**Lab 5**

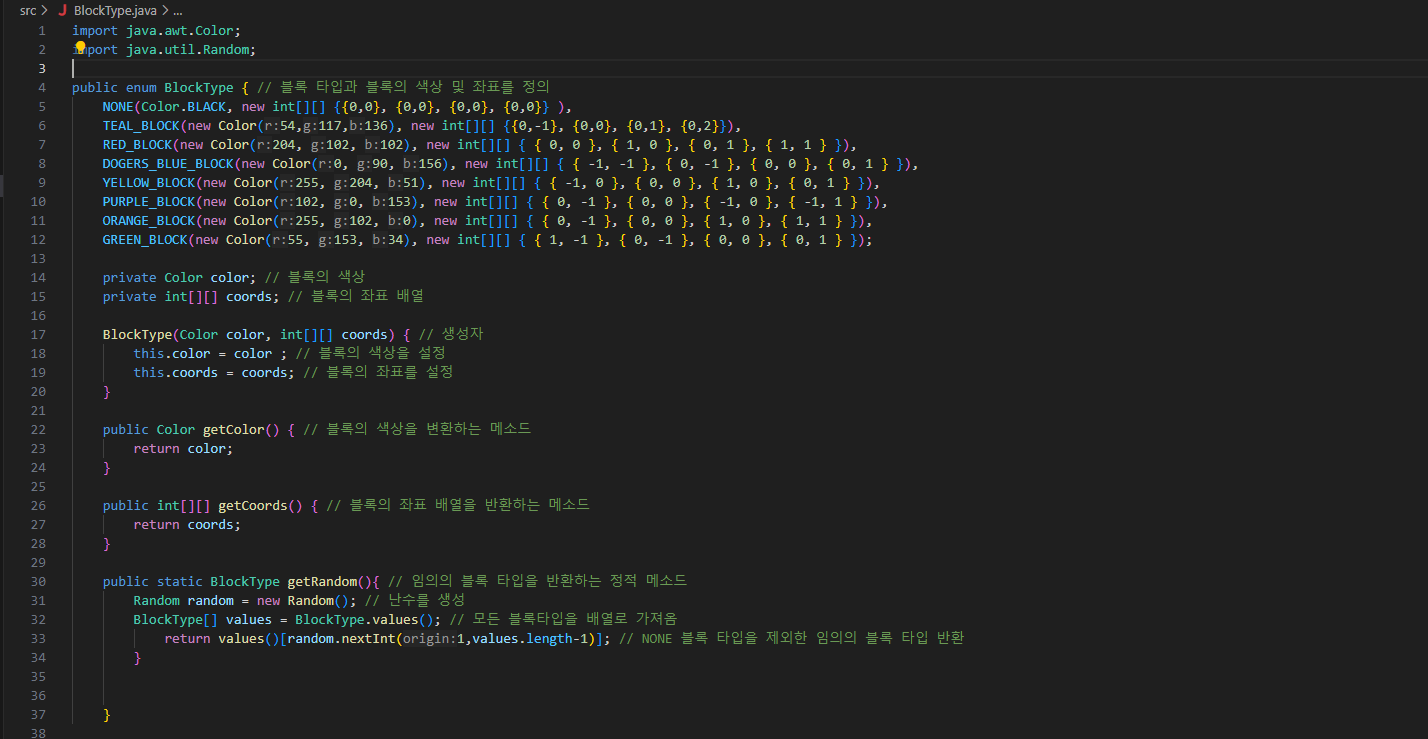
**1분반**

**2024년 06월 03일**

**32211792**

**박재홍**

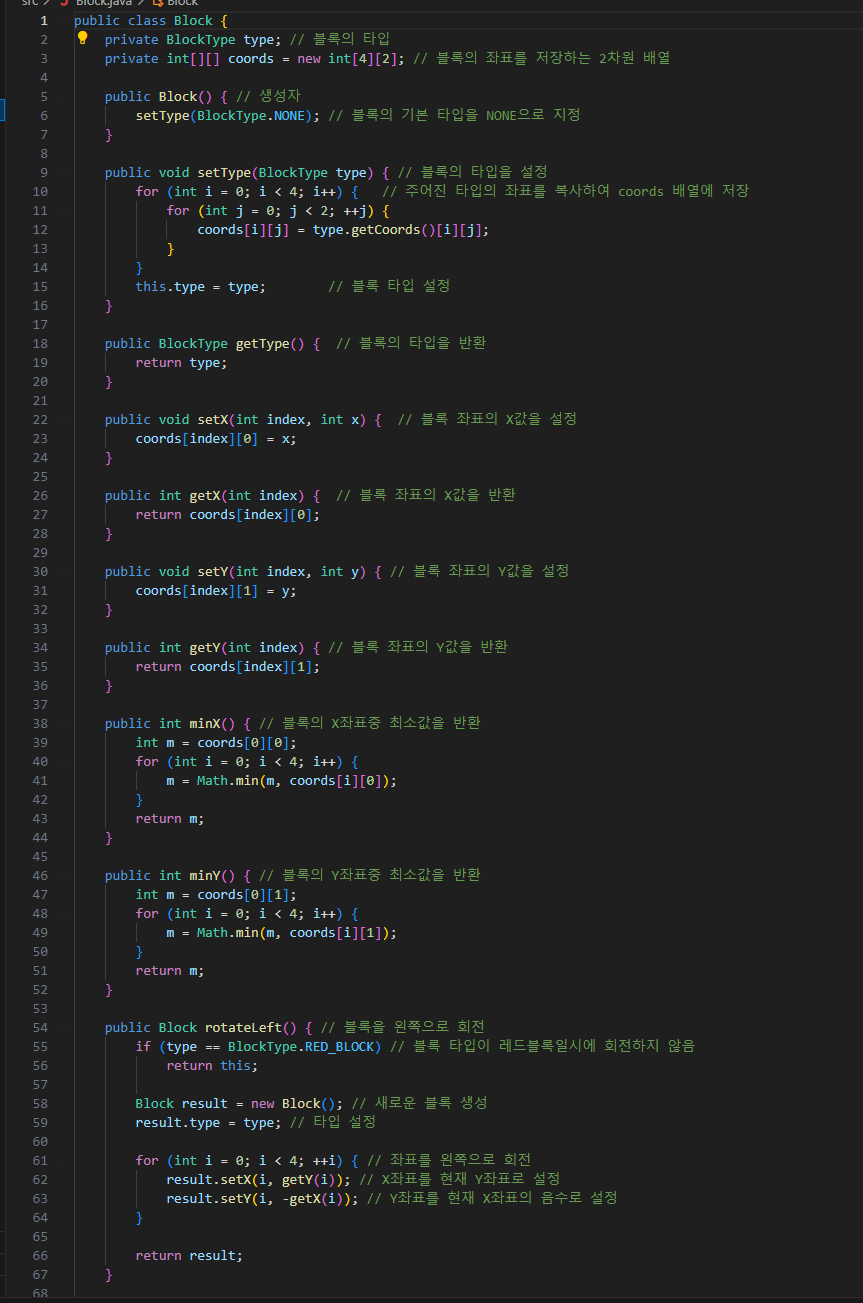
**BlockType 열거형 구현)**

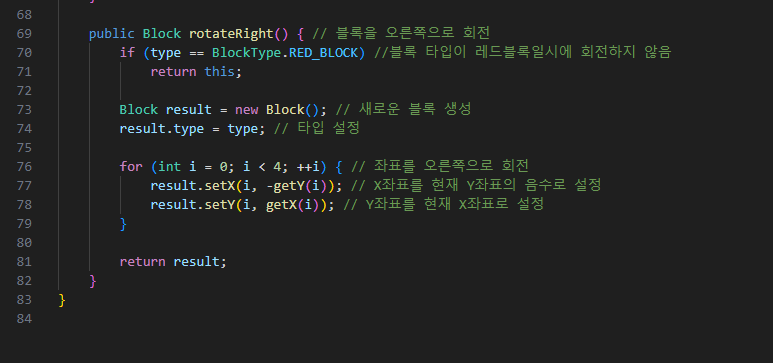


**열거형으로 구현이기에 enum을 이용해 구현해주었고 각 블록의 타입들과 블록의 색상 및 좌표들을 정의해주었다. Private Color color , private int[][] cords 를 사용해 각 블록의 색상과 좌표배열들을 정의했고 그 아래에는 color 와 coords에 대한 생성자, 게터에 대한 내부 구현을 해주었고 마지막엔 Random 메소드를 통해 블록 타입을 랜덤으로 반환하는 정적메소드로 정의해주었다.**

**그리고 BlockType[] values = BlockType.values();를 통해 모든 블록 타입을 배열로 가져오게 했고 retrun values()[random.nextInt(1,values.length-1)];를 통해 블록타입이 랜덤으로 생성될 때 NONE 블록은 생성되지 않도록 배열의 길이에서 -1 해주었다.**

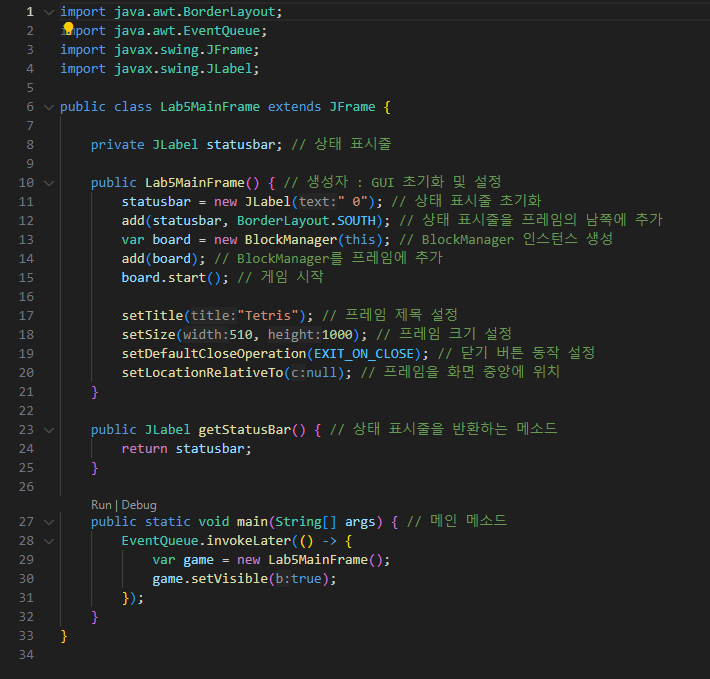
**Block 클래스)**





**처음에 private BlockType type; , private int [][] coords = new int[4][2]; 를 사용해 블록의 타입과 블록의 좌표를 저장해주는 2차원 배열을 정의해주었다. 다음 블록에 대한 생성자를 정의 해주었고 setType(BlockType.NONE)를 통해 블록의 기본타입을 NONE로 정의해주었다. 그 후 public void setType(BlockType type) { for (int i = 0; i<4 i++) { for ( int j = 0; j < 2; j ++ ) { coords[i][j] = type.getCoords()[i][j];}} this.type = type; } 를 사용해 블록의 타입을 설정하고 주어진 타입의 좌표를 복사하여 coords 배열에 저장하였다. 그 다음 type 에 대한 게터를 사용해 블록의 타입을 반환 시켜주었고 게터와 세터를 사용해 X와Y에 대한 좌표의 값을 설정하고 반환해주었고 public int minX() 메소드를 사용해 블록의 X좌표중 최소값을 반환해주었고 public int minY를 사용해 블록의 Y좌표중 최소값을 반환해주었다. 그 다음 public Block rotateLeft() 메소드를 사용해 블록을 왼쪽으로 회전하게 했고 메소드 안에 if문을 사용해 블록타입이 레드블록일 때 블록이 회전하지 않도록 조건을 달았고 for문을 사용해 좌표를 오른쪽으로 회전할 때 X좌표를 현재Y좌표의 음수로 설정, Y좌표를 현재 X좌표로 설정하게 정의했다.**

**Lab5MainFrame 클래스)**



**Lab5MainFrame 클래스는 JFrame을 상속받게 했고 private JLabel statusbar를 선언해 상태 표시줄을 만들었다. 그 후 public Lab5MainFrame() 을 선언해 생성자를 만들었고 GUI 초기화 및 설정을 할 수 있게 했다. Statusbar = new JLabel(“0”); , add(statusbar, BorderLayout.SOUTH); 를 사용해 상태표시줄을 0 으로 초기화 시켰고 상태 표시줄을 프레임의 남쪽에 추가해주었다. Var board = new BlockManager(this); 와 add(board); 를 사용해 BlcokManager에 대한 인스턴스를 생성했고 프레임에 추가해주었다. board.start(); 를 사용해 게임을 시작해주었다. 그 후에 setTitle(“Tetris”)를 사용해 프레임의 제목을 Tetris로 설정해주었고 setSize(510,1000) 을 통해 프레임 크기를 510x1000으로 설정했고 setDefaultCloseOperation(EXIT\_ON\_CLOSE)를 통해 닫기버튼 동작 설정을 해주었고 setLocatinRelativeTo(null)를 통해 프레임이 화면 중앙에 위치하도록 설정해주었다. 그 후 getStatusBar 메소드를 통해 상태 표시줄을 반환해주었다. 메인 메소드에선 Lab5MainFrame 객체를 생성하고 이를 화면에 표시하는 작업을 수행하게 해주었다.**

**BlockEvent 클래스 구현)**

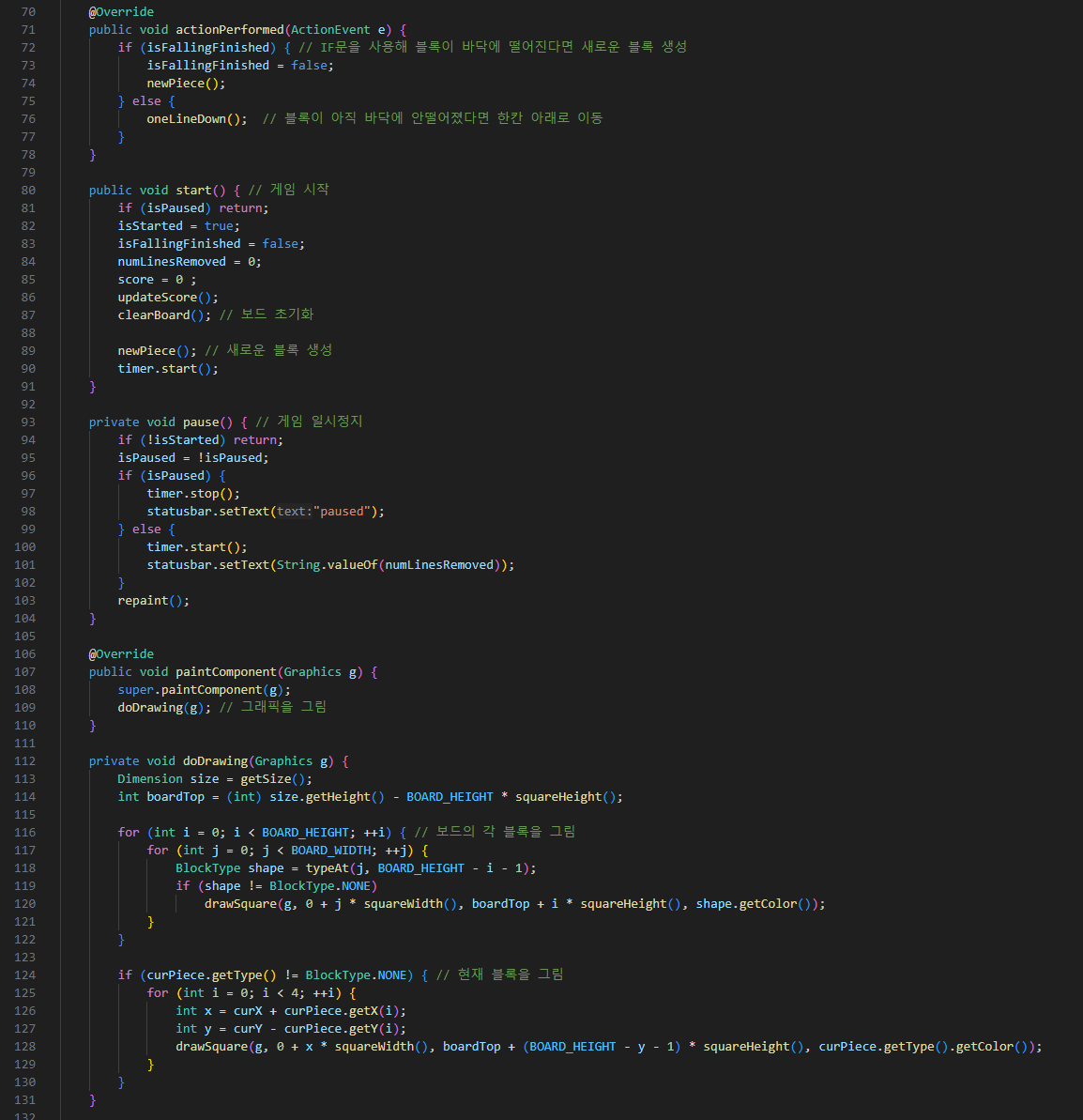


**Private String name, private BlockType type, private List<String> actions를 사용해 블록의 이름과 블록의 타입과 블록 관련 액션 목록들을 선언해주었고 그 아래에 name, type, actions 에 대한 생성자를 선언하고 name, type, actions에 대한 게터와 세터를 선언해 블록 이름을 설정하고 반환하는 메소드를 선언하고 블록 타입을 설정하고 반환하는 메소드를 선언하고 액션 목록을 설정하고 반환하는 메소드를 선언해주었다. 마지막엔 toString 메소드를 통해 객체의 문자열 표현을 반환하게 해주었고 type 이나 actions의 값이 null 일경우 null 로 출력되게 선언해주었다.**

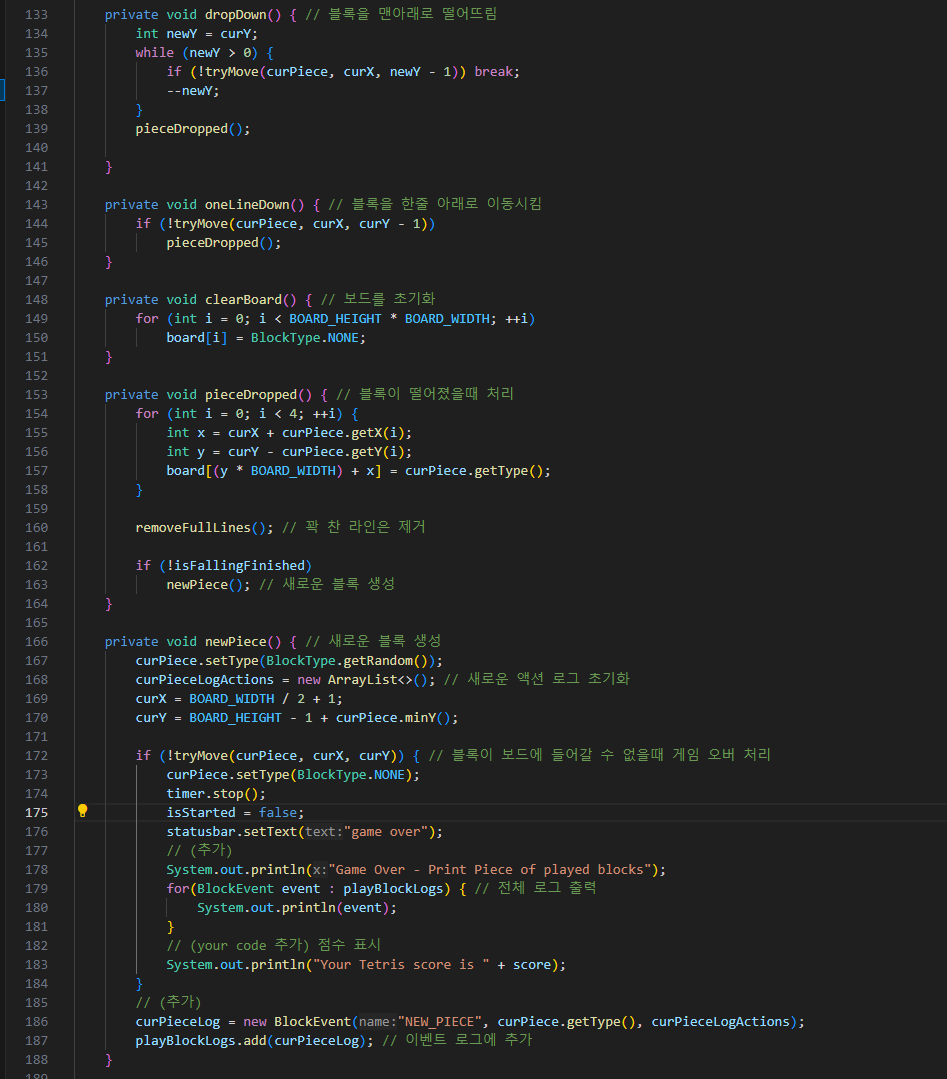
**BlockManager 클래스)**



**BlockManager는 JPanel 클래스를 상속 받고 ActionListener 과 KeyListener 인터페이스를 구현하여 특정 이벤트를 처리하도록 했다. 그 다음 private final int BOARD\_WIDTH = 10; 를 사용해 게임 보드의 넓이를 10으로 설정, private final int BOARD\_HEIGHT = 22; 를 사용해 게임 보드의 높이를 22로 설정, private final int PERIOD\_INTERVAL = 300; 를 사용해 타이머 간격을 300으로 설정, private final int BLOCK\_SIZE = 50; 를 사용해 블록 크기를 50으로 설정해주었다. 그 다음 private Timer timer 를 사용해 게임 루프 타이머를 선언, private Boolean isFallingFinished= false 를 사용해 블록이 떨어지는것에 대해 끝났는지에 대한 여부, private Boolean isStarted = false 를 사용해 게임이 시작됐는지에 대한 여부, private Boolean isPaused = false 를 사용해 게임이 멈췄는지에 대한 여부, private int numLinedRemover =0을 사용해 제거된 라인의 수를 0으로 초기화, private int curX =0 를 사용해 현재 블록의 X좌표, private int curY = 0 를 사용해 현재 블록의 Y좌표 , private Block curPiece를 사용해 현재 블록의 객체, private BlockType [] board 를 사용해 게임 보드 배열, private JLabel statusbar 를 사용해 상태표시줄을 선언해주었고 private ArrayList<BlockEvent>playBlockLogs 를 사용해 블록 이벤트 로그 목록을 선언했고 private BlockEvent curPiece 를 사용해 현재 블록 이벤트 로그를 선언했고 private ArrayList<String> curPieceLogActions 를 사용해 현재 블록 이벤트 로그에 대한 액션 목록을 선언했다. 이 3가지를 선언해줌으로써 BlockManger 클래스에 BlockEvent 클래스와 관련된 멤버들을 추가시켜주었다. 그 다음 BlockManger의 생성자를 선언해주었고 curpiece = new block() 을 사용해 새로운 블록을 생성하게 했고 timer = new Timer(PERIOD\_INTERVAL, this) 를 사용해 타이머를 생성해주었다. 그 후 status = parent.getStatusBar()를 사용해 부모프레임으로부터 상태표시줄을 가져왔다. 그 후 board = new BlockType[BOARD\_WIDTH \* BOARD\_HEIGHT]를 사용해 보드 배열 초기화시켜주었고 addKeyListener(this)를 사용해 키 리스너를 추가시켰고 clearBoard()를 사용해 보드를 초기화 시켜주었고 playBlockLogs = new ArrayList<>()를 사용해 블록 이벤트 로그 목록을 초기화 시켜주었다. Private int squareWidth() 과 private int squareHeight() 를 사용해 블록 넓이와 높이를 반환시켜주었다. 그 다음 private BlockType typeAt(int x, int y){ return board[(y \* BOARD\_WIDTH) + x]; 를 사용해 주어진 좌표의 블록 타입을 반환시켜주었다.**



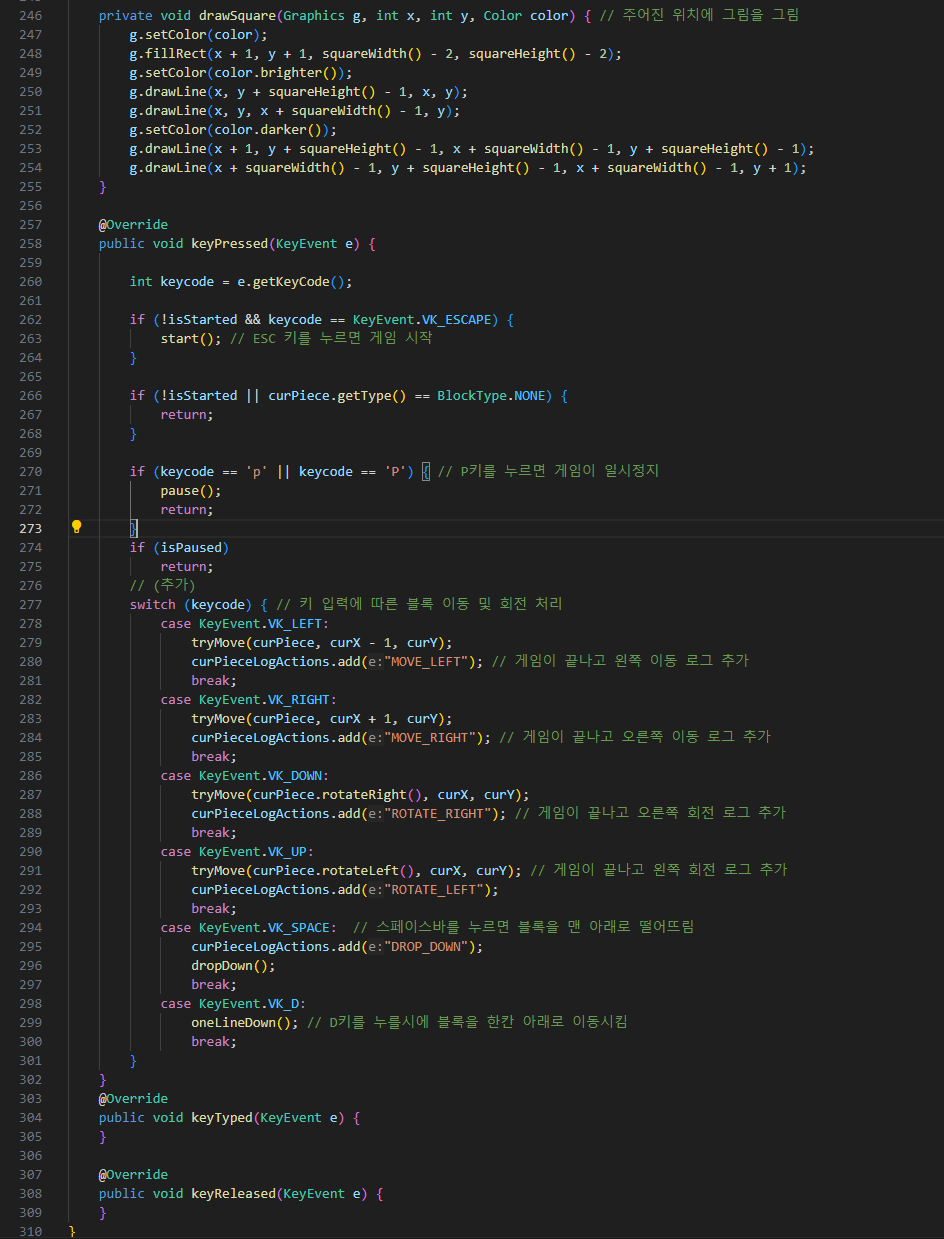
**actionPerformed 메소드 안에서는 if문을 사용해 블록이 바닥에 떨어지면 새로운 블록을 생성하도록 선언했고 블록이 아직 바닥에 안떨어졌다면 한칸 아래로 이동하도록 설정했다. 그 다음 start 메소드 안에서는 게임이 시작 되도록 선언했고 점수, 제거된 라인의 수를 0으로 초기화 시켰고 보드를 초기화 하도록 선언했고 새로운 블록이 생성되도록 선언해주었다. 그 다음 pause 메소드 안에서는 게임이 일시정지 되도록 선언해주었다. Statusbar.setText(“PAUSED”)를 사용해 게임이 일시정지 되었을 때 paused라고 뜨도록 설정해주었다. paintComponent 메소드를 사용해 그래픽을 그리도록 설정했고 doDrawing 메소드에서는 보드의 이중 for문과 if 문을 사용해 보드의 각 블록을 그리도록 설정했고 if-for문을 사용해 현재 블록을 그리도록 설정했다.**



**dropDown 메소드 안에서는 while 문과 if문을 통해 블록이 맨아래로 떨어지도록 선언했고 oneLineDown 메소드 안에서는 블록을 한줄 아래로 이동시킬 수 있도록 선언해주었고 clearBoard 메소드 안에서는 보드를 초기화 시킬 수 있도록 선언했고 pieceDropped 메소드 안에서는 블록이 떨어졌을 때 처리할 수 있게 선언했고 removeFullLines()를 사용해 꽉 찬 라인은 제거 되도록 설정했고 newPiece()를 사용해 새로운 블록이 생성 되도록 선언해주었다. 그 다음 newPiece 메소드 안에서는 새로운 블록이 생성되도록 선언해주었고 curPieceLogActions = new ArrayList<>()를 사용해 새로운 액션 로그가 초기화 되도록 선언해주었고 if문을 통해 블록이 보드에 들어갈 수 없을 때 타이머가 멈추고 게임이 멈추고 game over라고 뜨도록 설정해주었다. 그 후 결과창에 포문을 사용해 “Game over – Print Piece of played blocks” 라는 문구와 함께 테트리스를 하면서 사용했던 전체 로그들을 출력되게 설정하였다. 그리고 curPieceLog= new BlockEvent(“NEW\_PIECE” , curPiece.getType() , curPieceLogActions); 와 playBlockLogs.add(curPieceLog)를 사용해 playBlockLogs에 curPieceLog가 추가 되도록 선언했다.**

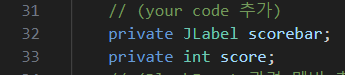


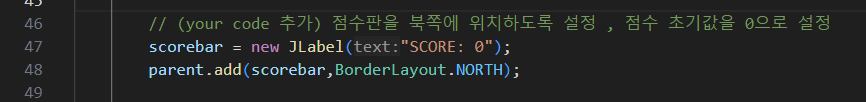
**tryMove메소드 안에서는 for문과 if문을 사용해 블록을 새로운 위치로 이동시킬 수 있게 선언해주었다. removeFullines 메소드에서는 처음에 int numFullLines =0 ; 을 사용해 numFullLines를 0으로 초기화 시켜주었고 for문을 통해 라인이 꽉 찼을 경우 꽉 찬 라인은 삭제 하고 ++numFullLines; 을 통해 라인이 꽉 차서 사라졌을 경우 numFullLines가 1증가 하도록 선언해주었다. 그 다음에 if문을 사용해 numLinesRemoved += numFullLines;를 사용해 게임 실행중 왼쪽 하단에 표시된 numLinesRemoved가 지워진 라인 하나당 1씩 증가하도록 설정해주었다. 그리고 playBlockLogs.add(new BlockEvent(”REMOVE\_FULL\_LINE”, null, null)); 를 사용해 게임종료시에 결과창에 REMOVE\_FULL\_LINE,null,null 이 표시되도록 설정해주었다.**

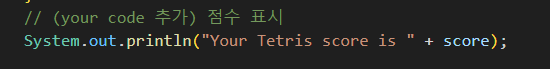


**drawSquare 메소드에서는 블록 타입에 따라 주어진 위치에 그림을 그리도록 설정해주었다. keyPressed 메소드에서는 if(!isStarted && keycode == KeyEvent.VK\_ESCAPE) { start()}; 를 사용해 ESC키를 누르면 게임이 실행 되도록 선언해주었고 if( keycode == 'p' || keycode == 'P') { pause();} 를 사용해 p키를 누르면 게임이 일시정지 되도록 선언해주었다. 그 후 Switch 문을 사용해 키 입력에 따라 블록 이동 및 블록 회전 처리가 되도록 설정해주었다. case KeyEvent.VK\_LEFT : tryMove(curPiece , curX -1, curY); curPieceLogActions.add(”MOVE\_LEFT”); 를 사용해 왼쪽 방향키를 누르게 되면 블록이 왼쪽으로 회전하고 게임이 끝나고 결과창 action 로그에서 MOVE\_LEFT가 추가되도록 설정해주었다. case KeyEvent.VK\_RIGHT : tryMove(curPiece , curX + 1, curY); curPieceLogActions.add(”MOVE\_RIGHT”); 를 사용해 오른쪽 방향키를 누르게 되면 블록이 오른쪽으로 회전하고 게임이 끝나고 결과창 action 로그에서 MOVE\_RIGHT가 추가되도록 설정해주었다. case KeyEvent.VK\_DOWN : tryMove(curPiece.rotateRight() , curX , curY); curPieceLogActions.add(”ROTATE\_RIGHT”); 를 사용해 아래쪽 방향키를 누르게 되면 블록이 오른쪽으로 회전하고 게임이 끝나고 결과창 action 로그에서 ROTATE\_RIGHT가 추가되도록 설정해주었다. case KeyEvent.VK\_UP : tryMove(curPiece.rotateLeft() , curX , curY); curPieceLogActions.add(”ROTATE\_LEFT”); 를 사용해 위쪽 방향키를 누르게 되면 블록이 왼쪽으로 회전하고 게임이 끝나고 결과창 action 로그에서 ROTATE\_LEFT가 추가되도록 설정해주었다. case KeyEvent.VK\_SPACE : curPieceLogActions.add(”DROP\_DOWN”); dropDown(); 을 사용해 스페이스바를 누르면 블록이 맨 아래로 떨어지고 게임이 끝나고 결과창 action 로그에서 DROP\_DOWN이 추가되도록 설정해주었다. case KeyEvent.VK\_D : oneLineDown();을 사용해 D키를 누를시에 블록을 한칸 아래로 이동시킬 수 있게 해주었다.**

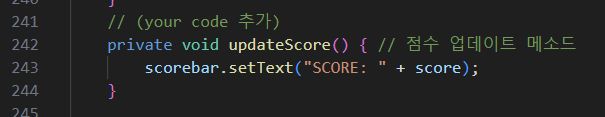
**Your Code 구현)**

****

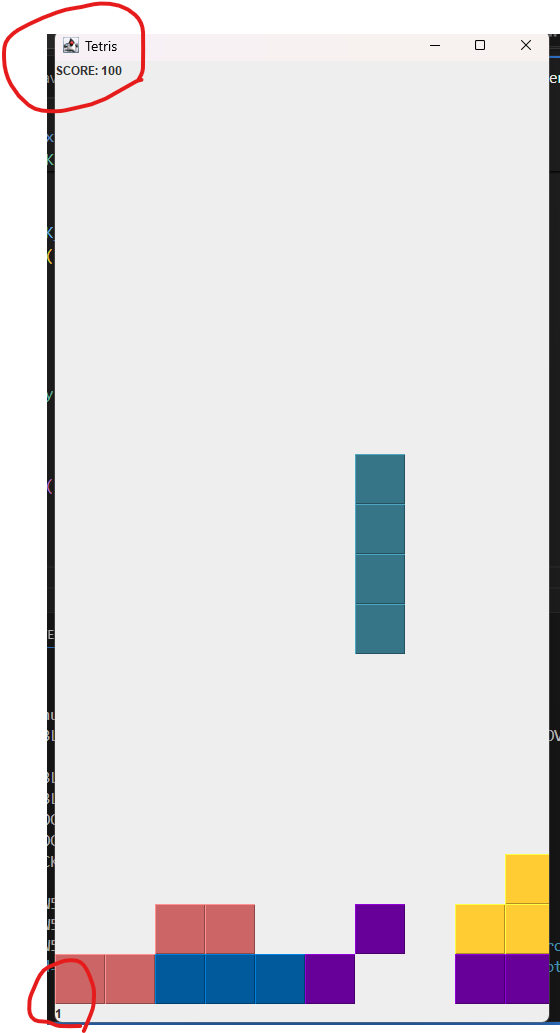
****

****

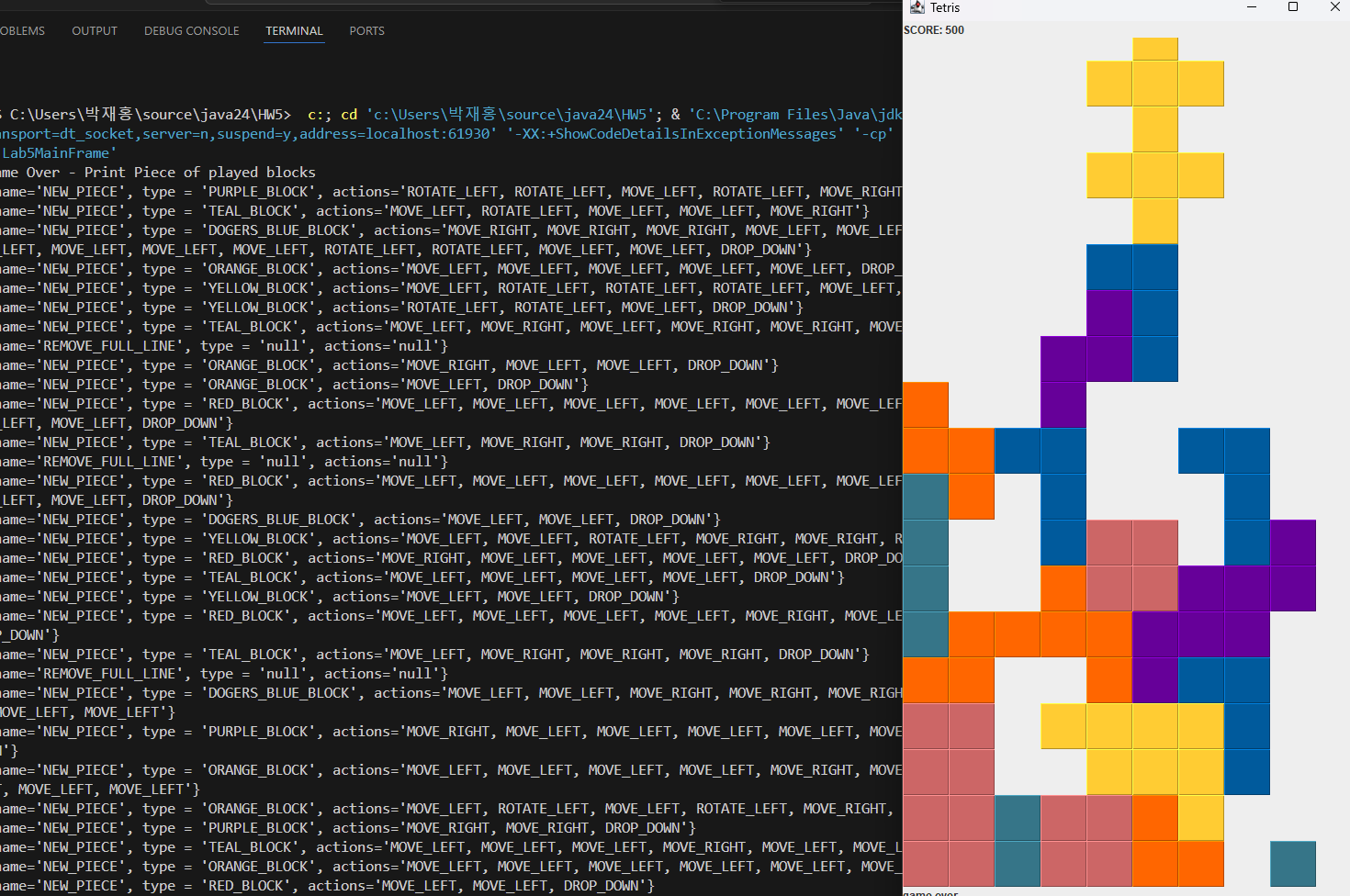
****

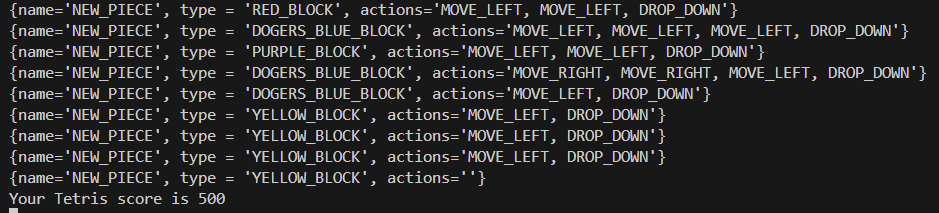
****

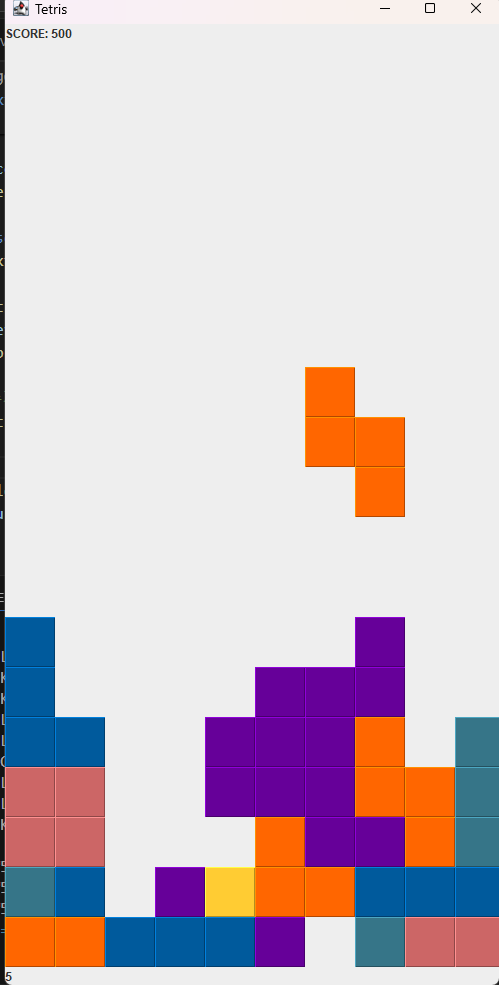
**Your Code 로는 테트리스 게임을 하는동안 점수를 알 수 있게 구현해주었다. 우선 scorebar 와 score를 선언해주었고 그 다음 JLabel를 사용해 왼쪽 위에 점수판이 위치하도록 해주었고 처음에 점수 초기값을 0으로 설정했고 SCORE : 0 으로 표시되도록 선언해주었다. 그 후 updateScore 메소드를 선언해주었고 지워진 라인 하나당 점수가 100점씩 오르도록 선언해주었다. 그 후 System.out.println(”Your Tetris score is “ + score) 를 통해 게임 종료 후 결과창에서 점수가 표시되도록 설정해주었다.**

****

**결과창)**

****

****

****

**게임을 시작하고 게임이 종료되었을때 블록의 타입, 블록이 했던 행동들에 대한 로그들을 출력하게 했고 한 줄이 꽉차 라인이 지워질시에 name은 REMOVE\_FULL\_LINE type 과 actions 는 null로 출력 되게 해주었고 왼쪽 하단에 지워진 줄의 수에 +1 하도록 만들었고 그에 따라 지워진 줄 수 x 100 점으로 왼쪽 상단에 점수가 표기되게 해주었다. 그리고 게임이 끝나고 점수를 알 수 있게 출력까지 되게 설정해주었다.**