자바기반응용프로그래밍
기말과제
보고서

과목명: 자바기반응용프로그래밍
교수님: 임광수 교수님
분반: 006분반
학과: 컴퓨터공학과
학번: 12191656
이름: 이채연

목차

1. 서론

1.1 설계 주제 및 목적

2. 본론

2.1 기능 구현 알고리즘

서론

1.1 설계 주제 및 목적

2020년 2학기 자바기반응용프로그래밍 전공과목의 기말 실습 주제는 자바로 구현하는 테트리스 만들기이다.
자바기반응용프로그래밍 강의에서 배운 이벤트처리, 스레드, 파일입출력을 바탕으로 테트리스를 구현한다.

테트리스의 구현 목적은 총 6가지이다.
첫 번째, 테트리스 실행 GUI을 만든다.
두 번째, 키보드의 입력을 받아 블록을 이동한다.
세 번째, 맨 아래 블록 한 줄이 채워졌을 때 아래 블록 한 줄은 삭제된다.
네 번째, 한 블록이 내려와서 착지할 때마다 블록들의 정보를 파일에 기록하고, 다음 블록이 내려오기 전에 파일에서 블록들의 정보를 가져온다.
다섯 번째, 블록의 유형과 색깔은 랜덤으로 지정한다.
여섯 번째, 특정 키보드를 입력했을 때 블록이 회전한다.
일곱 번째, 테트리스 종료 조건과 같이, 블록이 위까지 다 차면 게임은 종료된다. 그리고 GUI 창 맨 위의 X버튼을 누르면 프로그램이 종료한다.

본론

2.1 기능 구현 알고리즘

Tetris 프로그램은 main함수에서 Tetris class의 객체를 생성하고, Tetris class의 생성자가 호출되면서 테트리스 게임이 실행된다.

Tetris class안에 있는 멤버변수는 다음과 같다.

```
//버튼객체의 좌표
public JButton jb[][];
//각 버튼에 색이 있으면 1, 화이트이면 0
public Boolean state[][];

//랜덤으로 색깔을 입히기 위한 배열
Color c1 = new Color(255,255,000);//노란색
Color c2 = new Color(153,153,255);//연한 보라색
Color c3 = new Color(153, 255,153);//연한 연두색
Color c4 = new Color(255,153,255);//연한 핑크색
Color c5 = new Color(153,255,255);//연한 하늘색
Color c6 = new Color(255,153,153);//연한 빨간색
Color c7 = new Color(153,204,255);//연한 파란색
public Color blockColor[]={c1, c2, c3, c4, c5, c6, c7};

//내려가는 블록의 cell들의 위치를 저장
public int x[] = new int[4];
public int y[] = new int[4];

//내려가는 도중의 cell 위치의 변화값 저장
public int plusX = 0;
public int plusY = 0;

//처음위치가 0, 시계방향으로 rotation은 0->1->2->3 으로 변함
public int rotation = 0;
```

우선 JButton 객체로 된 2차원 배열 jb는 블록의 각 cell을 담당한다.
그리고 Boolean 타입의 2차원 배열 state는 각 cell에 블록이 차있는지 여부를 저장한다.
Color 객체로 이루어진 1차원 배열 blockColor은 후에 random 함수와 함께 블록의 색깔을 결정한다.
정수형 타입 x와 y는 현재 떨어지고 있는 블록이 위치한 각각의 cell 좌표를 저장한다.
정수형 타입 plusX와 plusY는 현재 떨어지고 있는 블록의 좌표 변화값을 저장한다. 이는 후에 블록이 회전할 때 쓰이는 변수이다.
정수형 타입 rotation은 블록의 회전 값을 나타낸다. 0부터 3까지의 값을 가진다.

```
//block의 모양, rotation, 그때의 cell 위치를 저장하는 배열
public int block[][][]= {
        {
                {0, 4}, {1, 4}, {2, 4}, {3, 4},
                {1, 3}, {1, 4}, {1, 5}, {1, 6},
                {0, 4}, {1, 4}, {2, 4}, {3, 4},
                {1, 3}, {1, 4}, {1, 5}, {1, 6}
        }, // I
        {
                {0, 4}, {0, 5}, {1, 4}, {1, 5},
                {0, 4}, {0, 5}, {1, 4}, {1, 5},
                {0, 4}, {0, 5}, {1, 4}, {1, 5},
                {0, 4}, {0, 5}, {1, 4}, {1, 5}
        }, // O
        {
                {0, 4}, {1, 4}, {2, 4}, {2, 3},
                {1, 3}, {2, 3}, {2, 4}, {2, 5},
                {0, 3}, {0, 4}, {1, 3}, {2, 3},
                {1, 3}, {1, 4}, {1, 5}, {2, 5}
        }, // J


        {
                {0, 4}, {0, 5}, {1, 3}, {1, 4},
                {0, 4}, {1, 4}, {1, 5}, {2, 5},
                {2, 3}, {2, 4}, {1, 4}, {1, 5},
                {0, 4}, {1, 4}, {1, 5}, {2, 5}
        }, // S
        {
                {0, 3}, {0, 4}, {1, 4}, {1, 5},
                {2, 4}, {1, 4}, {1, 5}, {0, 5},
                {1, 3}, {1, 4}, {2, 4}, {2, 5},
                {2, 4}, {1, 4}, {1, 5}, {0, 5}
        }, // Z
        {
                {0, 3}, {0, 4}, {0, 5}, {1, 4},
                {1, 4}, {1, 5}, {0, 5}, {2, 5},
                {2, 3}, {2, 4}, {2, 5}, {1, 4},
                {0, 3}, {1, 3}, {2, 3}, {1, 4}
        }, // T
        {
                {0, 4}, {1, 4}, {2, 4}, {2, 5},
                {1, 3}, {2, 3}, {1, 4}, {1, 5},
                {0, 4}, {1, 5}, {2, 5}, {0, 5},
                {2, 3}, {2, 4}, {2, 5}, {1, 5}
        } // L
};
```

테트리스에서 블록 타입은 7가지이다. 7가지 모양은 각각 알파벳 I, O, J, S, Z, T, L 와 같다. 그리고 블록들이 각각 회전하면 cell의 좌표 값이 변할 것이다. 그래서 정수형 타입의 3차원 배열인 block 멤버 변수는 첫 번째 인덱스로 블록의 타입을 결정한다. 두 번째 인덱스에서 0~3까지는 rotation이 0일 때, 즉 초기 값을 나타낸다. 4~7까지는 rotation이 1일 때, 즉 초기 값에서 시계방향으로 한번 회전했을 때의 cell값을 나타낸다. 8~11까지는 rotation이 2일 때, 즉 초기 값에서 시계방향으로 두 번 회전했을 때의 cell값을 나타낸다. 12~15까지는 rotation이 3일 때, 즉 초기 값에서 시계방향으로 세 번 회전했을 때의 cell값을 나타낸다.

마지막 세 번째 인덱스가 0일때는 첫 번째 인덱스 유형의 블록이 두 번째 인덱스 값만큼 회전했을 때의 각 cell의 y축(세로) 값을 저장하며, 세 번째 인덱스가 1일때는 x축(가로) 값을 저장한다.

```
//게임의 종료 유무를 저장하는 Boolean 타입 변수
public Boolean stop;

//블록이 하나하나 내려올때마다 블록의 색깔과 유형이 랜덤으로 결정됨
public int bColor;
public int bType;
public boolean b;
```

Boolean 타입의 변수 stop은 블록이 위에까지 차서 더 이상 내려올 수 없을 때 게임이 종료된다는 의미에서 true값을 저장한다.
정수형 변수 bColor와 bType은 각각 블록이 하나가 새로 내려올 때마다 블록의 색깔과 유형을 랜덤하게 지정하여 이에 해당하는 인덱스 값을 저장한다.

Tetris class안에 있는 멤버변수와 클래스는 다음과 같다.

```
public void write() {

public void read() {

//맨 아래 한줄이 채워졌을때 아래줄을 삭제하는 함수
public void eraseLine() {

//사용자가 누르는 키보드에 반응
//'a'는 블록이 반시계방향으로 회전
//'b'는 블록이 시계방향으로 회전
//'왼쪽''오른쪽''아래' 화살표에 따라 블록이 이동함
class key extends KeyAdapter{
```

write()함수는 하나의 블록이 착지 할 때마다 각 cell의 상태값을 파일에 저장하는 함수이다.
read()함수는 다음 블록이 내려오기 전에 파일에 있는 각 cell의 상태 값을 읽어 그에 해당하는 색깔로 채워준다.
eraseLine()함수는 맨 아래 한줄이 채워졌을 때 아래 줄을 삭제하는 함수이다.

마지막으로, key class는 사용자가 왼쪽, 오른쪽, 아래 화살표를 누르거나, 알파벳 a 혹는 d를 눌렀을 때 그 값을 받아와서 블록을 이동시키거나 회전시켜주는 기능을 한다.

이러한 멤버변수와 멤버함수, 그리고 클래스를 바탕으로 Tetris class의 생성자가 호출됐을 때 6가지 기능을 구현하는 방법은 다음과 같다.

첫 번째 기능인 '테트리스 실행 GUI을 만든다.'는 다음과 같이 구현했다.
먼저, JFrame의 title을 테트리스 프로그램의 제목인 '안녕, 테트리스!' 로 setting 해주고, setVisible(true)를 해준다.
그리고 20행 10열로 구성된 GridLayout 객체를 생성하여 레이아웃으로 설정해주며, 사이즈도 setting 해준다.
2차원 배열인 jb와 state는 각각 JButton 과 Boolean 타입인 20행 10열의 크기의 배열로 동적 할당해준다.
각각의 cell에는 JButton 객체가 위치하며, JButton 객체는 바탕색을 하얀색으로 초기화해준다. 또한, 각각의 cell의 state 변수는 블록이 아무것도 없으므로 false로 초기화 해준다.

```
//제목
setTitle("안녕, 테트리스!");
//x를 누르면 프로그램 종료
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
//가로 10, 세로20 개의 버튼으로 구성된 레이아웃
//색깔은 화이트로 초기화한다
GridLayout gd = new GridLayout(20,10);
setLayout(gd);
jb = new JButton[20][10];
state= new Boolean[20][10];
setSize(500,1000);
for(int i = 0; i < 20; i++) {
    for(int j = 0; j < 10; j++) {
        jb[i][j] = new JButton();
        add(jb[i][j]);
        jb[i][j].setBackground(Color.WHITE);
        state[i][j]=false;
    }
}
```

두 번째 기능인 '키보드의 입력을 받아 블록을 이동한다.' 는 다음과 같이 구현했다. 먼저, Tetris class 내부에 있는 key class 의 내부 코드에서 KeyEvent 객체를 이용해 입력받은 key값을 저장한다.

```
else if(key == e.VK_RIGHT) {
    move = true;
    for(int i = 0; i < 4; i++) {
        if(x[i]+1>=10) {
            move = false;
        }
        else if(state[y[i]][x[i]+1]==true) {
            move = false;
        }
    }
    if(move==true) {
        for(int i = 0; i <4; i++) {
            jb[y[i]][x[i]].setBackground(Color.WHITE);
            x[i]++;
        }
        plusX++;
        for(int i = 0; i < 4; i++) {
            jb[y[i]][x[i]].setBackground(blockColor[bColor]);
        }
    }
}
```

만약 사용자가 -> 화살표 키보드를 누르면 블록이 오른쪽으로 이동해야한다. 이동시키기 전에 확인해야 할 것은 이동시키면 배열의 범위를 넘어가는지 이다. 따라서 Boolean 타입의 변수 move를 이용해 범위를 벗어난다면 false 값을 저장한다. 블록은 move 값이 true 일 때만 오른쪽으로 이동할 수 있다. 블록을 오른쪽으로 이동시킬 때는 블록의 각 cell의 좌표 값을 저장하는 멤버 변수 x와 y 변수를 이용한다. 해당 블록을 화이트로 초기화 시켜주고, 변하는 좌표 값을 x와 y변수에 넣어준 뒤, 그 좌표에 랜덤 색깔인 bColor를 색칠해준다. 왼쪽, 아래쪽 화살표도 마찬가지이다.

세 번째 기능인 ‘ 맨 아래 블록 한 줄이 채워졌을 때 아래 블록 한 줄은 삭제된다.’ 는 다음과 같이 구현했다.

```
Boolean erase = true;
for(int i =0; i<10;i++) {
    if(state[19][i]==false) {
        erase = false;
    }
}
if(erase==true) {
    eraseLine();
    score+=10;
    strScore = Integer.toString(score);
    score2 = new JMenu(strScore);
}
```

맨 마지막 줄의 cell에서 블록이 존재하지 않는 cell이 없다면, 맨 마지막줄이 채워진 것이므로 eraseLine() 함수를 호출한다.

```
//맨 아래 한줄이 채워졌을때 아래줄을 삭제하는 함수
public void eraseLine() {
    synchronized(this){
        for(int i = 19; i >=0;i--) {
            for(int j=0;j<10;j++) {
                if(i==19) {
                    jb[i][j].setBackground(Color.WHITE);
                }
                else {
                    Color before =jb[i][j].getBackground();
                    jb[i+1][j].setBackground(before);
                    state[i+1][j] = state[i][j];
                }
            }
        }
    }
}
```

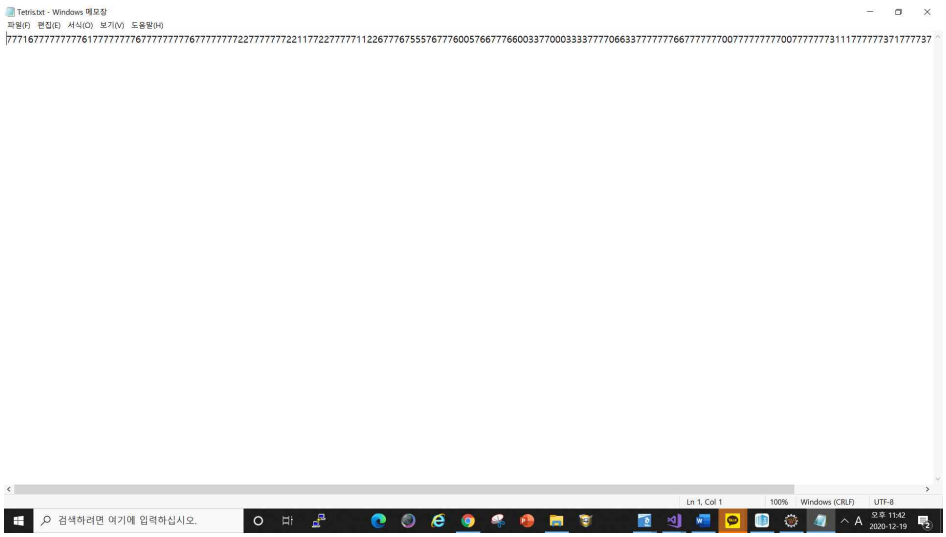
19번째 행을 하얀색으로 초기화 한 뒤, 반복문을 통해 각 행의 색깔이 아래에 그대로 복사되도록 한다.

네 번째 기능인 ‘ 한 블록이 내려와서 착지할 때마다 블록들의 정보를 파일에 기록하고, 다음 블록이 내려오기 전에 파일에서 블록들의 정보를 가져온다.’ 는 다음과 같이 구현했다.

```
public void write() {
    Charset cs = null;
    Path p = null;
    try {
        cs = Charset.defaultCharset();
        p = new File("C:\\homework\\Tetris.txt").toPath();
        if(!Files.notExists(p)) {
            Files.delete(p);
        }
        Files.createFile(p);

        for(int i = 0; i<20;i++) {
            for(int j = 0; j<10;j++) {
                if(jb[i][j].getBackground()==Color.WHITE) {
                    String s = Integer.toString(7);
                    byte data[] = s.getBytes();
                    Files.write(p,data,StandardOpenOption.APPEND);
                }
                else {
                    for(int k = 0; k<7;k++) {
                        if(jb[i][j].getBackground()==blockColor[k]) {
                            String s = Integer.toString(k);
                            byte data[] = s.getBytes();
                            Files.write(p,data,StandardOpenOption.APPEND);
                            break;
                        }
                    }
                }
            }
        }
    }catch(IOException e) {
        e.printStackTrace();
    }
}
```

homework 폴더에 Tetris.txt 파일을 만들고 여기에 각 cell의 색깔에 해당하는 인덱스 숫자를 넣는다. 그러면 프로그램 실행 후 Tetris.txt 파일은 다음과 같다.



마찬가지로 read함수는 Tetris.txt 파일에 있는 각 cell의 색깔에 해당하는 인덱스 숫자를 읽어서 해당 Jb객체에 setBackGround를 해준다.

```
public void read() {
    Charset cs = null;
    Path p = null;
    try {
        cs = Charset.defaultCharset();
        p = new File("C:\\homework\\Tetris.txt").toPath();
        List<String>lines = Files.readAllLines(p,cs);
        int i = 0; int j = 0;
            for(String line: lines) {
                int n = Integer.parseInt(line);
                if(n==7) {
                    jb[i][j].setBackground(Color.WHITE);
                }
                else {
                    jb[i][j].setBackground(blockColor[n]);
                }
                j++;
                if(j==10) {
                    i++;
                    j=0;
                }
            }
    }catch(IOException e) {
        e.printStackTrace();
    }
}
```

다섯 번째 기능인 ‘ 블록의 유형과 색깔은 랜덤으로 지정한다.’ 는 다음과 같이 구현했다. 랜덤함수를 이용하여 블록의 타입과 색깔을 랜덤하게 배정해준다.

```
//블록과 색깔을 랜덤하게 지정하여 초기 블록의 cell위치 지정
bType = random.nextInt(7);
bColor = random.nextInt(7);
for(int i = 0; i <4; i++) {
    y[i] = block[bType][i][0];
    x[i] = block[bType][i][1];
    Color old = blockColor[bColor];
    if(state[y[i]][x[i]]==true) {
        stop = true;
        old = jb[y[i]][x[i]].getBackground();
    }
    jb[y[i]][x[i]].setBackground(old);
}
```

여섯 번째 기능인 ‘ 특정 키보드를 입력했을 때 블록이 회전한다.’ 는 다음과 같이 구현했다.
만약 키보드 a가 눌리면 반시계방향으로 회전하므로 rotation을 1감소시켜주고, 키보드 d가 눌리면 시계방향으로 회전하므로 1을 증가시켜준다..

```
@Override
public void keyPressed(KeyEvent e) {
    // TODO Auto-generated method stub
    int key = e.getKeyCode();
    Boolean move = true;
    if(key == 68) {
        move = true;
        int change = 4*((rotation+1)%4);
        for(int i = 0; i < 4; i++) {
            if(plusY+block[bType][change+i][0]<0||plusY+block[bType][change+i][0]>=20||
                    plusX+block[bType][change+i][1]<0||plusX+block[bType][change+i][1]>9) {
                move = false;
            }
            else if(state[plusY+block[bType][change+i][0]][plusX+block[bType][change+i][1]]==true) {
                move = false;
            }
        }
        if(move==true) {
            rotation = (rotation+1)%4;
            for(int i = 0; i <4; i++) {
                jb[y[i]][x[i]].setBackground(Color.WHITE);
                y[i]=plusY+block[bType][change+i][0];
                x[i]=plusX+block[bType][change+i][1];
            }
            for(int i = 0; i < 4; i++) {
                jb[y[i]][x[i]].setBackground(blockColor[bColor]);
            }
        }
    }
```

다음 코드는 a가 눌렸을 때의 코드이고 d가 눌렸을 때도 같은 방식이다.

일곱 번째 기능인 ‘ 테트리스 종료 조건과 같이, 블록이 위까지 다 차면 게임은 종료된다. 그리고 GUI 창 맨 위의 X버튼을 누르면 프로그램이 종료한다.’ 는 다음과 같이 구현했다. 이 코드를 추가함으로 써 GUI 창 맨 위의 X버튼을 누르면 프로그램이 종료한다.

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

그리고 블록이 위까지 차는 것은 boolean 타입의 stop 변수를 제어하여 구현하였다.

블록은 1초에 한번씩 스레드가 반복하여 한 칸씩 내려온다.