**<테트리스>**

**12171588 김민석**

**목적** : 이 과제의 목표는 테트리스 게임을 구현하는 것입니다. 보고서의 순서는 클래스 구조, 소스코드의 전역변수들, 프로그램 전체 흐름, 이후에 기능 구현 부분과 느낀 점 순으로 기술하겠습니다. 구현한 테트리스의 블록들은 다음과 같습니다.

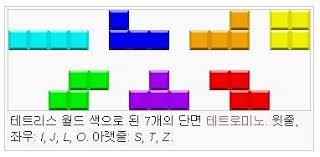
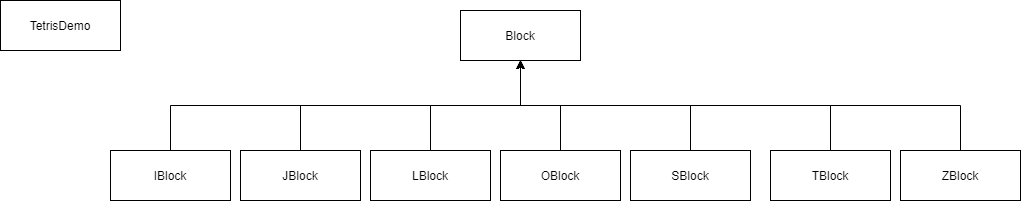


그림 1윗줄 : I, L, J, K / 아랫줄 : S T Z

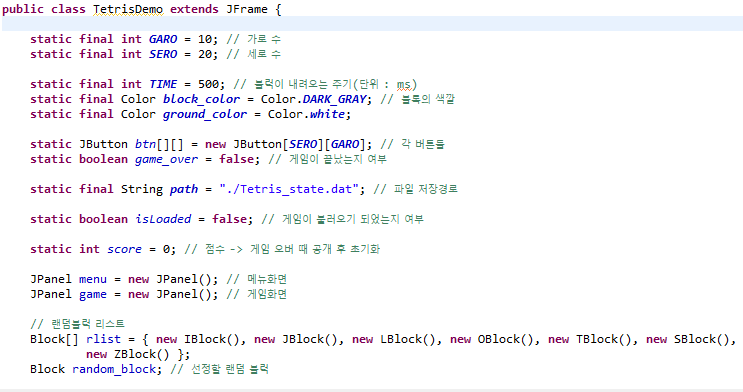
1. **클래스 구조**

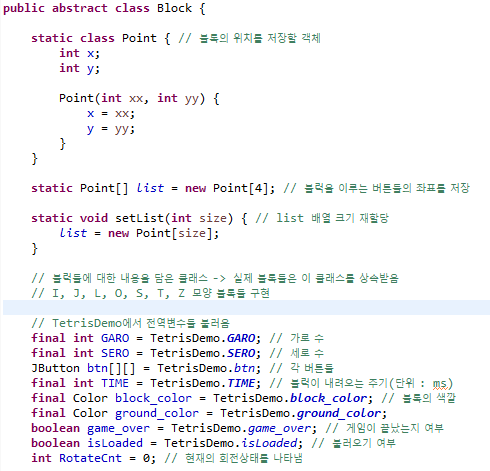


위 그림은 클래스 구조를 나타낸 그림입니다. TetrisDemo.java에서 main함수가 실행되고, Block의 자식들에 다양한 테트리스 블록들의 모양이 구현되어 있습니다.

Block 클래스에서는 TetrisDemo의 여러 변수들을 가져와서 사용하는데, 그럼에도 상속관계가 아닌 이유는 TetrisDemo의 생성자에서 UI화면구성을 진행했기 때문입니다. 즉, 블록은 반복적으로 생겼다 사라지므로 그 때마다 부모의 생성자를 호출하면 안되는 상황이었습니다.

1. **TetrisDemo.java / Block.java의 전역변수들**





TetrisDemo의 대부분의 전역변수를 Block에 불러온 것을 확인할 수 있습니다.

특히, TetrisDemo에는 여러 모양의 블록 형태를 저장하는 rlist가 있으며, 이를 통해 블록을 랜덤으로 배정합니다.

또한 Block에는 자신의 블록 위치(4개의 버튼을 차지)를 저장하는 list라는 변수가 존재합니다. 이 list 위치들은 Block의 자식들에 의해 결정되고, 그에 따라 makeBlock(), Rotate()함수가 다르게 작동합니다.

1. **프로그램 전체 흐름(TetrisDemo)**



위 그림은 테트리스 프로그램의 전반적인 흐름입니다. 처음에 테트리스의 기능을 설명하는 메시지를 띄우고, 생성자에서 UI와 KeyListener들을 구성합니다. 이후에는 키 입력을 받고, 이들을 처리하면서 게임이 진행됩니다.

UI구성에서 makeUI()함수를 만들어서 사용했고, 리스너 구현 부분에서 MyListener객체를 생성해서 사용했습니다.

또한, 추가적으로 점수 기능을 구현했습니다. 저장 및 불러오기 시에 점수도 같이 처리되고, 게임오버로 종료되었을 시 점수를 같이 출력하도록 코드를 작성했습니다.

1. **기능 구현**

* 게임 시작

Enter키를 누르면 게임을 시작합니다. 이 키를 누를 때마다 항상 새로운 게임이 시작되며, 이전의 상황들은 초기화됩니다. 새로운 시작을 위해 gameQuit()함수를 호출한 후, 관련 변수를 초기화하고 gameStart()함수를 실행했습니다. gameStart()에서 전반적인 게임 한 판을 진행하게 됩니다. 불러오기 여부를 판단하고, gamePlay 스레드를 통해 playing()을 실행합니다. playing()에서 gameover 이전까지 랜덤블록을 계속 만들어내고, 처리합니다.

blockPlay() 부분에서 블록 하나가 생성되고 고정되기 까지의 루틴을 구현했습니다. 아래 blockPlay()의 로직 그림과 같이 하나의 랜덤블록이 생성되고 고정됩니다. 게임이 계속 진행되다가, 블록이 천장에 닿았을 경우 점수와 함께 game over가 표시되면서 게임 한 판이 종료됩니다.



blockPlay() 순서도 – 하나의 블록 처리

* 블록 이동 : 왼쪽, 오른쪽, 아래 방향키 입력 시 블록이 해당 방향으로 한 칸 이동합니다. Block.java에서 구현했으며, 모든 모양의 블록이 동일한 동작을 수행합니다.

각 이동방향으로 버튼을 이동시킨 후 다시 색칠하되, 색칠과정에서 바닥에 고정되었거나 범위를 벗어난 경우 등의 예외를 처리하면서 블록 이동을 구현했습니다.

* 블록 한 줄 삭제 : 블록 하나가 바닥에 도착할 때마다 다 채워진 줄이 있는지 확인하고, 삭제까지 진행하는 lineCheck()함수를 작성했습니다. 이중 for문을 통해 화면의 모든 줄을 탐색하면서, 한 줄이 모두 색칠 된 경우에는 그 줄을 제거하는 removeLine(int j)함수를 실행함으로써 줄 삭제를 구현했습니다.
* 빠른 블록 착지 : Space bar키 입력을 통해 블록이 빠르게 바닥에 도착하도록 구현했습니다. 블록이 바닥에 닿았는지 여부를 확인하면서 계속 blockDown()을 진행한 후, 상황에 따라 줄 삭제를 진행하는 방식으로 동작합니다.
* 블록 회전 : R키를 누르면 블록이 반시계 방향으로 90도 회전합니다.

회전 부분은 각 블록마다 자신의 위치를 가리키는 list와, 현재의 회전 상태를 나타내는 RotateCnt, cases를 활용해서 구현했습니다. 블록의 종류에 따라 회전의 경우의 수가 다를 수 있기 때문에, 각각 해당하는 블록 클래스로 오버라이딩 해서 Rotate함수를 작성했습니다.

현재의 회전상태와, list에 있는 위치를 기반으로 버튼의 각 위치를 90도 회전한 곳의 위치로 이동시켰고, 이후에 rotateDraw()함수를 통해 회전된 도형을 화면에 나타냈습니다. 단, 범위를 벗어나거나 이미 색칠되어 있는 블록으로 인해 회전이 불가능한 경우 등에는 예외처리를 해서 회전이 되지 않도록 작성했습니다.

* 저장 및 불러오기 : S키를 통해 현재상태를 저장하고, O키를 통해 저장된 상태를 불러오도록 구현했으며, 이 과정에서 파일 입출력을 이용했습니다. 각 키를 입력했을 때, 저장/불러오기가 완료되었다는 메시지를 띄웁니다. 이후 저장의 경우에는 현재 상태를 초기화하고, 불러오기의 경우에는 메시지를 없앤 뒤 3초 후에 불러온 게임이 진행됩니다.

먼저, 저장의 경우 상대경로 "./Tetris\_state.dat"에 현재의 상태를 저장합니다. fileSave(path)를 통해 현재 버튼의 색칠상태, list의 x,y값들, 현재 진행중인 블록의 회전상태, 현재 점수, 현재 블록의 종류 등을 특정 형식에 맞게 저장합니다. 그리고 저장메시지를 띄우고 게임을 초기화 시킵니다. 불러오기의 경우, 진행중인 게임을 초기화 시킨 후에 저장된 형식에 맞게 파일을 읽으면서 각각 해당하는 변수에 저장합니다. 이후 메시지를 띄우고 3초후에 gameStart()를 통해 게임을 진행합니다.

* 현재 게임 종료 : Q키를 누를 경우 현재 게임이 종료(초기화)됩니다.

gameQuit()함수에서 gamePlay스레드를 interrupt시키고, 각 버튼들의 색깔을 초기화 시킴으로써 현재 게임 종료를 구현했습니다. 위의 여러 이벤트 들에서 게임을 초기화할 때 gameQuit()를 이용했습니다.

* 프로그램 종료 : ESC키를 누르거나 게임 창의 X버튼을 클릭하면 프로그램이 정상종료 됩니다. 프로그램 종료부분을 quit()함수를 통해 따로 작성했으며, setVisible(false); System.exit(0);을 수행합니다. 기존에 makeUI()함수에서 setDefaultCloseOperation(***EXIT\_ON\_CLOSE***)를 작성했으므로 창이 종료되면 프로세스까지 정상적으로 종료됩니다.

기능구현에 활용된 여러 함수들은 코드의 주석을 통해 어떤 동작을 수행하는지 확인할 수 있습니다.

**5. 느낀 점 :** 이 과제를 통해 각 부분의 구현은 쉬울 수 있어도, 그 부분들을 통합하여 관리하는 것이 어렵다는 것을 느꼈습니다. 알고리즘이나 수행시간을 단축시키는 부분도 중요하지만, 이 과제를 하면서 전체적인 프로그램의 흐름이나 순서도 등을 잘 작성하는 것이 중요하다는 사실을 깨달았습니다.