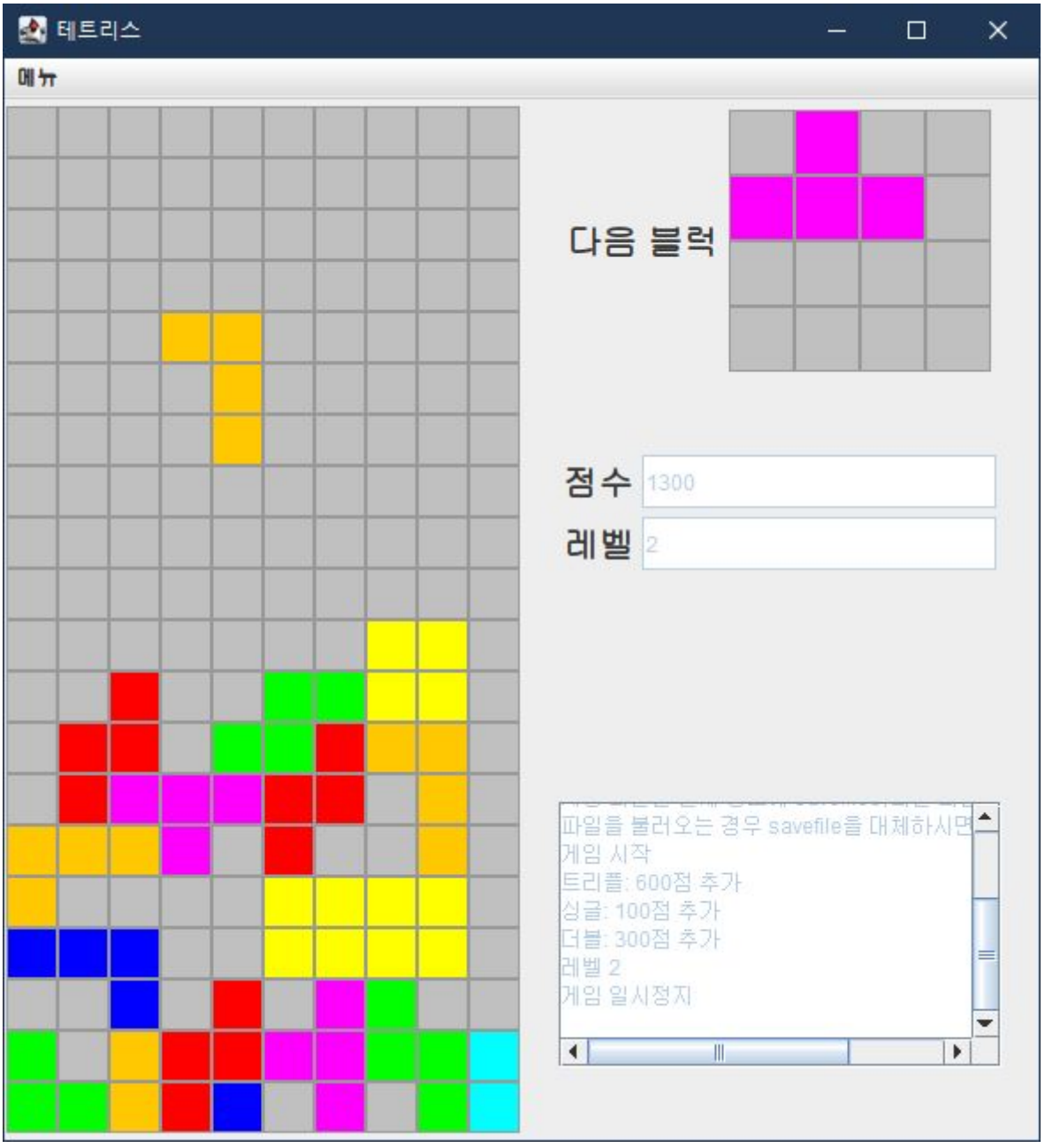
자바 프로그래밍 기말 과제 보고서

12191621 안용모

실행 화면



게임 실행 화면은 다음과 같이 테트리스 게임 화면, 다음 블럭 표시, 점수와 레벨 표시, 그리고 여러가지 정보를 표시하는 텍스트 창으로 구성되어 있다.

점수는 라인을 지우거나 블럭을 놓을 때 마다 추가되며, 레벨은 5줄을 지울 때 마다 늘어나게 된다. 레벨이 늘어날수록 블럭이 떨어지는 속도가 빨라지게 된다.

조작 키는 게임을 실행할 때 텍스트 창에 표시되므로 거기를 참조하도록 한다.

메뉴 버튼을 누르면 다음과 같은 항목이 나타난다.

- 재시작: 게임을 재시작한다.
- 저장: 현재 게임의 상태를 저장한다.
- 불러오기: 저장한 게임의 상태를 불러온다.
- 종료: 프로그램을 종료한다.

Main

프로그램이 시작되는 main() 함수가 존재하는 동시에 JFrame을 상속받아 화면에 띄우는 클래스이다.

생성자

테트리스 게임을 실행할 JFrame의 기본적인 속성을 지정하고 JFrame에 실제 게임 화면을 출력하는 GameScreen이라는 JPanel을 추가하고 메뉴인 MenuBar 클래스를 추가한 뒤, 플레이어의 입력을 관리하는 Player 객체를 생성하고 JFrame의 KeyLister에 추가한다.

그리고 추가로 GameScreen의 logging 함수를 이용해 게임 조작법 등의 여러가지 정보를 출력한다.

public static void main(String[] args)

프로그램이 실행되는 함수이다. 여기에서 새 Tetris 객체를 생성하고 (게임 시작) Main의 생성자를 호출에 UI가 출력되도록 한다.

logic

실제 게임 진행과 관련된 클래스들이 포함된 패키지이다.

Tetris

실제 게임 진행의 대부분을 담당하는 클래스이다. Tetris의 새 인스턴스를 생성할 시 새로운 게임이 실행된다.

변수

public static final int GRID_ROW = 20
public static final int GRID_COLUMN = 10
플레이어에게 보여질 테트리스 grid의 행과 열의 크기를 정하는 상수이다.
public int[][] grid
테트리스 게임이 이루어지는 2차원 행렬이다. 빈 공간은 0, 고정된 블럭이 존재하는 공간은 0보다 작은 수, 현재 움직이는 블럭이 존재하는 공간은 0보다 큰 수로 나타내며 절댓값은 블럭의 색깔을 결정하게 된다.

주의점은 grid를 초기화할 때 grid의 행은 GRID_ROW + 5개 존재하고, grid의 열은 GRID_COLUMN + 2개 존재하도록 초기화되는데, 이는 다른 처리 없이 보이지 않는 공간에서의 블럭의 생성과 배치, 그리고 테트리스가 좌우나 아래로 움직일 때 보이지 않는 벽에 막히도록 구현할 때 별도의 처리 없이 구현하도록 하기 위함이다.
즉 grid는 처음에 다음과 같은 형태로 초기화되게 된다.

```
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0  0  0  0 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

(주황색 박스) 블록 생성을 위한 공간
(파란색 박스) 플레이어에게 보이는 공간
-1 테트리스 이동 제한을 위한 공간

public int pieceCycle
테트리스 조각이 몇번째 조각을 순회하는지 나타내는 변수이다.
public ArrayList<Integer> pieceArr
테트리스 조각이 순회할 순서를 결정하는 변수이다. {0, 1, 2, 3, 4, 5, 6}으로 초기화되며, pieceCycle이 한번의 순회를 마칠 때 (pieceCycle == 7일 때)마다 셔플하고 pieceCycle의 값을 0으로 초기화해 7-bag role (7번 안에는 모든 블럭의 종류가 나타나는 규칙)을 구현한다.
public Piece curPiece
public Piece nextPiece
각각 현재 떨어지고 있는 테트리스 블럭의 정보와 다음에 떨어질 블럭의 정보를 나타내는 Piece의 인스턴스 변수이다.
public int score
public int level
public int dropInterval
public int deletedLines
각각 현재 테트리스 게임에서의 정보, 레벨, 블럭이 떨어지는 주기 (ms)와 레벨을 갱신하는데 사용되는 현재까지 블럭을 몇개 지웠는지 확인하는 변수이다.
public boolean isGamePaused
public boolean isGameOver
각각 현재 게임이 정지된 상태인지, 혹은 종료된 상태인지 확인하는 변수이다.
게임이 끝난 상태에서는 모든 키 리스너가 동작하지 않는다.

게임이 정지된 상태에서는 스레드에서 update()가 실행되지 않으며, Player 클래스에서 게임 정지/시작 버튼에 대한 키 리스너만 동작하게 된다.

private Thread gameThread
테트리스에서 주기적으로, 즉 (dropInterval)ms마다 update()를 호출하는 동작을 수행하는 스레드이다. 만약 dropCheck()에서 게임 오버인 상태로 만든 경우에는 이 스레드는 interrupt된다.

생성자

grid 배열을 초기화하고 grid 배열의 주소를 Piece의 curGrid에 대입시켜주고, pieceArr을 초기화해준 뒤 curPiece와 nextPiece를 두번의 getPiece() 호출을 통해 초기화해준다. 이때 초기 pieceCycle의 값은 7이므로, 처음 7개 블럭의 순서는 랜덤함이 보장된다. 그리고 curPiece를 grid에 표현해주고 (curPiece.drawShape()) GameScreen의 render()를 호출한다.

그리고 다른 클래스의 static 변수로 (Player.tetris, SaveFile.tetris, MenuBar.tetris) 현재 인스턴스의 주소(this)를 대입해 준 뒤, gameThread를 생성한 뒤 start()한다.

private Piece getPiece()

다음 블럭이 될 블럭, 즉 nextPiece가 될 Piece를 얻는다. 이때 테트리스의 7-bag role (7번 안에는 모든 블럭의 종류가 나타나는 규칙)을 만족하기 위해 pieceCycle == 7인 경우 pieceArr을 shuffle해 앞으로 나올 7개의 블럭의 순서를 정해주고 pieceCycle = 0으로 한 뒤 pieceArr[pieceCycle]에 맞는 블럭을 return한다. 이때 return할 때마다 pieceCycle의 값을 더해준다.

private void update()

gameThread에서 (dropInterval)ms마다 호출하는 함수이다. 게임오버가 아닌 경우 현재 블럭을 한칸씩 내리며, 만약 블럭을 더 내릴 수 없는 경우 score에 10을 더하고, 현재 블럭을 고정한 뒤, dropCheck()를 호출하고 curPiece에 nextPiece를 대입하여 다음 블럭이 내려오도록 한 뒤, nextPiece를 getPiece()를 호출한 값으로 정하여 다음 블럭을 정한다.

그리고 현재 블럭을 한칸씩 내린 이후 주기적으로 GameScreen의 render()를 호출한다.

private void dropCheck()

블럭을 고정한 시점에서 호출되는 함수로서, 다음과 같은 기능을 수행한다.

1. 꽉 찬 열을 지우고 grid 갱신
   블럭을 고정했을 때 지워져야 하는 열이 존재하는 경우 다음과 같은 과정을 통해 grid를 갱신한다.
   - 꽉 찬 열의 번호를 담는 큐 (fullLines)를 만든다.
   - 높은 번호의 열부터 그 열이 꽉 찼는지 확인한다. 만약 꽉찬 열이라면 fullLines에 그 번호를 enqueue한다.
   - 꽉찬 열이 아닌 경우, fullLines에 값이 존재한다면 가장 앞의 번호의 열에 현재 번호의 열의 값을 복사한다. 그리고 현재 열의 번호를 fullLines에 삽입한다.
   - 이를 확인해야 할 가장 작은 grid의 열(4열)까지 반복하여 grid를 갱신한다.

2. 점수 추가
   fullLines의 크기가 현재 지워질 열의 개수일 때, 그 값을 이용해 점수를 더해준다.
   - 싱글 (한 라인을 지움): 100점 추가
   - 더블 (두 라인을 지움): 300점 추가
   - 트리플 (세 라인을 지움): 600점 추가
   - 테트리스 (네 라인을 지움): 1000점 추가

3. 레벨 업데이트
   지워진 라인의 누적 개수(deletedLines)를 fullLines의 크기만큼 추가하고, 만약 deletedLines가 5보다 크거나 같은 경우 레벨을 1 올린다.
   이때 레벨에 맞게 떨어지는 속도 (dropInterval) 역시 수정한다.

4. 게임 오버 확인
   보이지 않는 grid 영역(0 ~ 3열)에 고정된 블럭이 존재하는 경우 게임 오버이므로, 주기적으로 update()해주는 gameThread를 interrupt해주고 isGameOver를 true로 한다.

Piece

테트리스에서 사용되는 블럭의 정보를 나타나는 클래스이다.

변수

public static int[][] curGrid
Piece가 실제로 존재하는 테트리스의 grid이다. 새 Tetris 인스턴스가 만들어질 때 curGrid는 그 인스턴스의 grid를 가리키도록 초기화된다.

public int kind
블럭의 종류를 나타내는 변수이다. 게임 저장/불러오기 때 사용된다.
public int shapeSize
블럭의 모양과 색깔을 나타내는 shape 2차원 배열의 크기를 나타내는 변수이다.
블럭마다 필요로 하는 최소 n*n 공간이 다르기 때문에 필요한 변수이다.
즉, shape는 shapeSize * shapeSize 크기의 배열로 초기화된다.
public int[][] shape
블럭의 모양과 색깔을 나타내는 2차원 배열이다.
값의 의미는 Tetris의 grid 배열과 동일하다.
public int curRow
public int curCol
현재 Piece가 grid에서 실제로 존재하는 행/열을 나타내는 변수이다.
현재 움직이고 있는 블럭의 실제 위치는 shape[i][j]이 나타내는 위치가 grid[curRow + i][curCol + j]와 동일하므로, shape[i][j] > 0인 위치에 대응되는 grid[curRow + i][curCol + j]이다.

생성자

생성자에 주어진 kind 변수에 따라 shape, shapeSize, kinds 변수를 초기화하고, rotateLeft()와 right()를 랜덤한 횟수만큼 호출하여 블럭이 생성될 때 회전 방향과 위치를 랜덤하게 정하도록 한다.

private boolean checkShape(int[][] newShape, int newRow, int newCol)

바뀐 블럭의 모양인 newShape, 바뀐 블럭의 위치인 newRow와 newCol을 인수로 받아 실제로 가능한 배치인지 확인하고, 맞다면 curGrid를 갱신한다. 주어진 움직임이 가능한가를 true/false로 return한다.

블럭이 새로운 위치에 있을 수 있는지 확인하는 방법은 newShape와 newRow, newCol에 따른 현재 위치 (newShape[i][j] != 0일 때 curGrid[newRow + i][newCol + j])에 고정된 블럭이 존재한다면 불가능한 이동이고, 모든 블럭을 확인해도 그러한 블럭이 없다면 가능한 움직임이다. (newShape[i][j] != 0 && curGrid[newRow + i][newCol + j] < 0 인 i, j가 존재하지 않아야 한다.)

갱신은 현재 위치의 블럭들을 모두 빈 공간 (0)으로 만들어주고, 새로운 위치에 대해 curGrid의 값을 newShape의 값과 같게 맞추면 된다.

public boolean left() / public boolean right()

블럭을 각각 좌우로 움직인다. 변한 블럭의 위치는 각각 curCol - 1/curCol + 1이다.
left()는 checkShape(shape, curRow, curCol - 1)와 같다.
right()는 checkShape(shape, curRow, curCol + 1)와 같다.

public boolean drop() / public void hardDrop()

drop()은 블럭을 아래로 움직인다. 변한 블럭의 위치는 drop()의 경우 curRow + 1이다.
즉 drop()은 checkShape(shape, curRow + 1, curCol)이다.
이때 harddrop()은 곧바로 블럭을 바닥으로 떨어지게 하는 경우, 즉 가능한 한 계속 drop()하는 경우이므로, 간단하게 drop()이 true인 동안 계속 drop()하도록 while문을 작성하여 구현할 수 있다.

public boolean rotateLeft() / public boolean rotateRight()

블럭을 반시계 방향이나 시계방향으로 회전시킨다.
블럭의 새 모양을 newShape라 할 때, rotateLeft()와 rotateRight() 모두 checkShape(rotated, curRow, curCol)와 같으며, newShape의 형태만 다르다.

rotateLeft()에서의 newShape는 shape 행렬에 대한 전치 행렬에서 행의 순서를 뒤집은 것과 같고, rotateRight()에서의 newShape는 shape 행렬에 대한 전치 행렬에서 열의 순서를 뒤집은 것과 같다.

public boolean drawShape()

현재 위치에 관해 curGrid의 값을 갱신한다. Tetris의 생성자에서 처음 블럭을 만드는 경우에만 쓰인다.

public void fixed()

호출한 Piece를 현재 위치에 고정시켜 움직이지 않게 한다.
즉, piece의 curGrid에서의 값을 전부 음수로 바꾸어준다.

Player

플레이어와 게임과의 상호작용, 즉 게임에 대한 플레이어의 입력을 관리하는 KeyListener이다.

public static Tetris tetris
게임을 조작할 때 사용할 Tetris 객체의 주소이다. Tetris 객체를 생성할 때 초기화된다.

public void keyPressed(KeyEvent e) - @Override

게임 오버된 경우 (tetris.isGameOver가 true인 경우)에는 모든 입력이 무시된다.
게임이 정지된 상태에서는 (tetris.isGamePaused가 true인 상태) 'P'키에 대해서만 작동한다.

- LEFT/RIGHT/DOWN키
  테트리스에서 현재 움직이는 블럭을 좌/우로 옮기거나 내린다.
  (tetris.curPiece.left(), tetris.curPiece.right(), tetris.curPiece.drop())
- A/D키
  테트리스에서 현재 움직이는 블럭을 반시계방향/시계방향으로 회전시킨다.
  (tetris.curPiece.rotateLeft(), tetris.curPiece.rotateRight())
- SPACE키
  현재 블럭에 대해 hard drop(떨어질 위치로 바로 떨어짐)을 수행한다.
  (tetris.curPiece.hardDrop())
- P키
  게임의 정지 상태를 토글한다.
  (tetris.isGamePaused = !tetris.isGamePaused)

ui

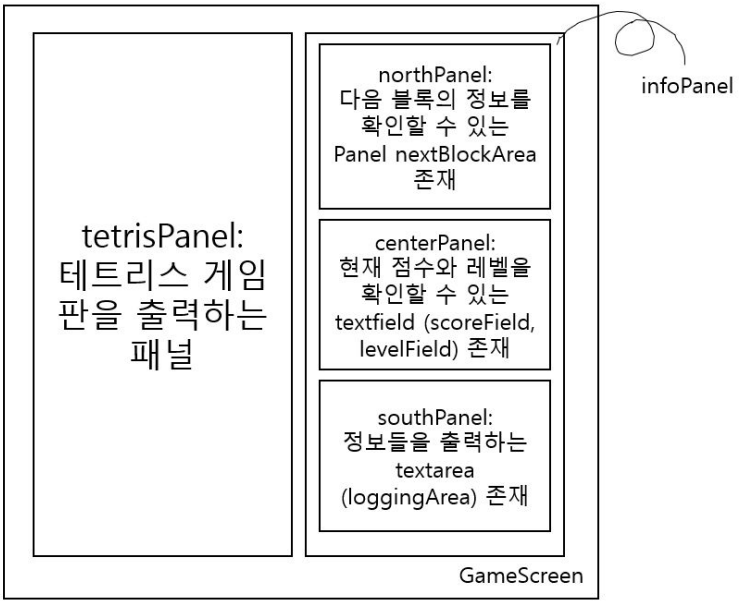
프로그램의 GUI와 관련된 클래스가 모인 패키지이다.
이 패키지 내의 클래스들은 프로그램 실행 언제나 인스턴스가 하나 존재해야 하기 때문에, 싱글턴 기법을 이용해 클래스들을 생성하였다.

싱글턴 기법: 특정 클래스의 인스턴스가 프로그램을 실행하고 오직 하나만 생성하고 실행되도록 보장하는 기법이다.

GameScreen

실제 게임을 실행하는 화면의 정보가 담긴 JPanel 객체이다.

변수



private Color[] colorset
tetrisPanel과 nextBlockArea에 표시할 블럭의 색깔을 정하는 배열이다.
private JPanel tetrisPanel
테트리스 게임판을 출력하는 패널이다. Tetris의 GRID_ROW * GRID_COLUMN과 같은 수의 Jbutton을 가지고 GridLayout을 따른다..
private JPanel infoPanel
테트리스 게임의 정보를 출력하는 패널이다.
private JPanel nextBlockArea
다음 블럭의 정보를 보여주는 패널이다. 4*4 Jbutton 행렬이 존재하는 GridLayout을 따른다.
private JButton[][] nextBlock
nextBlockArea에서 표시되는 JButton 행렬이다.
private JTextField scoreField
현재 점수를 보여주는 패널이다.
private JTextField levelField
현재 레벨을 보여주는 패널이다.
private JTextArea loggingArea
게임의 정보를 출력하는 textarea이다.

생성자

패널을 1*2 GridLayout으로 초기화하고, setTetrisPanel(), setInfoPanel()을 호출한다.

private void setTetrisPanel()

tetrisPanel을 구성하고 GameScreen에 add한다.

private void setInfoPanel()

infoPanel을 구성하고 GameScreen에 add한다.

public void render(Tetris tetris)

현재 존재하는 Tetris 객체의 정보에 따라 tetrisPanel, nextBlockArea, scoreField, levelField의 정보를 갱신한다. 이때 테트리스의 색깔은 grid나 shape 배열의 값의 절대값으로 정해진다.

public void logging(String str)

loggingArea에 str을 append하여 출력하고 줄바꿈한다.
이때 스크롤바의 위치는 문서의 끝으로 한다.

MenuBar

프로그램의 MenuBar 클래스이다.

변수

public static Tetris tetris
게임을 중단하거나 게임이 종료되었는지 확인할 때 사용할 Tetris 객체의 주소이다. Tetris 객체를 생성할 때 초기화된다.
private ActionListener restartAction
메뉴의 “재시작” 버튼을 누를 때 실행되는 함수이다. 게임을 정지하고 새 Tetris 인스턴스를 만들어 새 게임을 실행한다.
private ActionListener saveAction
메뉴의 “저장” 버튼을 누를 때 실행되는 함수이다. SaveFile.save()를 호출한다.
private ActionListener loadAction
메뉴의 “불러오기” 버튼을 누를 때 실행되는 함수이다. SaveFile.load()를 호출한다.
private ActionListener quitAction
메뉴의 “종료” 버튼을 누를 때 실행되는 함수이다. 게임을 정지 상태로 한 뒤 종료한다.

생성자

MenuBar에 주어진 메뉴와 맞도록 JMenu와 JMenuItem을 추가한다.

file

파일 입출력을 관리하는 클래스들이 저장된 패키지이다. 여기에는 현재 테트리스 게임의 상태를 저장하고 불러올 수 있게 하는 SaveFile 클래스가 존재한다.

SaveFile

현재 진행중인 Tetris 객체의 상태를 파일 형태로 저장하고 불러올 수 있게 해주는 함수들을 모은 파일이다.

변수

public static Tetris tetris
정보를 저장할 테트리스 객체의 주소이다. Tetris 객체를 생성할 때 초기화된다.
private final static String filename = “savefIle”
정보를 저장하고 불러올 파일의 이름이다. 경로는 실행 파일의 경로와 동일하다.

save()와 load()에서는 현재 tetris 객체에 대해 다음과 같은 정보를 저장/불러오기한다.

- 현재 grid의 모든 값들
- 블럭의 순서를 관리하는 pieceCycle과 pieceArr
- 현재 블럭과 다음 블럭의 정보: curPiece, nextPiece
- 기타 정보들: score, level, dropInterval, deletedLines

public static void save()

위의 모든 정보를 int 형태로 저장 파일에 FileWriter를 이용해 텍스트 파일로 출력한다. 텍스트 파일에 저장되는 순서는 위 정보의 순서와 같다. 이때 Scanner를 위한 load를 쉽게 하기 위해 값 사이사이에 whitespace를 넣어주었다. 게임이 종료된 상태에서는 실행되지 않는다.

public static void load()

위의 모든 정보를 int 형태로 저장한 파일의 정보를 savefile에 대한 Scanner를 통해 읽어들인다. 읽을 때 nextint()를 이용하기 때문에 save()된 정보의 순서와 동일하게 정보를 읽는다. 만약 파일이 존재하지 않으면 오류가 발생했음을 알리는 메시지를 텍스트 창에 출력한다.