# REPORT


**과목명** 자바기반응용프로그래밍

**학과** 컴퓨터공학과
**학번** 12173896
**이름** 김아영

# 테트리스 보고서

## 1. 구현 방법

### 1) 블록들의 이동 및 키보드를 통한 이동

블록들의 이동은 'moving' 이라는 함수를 'move'라는 함수가 받아서 사용한다. 이때 'move'에는 각 키보드의 문자를 눌렀을 때 호출이 되게 하는 'KeyListener1'이라는 함수를 받아 사용한다. 처음 move라는 함수에서 moveing(n)을 받는데 이때 n은 move의 왼쪽과 오른쪽을 받는 함수이다. n=0일 경우 왼쪽, n=1일 경우 오른쪽을 나타낸다. 또한 왼쪽으로 가게 되면 x를 –해주고 오른쪽으로 가게 되면 x를 ++해준다. Moving은 왼쪽 벽면과 오른쪽 벽면을 넘어가지 않게 해주는 역할을 하고 있다.

```
class Keylistener1 implements KeyListener{
	static public Keylistener1 instance = new Keylistener1();

	int[] k = new int[100];
	@Override
	public void keyPressed(KeyEvent e) {
		int code = e.getKeyCode();
		k[code]=1;
	}
	@Override
	public void keyReleased(KeyEvent e) {
		int code = e.getKeyCode();

		k[code] = 0;
	}
	@Override
	public void keyTyped(KeyEvent arg0) {
	}
	public boolean isKeyDown(int e) {
		if (k[e] == 1) {
			k[e] = 2;
			return true;
		}
		return false;
	}
}
```

```
int moveing(int n) {
		for(int i=0; i<3; i++) {
			for(int j=0; j<3; j++) {
				if(p.blocks[i][j] == 1) {
					if(n == 0 &&(show(p.y+i,p.x+j-1) == 1 ||
p.x+j < 1)) {
```

```
                                        return 0;

                                    }else if(n == 1 &&(show(p.y+i,p.x+j+1) == 1
|| p.x+j > 8)) {
                                        return 0;
                                    }
                                }
                            }
                        }
                        return 1;
                }

void move() {
                    if(Keylistener1.instance.isKeyDown(KeyEvent.VK_LEFT)) {
                        if(moveing(0)==1) {
                            p.x--;
                        }
                    }
                    else if(Keylistener1.instance.isKeyDown(KeyEvent.VK_RIGHT)) {
                        if(moveing(1)==1) {
                            p.x++;
                        }
                    }
                    if(Keylistener1.instance.isKeyDown(KeyEvent.VK_UP)) {
                        p.cycle(1);
                    }
                    if(Keylistener1.instance.isKeyDown(KeyEvent.VK_DOWN)) {
                        while(fall()==1);
                    }
                }
```

2) 랜덤블록과 회전

블록들은 각각 배열로 총 6가지 블록을 만들어 줬습니다. 각 6가지 블록은 random을 써 랜덤하게 나오게 했습니다. 랜덤한 블록들을 통해 회전을 시켰다. 회전의 방법은 각각 배열로 저장을 했기 때문에 보여지는 배열과 블록의 임의의 값을 저장하는 배열을 만들어 90도씩 배열에 맞게 돌리고 돌린 배열의 값을 받아 저장한다고 생각하면 된다. 또한 벽에서 돌릴 때 넘어가지 않게 처리도 해주었다. 만약 돌리면서 벽에 닿으면 'l'의 값을 '0'으로 처리를 해주었고 'l'의 값이 '1'이 될 때만 잘 작동되는 블록으로 저장하게 해주었다.

```
void cycle(int a) {
                    int [][]realshape = new int[3][3];
                    int [][]temp = blocks;
                    if(num == 0 || num == 1 || num == 2 ||num == 3 ||num == 4 || num ==
5 || num == 6) {
                        if( a > 0) {
                            for(int i=0 ;i<3; i++) {
                                for(int j=0; j<3; j++) {
                                    realshape[j][2-i] = temp[i][j];
                                }
```

```
                        }
                    }
                }

                int l = 1;
                .
                for(int i=0; i<realshape.length; i++) {
                    for(int j=0; j<realshape[0].length; j++) {
                        if(Boardtetris.panel[y+i][x+j] == 1 &&
realshape[i][j] == 1) {
                            l = 0;
                        }
                    }
                }

                if(l == 1) {
                    blocks = realshape;
                }
        }
```

3) 블록삭제와 점수 획득

블록을 삭제해주려면 일단 count을 이용해 블록이 다 채워졌는지를 확인한다. Count의 값이 10이 될 경우 블록이 채워졌고 블록의 한 줄을 삭제한다. 또한 블록의 한 줄 삭제를 할 경우 점수 획득이라는 문장이 나오도록 코딩을 했다.

```
void clear() {
                for(int i=0; i<3; i++) {
                    for(int j=0; j<3; j++) {
                        if(p.blocks[i][j] == 1) {
                            panel[p.y+i][p.x+j] = 1;
                        }
                    }
                }
                p = null;
                p = new Point();
                int sum=0;
                for(int i=0; i<20; i++) {
                    int count = 0;
                    for(int j=0; j<10; j++) {
                        if(panel[i][j] == 1) {
                            count++;
                        }
                    }
                    if(count == 10) {
                        int [][]tmp = panel;
                        panel = new int[20][10];
                        int n = 19;
                        for(int ch=19; ch>=0; ch--) {
                            if(ch != i) {
                                panel[n] = tmp[ch];
                                n--;
```

```
                                        }
                                        p.score++;
                                }
                                System.out.println("점수 획득 ");
                                sum=sum+p.score;
                                System.out.println(sum);
                        }
                }
        }
```

4) 블록이 내려옴

블록이 내려오게 하는 과정은 블록이 가장 바닥일 때 즉 p.y+1이 19일 때 블록이 멈추게 하면서 y++을 통해 내려오게 한다.

```
int fall() {
                p.y++;
                for(int i=0; i<3; i++) {
                        for(int j=0; j<3; j++) {
                                if(p.blocks[i][j] == 1) {
                                        if(p.y+i == 19) {
                                                clear();
                                                return 0;
                                        }else if(show(p.y+i+1,p.x+j) == 1){
                                                clear();
                                                return 0;
                                        }
                                }
                        }
                }
                return 1;
        }
```

5) 파일 저장, 가져오기

파일 저장과 가져오는 방법은 잘 구현하지 못해 메모장과 같은 형태로 디자인을 해보았다.

```
void makeMenu() {
                JMenuItem item;
                KeyStroke key;
                JMenuBar mb=new JMenuBar();
                JMenu m1=new JMenu("저장하기");
                m1.setMnemonic(KeyEvent.VK_F);
                JMenu m2=new JMenu("불러오기");
                m2.setMnemonic(KeyEvent.VK_C);
                item=new JMenuItem("새로만들기",KeyEvent.VK_N);
                item.addActionListener(this);
                m1.add(item);
```

```
        item=new JMenuItem("저장",KeyEvent.VK_S);
        item.addActionListener(this);
        m1.add(item);
        item=new JMenuItem("다른 이름으로 저장",KeyEvent.VK_A);
        item.addActionListener(this);
        m1.add(item);
        m1.addSeparator();
        m1.add(new JMenuItem("종료"));
        m1.add(item);
        mb.add(m1);
        mb.add(m2);

        setJMenuBar(mb);
}
public static void main(String[] args) {
        new Tetris();
}
@Override
public void actionPerformed(ActionEvent e) {
        JMenuItem mi=(JMenuItem)(e.getSource());
        switch (mi.getText()) {
        case "새로만들기":
                System.out.println("새 파일을 눌렀습니다.");
                break;
        case "저장":
                System.out.println("저장을 눌렀습니다.");
                break;
        case "다른 이름으로 저장":
                System.out.println("다른 이름으로 저장을 눌렀습니다.");
                break;
        }
}
```

## 2. 소감

처음 자바를 통해 테트리스를 만들면서 어려운 부분이 많았다. 어떻게 구현을 해야 할까 고민이 많이 되었고 계속 바꾸면서 구현을 했다. 이렇게 구현하면 될 것 같지만 막상 안되는 부분에 있어서 힘들었다. 하지만 계속 수정을 하고 더 추가하고 필요 없는 부분은 제거하면서 잘 구현을 되었을 때는 뿌듯했다. 저장과 가져오기와 같이 비록 완벽하게 완성을 하지 못했지만 최대한 구현을 하려고 했고 다른 동작에서 구현이 된 부분에 대해서는 뿌듯하다.