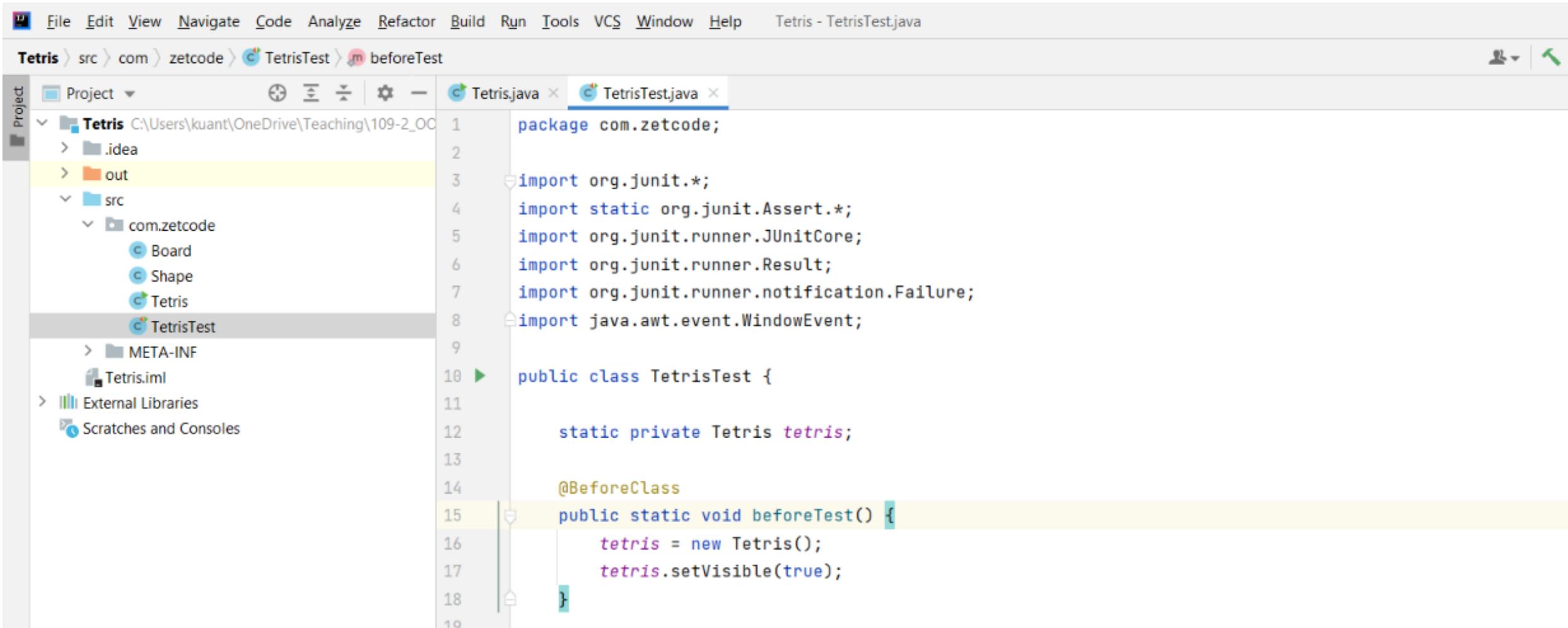
Adding Test APIs



Add Test APIs in Tetris.java

```
public class Tetris extends JFrame {
    private JLabel statusBar;
    private Board board;
    public Tetris() { initUI(); }
    private void initUI() {...}
    public JLabel getStatusBar() { return statusBar; }
    public void dropDown() { board.dropDown(); }
    public boolean isGameOver() { return (statusBar.getText() == "Game over"); }
    public int getLinesRemoved() { return board.getLinesRemoved(); }
    public void restart() {
        board.start();
        statusBar.setText("");
    }
    public boolean move(int x_val) {
        return board.tryMove(board.getCurPiece(), board.curX + x_val, board.curY);
    }
    public boolean rotate(boolean clockwise) {
        Shape piece;
        if (clockwise)
            piece = board.getCurPiece().rotateLeft();
        else
            piece = board.getCurPiece().rotateRight();
        return board.tryMove(piece, board.curX, board.curY);
    }
    public static void main(String[] args) {...}
}
```

Add class TetrisTest



The screenshot shows an IDE window with the following elements:

- Menu Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help. The title bar indicates the file is "Tetris - TetrisTest.java".
- Breadcrumbs:** Tetris > src > com > zetcode > TetrisTest > beforeTest.
- Project View (Left):** Shows the project structure. The "com.zetcode" package is expanded, showing classes "Board", "Shape", "Tetris", and "TetrisTest". "TetrisTest" is selected.
- Code Editor (Right):** Displays the content of TetrisTest.java:

```
1 package com.zetcode;
2
3 import org.junit.*;
4 import static org.junit.Assert.*;
5 import org.junit.runner.JUnit4;
6 import org.junit.runner.Result;
7 import org.junit.runner.notification.Failure;
8 import java.awt.event.WindowEvent;
9
10 public class TetrisTest {
11
12     static private Tetris tetris;
13
14     @BeforeClass
15     public static void beforeTest() {
16         tetris = new Tetris();
17         tetris.setVisible(true);
18     }
19
```

Adding Test Cases

1. `testGameOver()`
2. `testRandomMove()`

```
package com.zetcode;
import org.junit.*;
import static org.junit.Assert.*;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
import java.awt.event.WindowEvent;

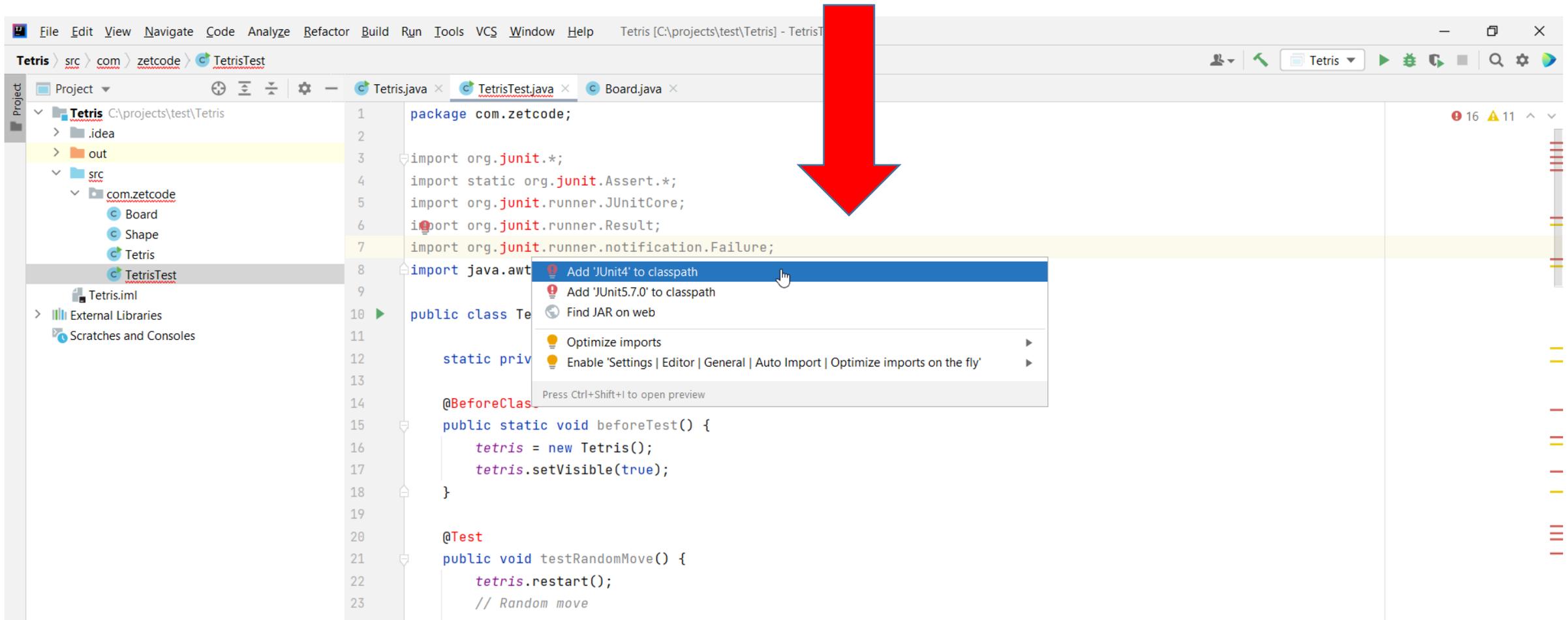
public class TetrisTest {
    static private Tetris tetris;

    @BeforeClass
    public static void beforeTest() {
        tetris = new Tetris();
        tetris.setVisible(true);
    }
    @Test
    public void testRandomMove() {...}
    @Test
    public void testGameOver() {...}

    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(TetrisTest.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());
        // Closing the window after the final result is printed
        tetris.dispatchEvent(new WindowEvent(tetris, WindowEvent.WINDOW_CLOSING));
    }
}
```

Add JUnit4 to classpath (ALT + enter)

- Press “ALT + enter”, select “Add JUnit4 to classpath “



Add default “junit:junit:4.13.1”

The screenshot shows an IDE window for a project named 'Tetris'. The main editor displays the source code for 'TetrisTest.java'. The code includes package declarations and imports for JUnit and Java AWT. A dialog box titled 'Download Library from Maven Repository' is open, showing a search for 'junit:junit:4.13.1'. The search results show 'Found: 1' and 'Showing: 1'. The dialog also includes a 'Download to' field and checkboxes for 'Transitive dependencies', 'Sources', 'Javadocs', and 'Annotations'. The 'Download to' field is set to 'C:\projects\test\Tetris\lib'. The 'Transitive dependencies' checkbox is checked. The 'OK' button is highlighted.

```
1 package com.zetcode;  
2  
3 import org.junit.*;  
4 import static org.junit.Assert.*;  
5 import org.junit.runner.JUnitCore;  
6 import org.junit.runner.Result;  
7 import org.junit.runner.notification.Failure;  
8 import java.awt.event.WindowEvent;  
9  
10 public class TetrisTest {  
11  
12     static private Tetris tetris;  
13  
14     @BeforeClass  
15     public static void setUpBeforeClass() throws Exception {  
16         tetris = new Tetris(10, 20);  
17         tetris.start();  
18     }  
19  
20     @Test  
21     public void testRandomMove() {  
22         tetris.restart();  
23         // Random move
```

Testing Game-over

```
@Test
public void testGameOver() {
    for (int i=0; i<10; i++)
        tetris.dropDown();
    boolean ret = tetris.isGameOver();
    assertTrue(ret);
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
    }
}
```

Testing Random Move

```
public void testRandomMove() {
    tetris.restart();
    // Random move
    int t = 0;
    try {
        while (t < 100) {
            if (Math.random() > 0.5)
                tetris.move(1);
            else
                tetris.move(-1);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
            }
            if (Math.random() > 0.5)
                tetris.rotate(false);
            else
                tetris.rotate(true);
            t++;
        }
    } catch (Exception e) {
        fail();
    }
}
```

Adding API in Board.java

- Making the following variables and functions public
 - `public int curX = 0;`
 - `public int curY = 0;`
 - `public boolean tryMove(Shape newPiece, int newX, int newY) {…}`
 - `public Shape getCurPiece() { return curPiece;}`
 - `public void dropDown() {…}`

Initializing Timer in start()

```
public void start() {  
    isStarted = true;  
    clearBoard();  
    newPiece();  
    numLinesRemoved = 0;  
    timer = new Timer();  
    timer.scheduleAtFixedRate(  
        new ScheduleTask(),  
        INITIAL_DELAY, PERIOD_INTERVAL  
    );  
}
```


Test Result: Game over Test

The screenshot shows an IDE window for a project named "Tetris". The main editor displays the file "TetrisTest.java" with the following code:

```
50 boolean ret =
51 assertTrue(ret)
52 try {
53     Thread.sleep(1000);
54 } catch (InterruptedException e) {}
55 }
56 }
57
58 public static void main(String[] args) {
59     Result result = new Result();
60     for (Failure failure : result.getFailures()) {
61         System.out.println(failure.getMessage());
62     }
63     System.out.println("Game over");
64     // Closing the dispatcher
65     tetris.dispatcher.close();
66 }
67 }
```

The IDE shows a red error bar in the code editor, indicating a test failure. A modal window titled "Tetris" is overlaid on the IDE, displaying a Tetris board. The board is filled with various colored blocks (red, green, blue, cyan, purple, pink). The top row is completely filled with red blocks, and the text "Game over" is displayed at the top of the board. The IDE interface includes a project view on the left, a run console at the bottom, and a status bar at the bottom right showing "58:26 CRLF UTF-8 4 spaces".