**JAVA** TERM-PROJECT

**Result Report**

|  |
| --- |
| **THE GENIUS-Prototype** |
|  |
| **TEAM : 튜브탄 펭귄** |
| **컴퓨터공학과 201636028 이선민**  **컴퓨터공학과201635949 김채윤** |
|  |

# 튜브탄 펭귄이란?

팀 멤버들 각자가 닮은 캐릭터 이름을 활용하니 ‘튜브탄 펭귄’ 이라는 독특한 팀 이름이 되었습니다.  
카카오프렌즈 중에 ‘튜브’라는 캐릭터는 오리를 닮은 것이 특징 이선민 학생이 학우들 사이에 ‘튜브’라고 불리는 것에 착안을 했습니다. 또한 김채윤 학생은 니니즈 캐릭터 중 ‘케로 & 베로니’ 의 동물 이름을 따서 ‘펭귄’ 으로 설정했습니다.

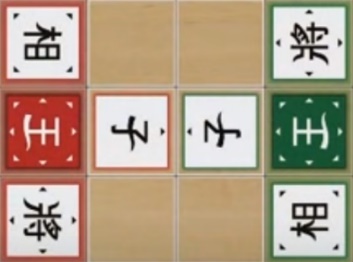
이선민 학생은 팀 내에서 팀장과 프로젝트 방향성 제시, UI 디자인, 코딩, 보고서 작성을 맡고 있습니다. 김채윤 학생은 팀 내에서 UI 디자인, 코딩, 보고서 작성, 보완점 채택을 맡고 있습니다.

# 프로젝트 제목과 이유

저희 프로젝트의 제목은 ***‘The Genius – prototype’*** 입니다. prototype이란 개발 중인 기기, 프로그램, 시스템 등의 성능 검증 및 개선을 위하여 상품화에 앞서 제작하는 시제품을 뜻합니다. 더 지니어스는 메인 매치와 데스 매치로 구성이 되는 게임을 하며 매 회마다 탈락자가 발생하는 규칙을 가지고 있는 게임쇼입니다. 우리가 프로젝트에서 구현하게 될 것은 더 지니어스 속 데스 매치를 위주로 구성할 계획이어서, 더 지니어스의 완성형이 아닌 듯한 생각이 들었고, 나중에는 더 나아가 데스 매치 뿐 아니라 메인 매치의 게임을 만들고 싶은 마음에서 들어서 prototype이라는 부제를 붙였습니다

또한 앞으로의 본격적인 졸업 프로젝트나 큰 프로젝트를 준비하기 전에 처음으로 제작하는 이 java 프로젝트를 통해 우리가 보완해 나갈 점을 찾아 나간다는 의미에서도 prototype이라고 할 수 있습니다.

### 최종적으로 결과물로 구현한 게임 설명 - 십이장기

일반 장기처럼 상대의 왕을 잡으면 승리하는 게임입니다. 가로 4칸, 세로 3칸 총 12칸으로 이루어진 장기판에서 진행되며 플레이어의 바로 앞쪽 3칸이 자신의 진영이 됩니다. 플레이어에게는 4가지의 말이 주어지며 각 말은 지정된 위치에 배치된 상태로 게임이 시작됩니다. 각 말들은 말에 표시된 방향으로만 진행할 수 있으며, 게임판의 사진은 다음과 같습니다.



장 : 자신의 진영 오른쪽에 놓이는 말로 앞, 뒤와 좌우로 이동이 가능합니다.



상 : 자신의 진영 왼쪽에 놓이는 말로 대각선 네 방향으로 이동할 수 있습니다.

왕 : 자신의 진영 중앙에 높이는 말로 앞, 뒤, 좌우, 대각선 방향까지 모두 이동 가능합니다.



자 : 왕의 앞에 놓이며 오로지 앞으로만 이동할 수 있습니다. 만약 상대의 진영에 진입하게 되면 말을 뒤집어 ‘후’ 로 사용할 수 있습니다. ‘후’ 는 대각선 뒤쪽 방향을 제외한 모든 방향으로 이동할 수 있습니다.

게임이 시작되면 선 플레이어부터 말 한 개를 한 칸 이동 가능할 수 있습니다 . 각 턴 당 플레이어가 자신의 말을 옮길 수 있는 시간은 30초입니다. 게임은 한 플레이어가 상대방의 ‘왕’ 을 잡으면 해당 플레이어의 승리로 끝납니다. 하지만 상대방의 ‘말’ 을 잡는 방법 외에 승리 방법 한 가지가 더 있습니다. 자신의 ‘왕’ 이 상대방의 진영에 들어가 자신의 턴이 돌아올 때까지 버티면 해당 플레이어의 승리로 끝납니다.

# 최종 목표를 이루기 위해 한 노력

계획서에 적은 내용들과는 달리 구현에 한계를 느꼈습니다. 한 가지를 구현에 너무나도 많은 구현방식이 존재한다는 것과 더불어 실력의 부재를 뼈저리게 알 수 있었습니다. 그 과정을 통해 나온 최종 결과물을 노력 과정 마지막에 적었습니다.

또한, 과거에 작성한 계획서를 보면 저희 팀은 UI제작에 모든 힘을 쏟겠다고 말씀드렸었습니다. 따라서 이번 최종 보고서에는 수많았던 저희 팀의 실패과정과 더불어 UI에 들어갈 모든 버튼, 게임배경 등 자체 제작한 것까지 모두 담았음을 알려드립니다.

1) 첫 번째 실패 (게임 구현 내용 정리 중 기능 포기) – 11월 27일  
계획서에 나온 게임 순으로 구현 방식을 하나하나 작성했다.  
서버 제작과 더불어 포커의 수많은 가짓수 종류에 짓눌려 일반 포커 게임과 서버, 1인 플레이를 포기했다. 포커 게임 카드는 40장인데, 풀하우스를 만들 수 있는 종류 수를 비롯해 모든 걸 결과창에 먼저 저장해야 플레이어가 조합하는 카드를 정확한 결과와 함께 화면에 출력할 수 있었기 때문이다.

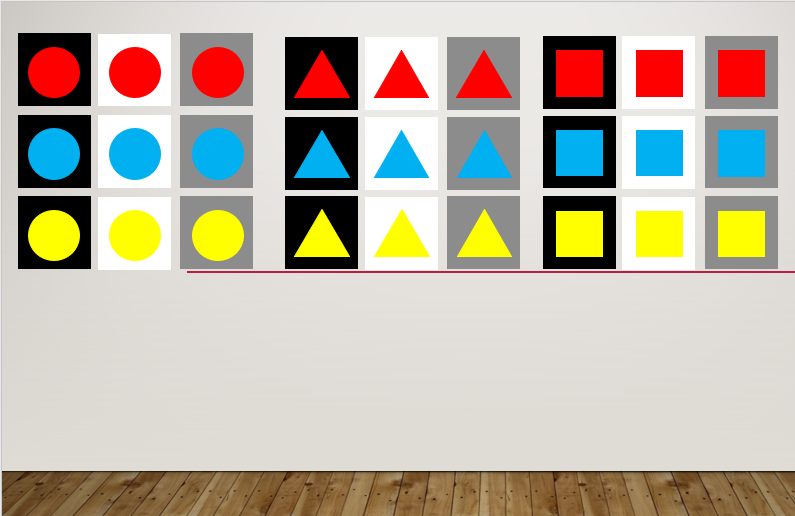
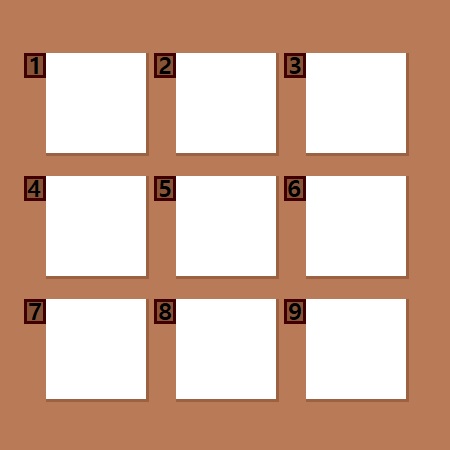
2) 두 번째 실패 (게임 구현 디자인 제작 중 기능 포기) – 12월 1일  
계획서에 나온 게임 중 남은 것은 결합 / 십이 장기 / 인디언 포커 / 양면 포커 였지만 소수의 게임을 또 포기하게 되었다.

인디언 포커에서 ‘자기 자신만 자신의 카드를 볼 수 없다’ 는 기능에서 UI기능과 같이 표현하는 방법을 찾을 수 없어서 포기했다. 게임 화면에서는 상대방이 내 카드를 볼 수 있지만(카드가 공개되어야 함), 내 카드를 내가 볼 수 없다면(카드가 엎어져 있어야 함) 어떻게 해야 하는지 알 수 없었다.  
 서버를 통해서는 구현이 가능하지만 서버를 포기한 시점에서는 불가능하다고 판단하여 인디언 포커 또한 포기하게 되었다.

양면 포커에서도 카드는 뒤집어져 있는 상태에서 내 카드를 나만 확인하는 방법을 찾을 수 없어서 포기했다. 위와 같은 문제점이 있기 때문이다.

회의를 통해 최종적으로 결정된 결합 / 십이장기 중 결합의 게임판과 필요한 것을 제작하게 된다.

이 날 완성한 UI 이미지 :



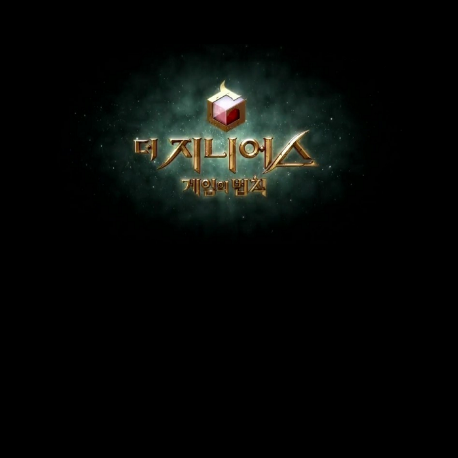
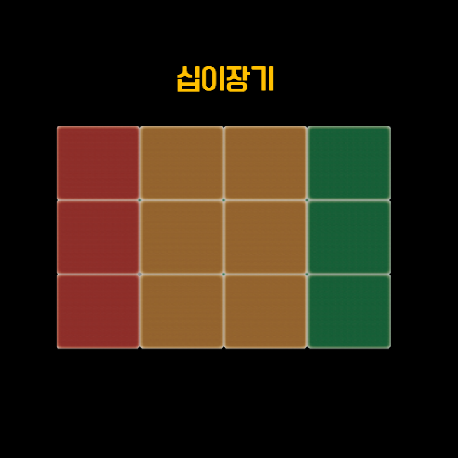
[그림 1 : 결합 장기판 말 종류(27종)] [그림 2 : 결합 게임판]

3) 세 번째 실패 (게임 구현 디자인 제작 중 기능 포기) – 12월 4일

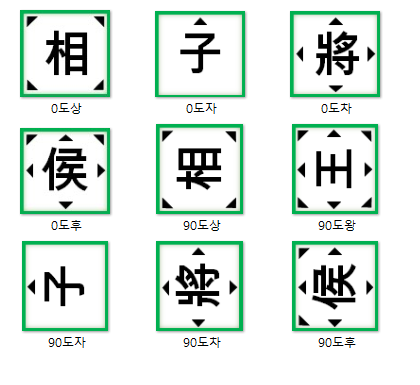
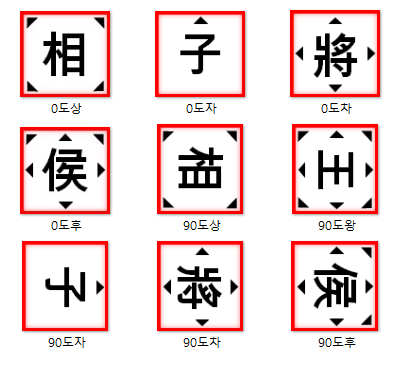
결합 디자인이 60% 정도 완료된 상태에서 가장 큰 문제점이 발생했다.  
십이 장기의 구현이 굉장히 어렵다는 사실을 뒤늦게 깨달았다. 상상 이상으로 사용해야하는 기능들이 많았고 위치 구현 설정하는 방식, 이미지 파일위에 버튼을 설정하면 버튼이 이미지 파일 밑에 존재하여 보이지 않는 등등 수정점도 나열할 수 없을 정도로 많았다.  
결합 또한 문제점이 있었다. 결합의 전체적인 디자인 중에서 상대방이 맞는 조합을 외칠 경우 게임판 옆에 ‘외쳐진 답 조합 배열’ 을 띄워 외쳐진 답의 조합을 써 놓아야 했는데, 그렇다면 어떤 구현 방법을 써야 하는지 와 실수로 JPG 형식으로 제작해버려서 이미지를 화면에 띄울 때 확대된 크기로 전부 깨져버리는 불상사 또한 존재했다. PNG로 제작했다 하더라도 스크린샷 기능으로 찍을 때 자동적으로 JPG로 설정되어서 발생한 문제였다.

덕분에 12월 4일 제작과정 전부를 이미지 제작에 써야 했고, 시험기간을 고려해서 12월 7일까지 완성하기로 했던 계획 또한 바뀌게되었다.

이 날 완성한 UI 이미지 :



[그림 3 : 십이장기 게임판 ] [그림 4 : 게임 시작 화면]



[그림 5 : 십이장기 초록말 ] [그림 6 : 십이장기 빨간말]

4) 네 번째 실패 (게임 구현 중 기능 포기) – 12월 7일

프로젝트에 시간을 투자하 대강의 디자인이 거의 다 끝난 상태였지만 구현은 역시 쉽지 않았다. 이미지를 띄운 상태에서 다른 이미지를 그 위에 띄워야 하는데 뜨지 않았고, 십이 장기의 말 또한 지정한 배열대로 가지 않았다. 콘솔창에 뜨는 배열위치는 자꾸 맞지 않게 떴고, 수식으로 작성해서 말의 배열 위치를 구현하려 했던 방법은 수많은 오류로 인해 수포로 돌아갔다. 말은 말을 먹지 않아서 자신의 배열 칸에 머물러 있었고, 이 외에도 자잘한 오류들이 속출했기에 밤을 새는게 일상이 된 시점이었다.

결합의 구현보다는 십이 장기의 구현에 거의 모든 노력을 다했고, 이 때까지는 사실상 결합의 구현 완성도는 10% 정도였다. 둘 다 구현하기에는 12월 11일에 시작하는 기말고사로 인해 일정상의 무리가 있다는 것에 팀 모두 동의하여 게임 둘 중 하나만 선택하기로 하였다. 따라서 최종적으로 십이 장기만 구현을 하게 된다.

이 날 완성한 UI 이미지 :

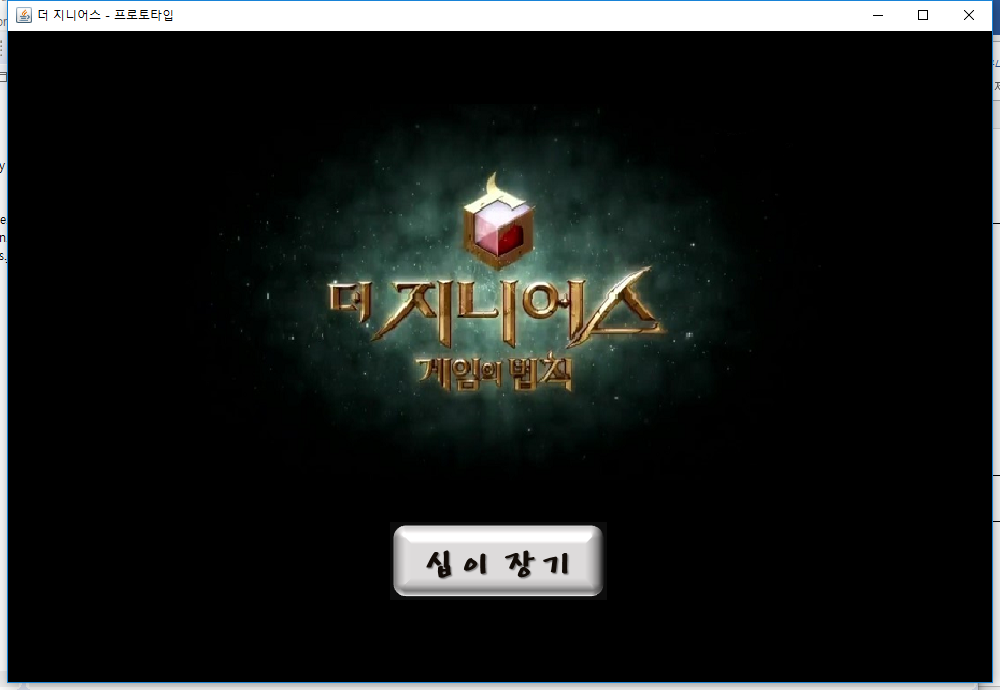


[그림 7 : 십이장기 게임룰 화면] [그림 8 : 게임에 쓰일 버튼 제작]

5) 다섯 번째 실패 (게임 구현 중 기능 포기) – 12월 11일

패널 위에 이미지가 뜨는 방식이 구현되지 않는 문제점 때문에, 등장하는 UI 순서를 대폭 수정하게 되었다. 이로 인해 [그림 8]의 물음표 버튼과 캔슬 버튼은 사용하지 않게 되었으며, 세번째 또한 이 날 이후로는 십이 장기 게임의 오류를 계속해서 수정하게 된다.

# 결과물 설명 - 구현 화면 순서

1) 게임 첫 번째 화면 – 오프닝 화면

화면 크기 : 1000 x 690

특징 : 버튼 클릭 시 옆 사진의 버튼이

와 같이 눌린 이미지로 변한다. 그 후 아래의 두 번째 화면으로 넘어간다.

2) 게임 두 번째 화면 – 게임 룰 설명 화면

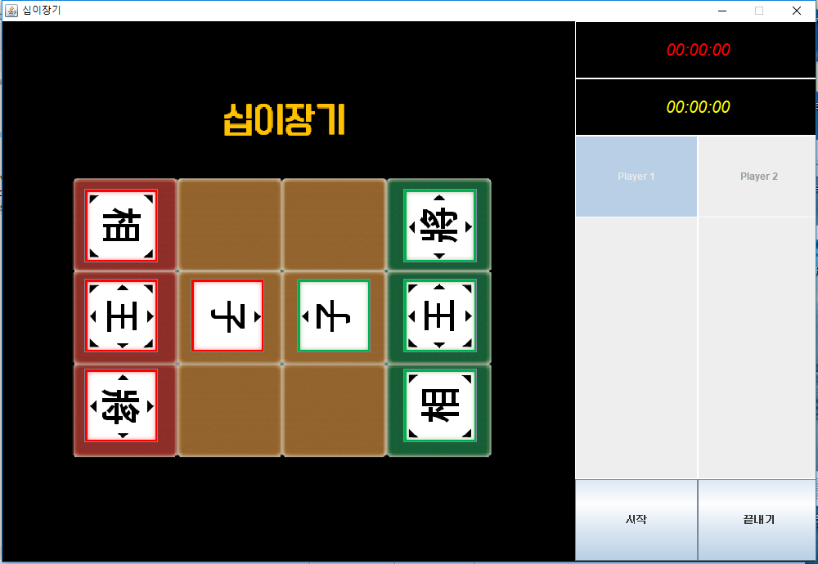
화면 크기 : 1000 x 690

특징 : 버튼 클릭 시 옆 사진의 버튼이

와 같이 눌린 이미지로 변한다. 그 후 아래의 세 번째 화면으로 넘어간다.

3) 게임 세 번째 화면 – 십이 장기 플레이 화면

화면 크기 : 1000 x 690

특징 : 가장 많은 기능들이 쓰인 화면이다.  
코드 파일이 총 4개 쓰는데, 이 화면을 구성하기 위해 3개의 파일이 쓰였다.

왼쪽 하단의 시작버튼을 누르면 player 1(빨간 말)부터 이동이 가능하다. 다른 내용은 아래에서 설명하겠다.

# 결과물 설명 – 화면 별 코드 작성 설명

**구현하려 했던 화면 -** 첫 번째 화면 & 두 번째 화면 **구현에 사용된 클래스 -** 4개 클래스 중 **Mainpage  
구현한 내용의 중점 설명 -** 게임 첫 번째 화면의 데이터를 담은 panel과 두 번째 화면을 담은 panel2가 있는데, 만약 panel의 십이장기 버튼을 누르면 panel이 setVisible(false)되고 panel2가 setVisible(true)된다. 또한 panel2의 플레이 버튼을 누르면 십이장기 게임판으로 화면이 전환된다.

**package** Prototype;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.Toolkit;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.ImageIcon;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

@SuppressWarnings("serial")

**public** **class** Mainpage **extends** JFrame{

Image img1 = Toolkit.*getDefaultToolkit*().getImage("images/게임화면/bg-1.png");

Image img2 = Toolkit.*getDefaultToolkit*().getImage("images/게임화면/규칙.png");

JPanel panel; //패널 1 - 게임 첫 번쨰 화면 구현용

JPanel panel2; //패널 2 - 게임 두 번쨰 화면 구현용

JButton main\_button1; //십이장기 버튼 구현용

JButton main\_button2; //플레이 버튼 구현용

**public** Mainpage()

{

panel = **new** JPanel()

{

**public** **void** paintComponent(Graphics g)

{

g.drawImage(img1, 0, 0, 1000, 690, **this**);

}

};

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

add(panel);

setBounds(0, 0, 1000, 690);

setSize(1000, 690); //초기 사이즈

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

add(panel);

setBounds(0, 0, 1000, 690);

setSize(1000, 690); //초기 사이즈 정의

panel.setLayout(**null**); //panel.setLayout이 절대관리자를 의미하는데, 만약에 ()이 괄호 안에 null이 있을 경우에 절대 관리자가 실행됨.따라서 null을 적어놓은 상태에서 65번 줄의 main\_button1.setBounds(100, 500, 700, 300); 숫자를 바꿔서 위치를 결정하면 됨.

ImageIcon normal\_janggi\_button = **new** ImageIcon("images/버튼종류/십이장기버튼1.png"); //이미지들 정의

ImageIcon press\_janggi\_button = **new** ImageIcon("images/버튼종류/십이장기버튼2.png"); //마우스 클릭될 때의 이미지

ImageIcon normal\_play\_button = **new** ImageIcon("images/버튼종류/플레이버튼1.png"); //이미지들 정의

ImageIcon press\_play\_button = **new** ImageIcon("images/버튼종류/플레이버튼2.png"); //마우스 클릭될 떄의 이미지

JButton main\_button1 = **new** JButton(normal\_janggi\_button); // 장기 버튼 이미지를 main\_button1에 추가

main\_button1.setPressedIcon(press\_janggi\_button); //main\_button1이 눌릴때 이미지 추가

JButton main\_button2 = **new** JButton(normal\_play\_button); //플레이 버튼 이미지를 main\_button2에 추가

main\_button2.setPressedIcon(press\_play\_button); //main\_button2가 눌릴 때 이미지 바뀜

//버튼 투명화

main\_button1.setBorderPainted(**false**);

main\_button1.setFocusPainted(**false**);

main\_button1.setContentAreaFilled(**false**);

main\_button2.setBorderPainted(**false**);

main\_button2.setFocusPainted(**false**);

main\_button2.setContentAreaFilled(**false**);

//패널에 버튼 추가

panel.add(main\_button1);

main\_button1.addActionListener(**new** ActionListener(){ //마우스로 십이장기 버튼 클릭하면

**public** **void** actionPerformed(ActionEvent e){

panel.setVisible(**false**); //panel 1이 안보이게 됩니다.

panel2 = **new** JPanel() //panel 2를 여기서 생성하고

{

**public** **void** paintComponent(Graphics g)

{

g.drawImage(img2, 0, 0, 1000, 690 , **this**);

//panel 2에 게임룰 이미지를 넣습니다.

}

};

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

add(panel2);

panel2.add(main\_button2); //panel 2에 플레이 버튼을 추가합니다

setBackground(Color.***BLACK***);//검정배경 만들기

setSize(1000, 690); //초기 사이즈

setVisible(**true**); //시야에 보이게 합니다

main\_button2.addActionListener(**new** ActionListener(){

//panel 2에 있는 플레이 버튼이 클릭되면

**public** **void** actionPerformed(ActionEvent e){

panel2.setVisible(**false**); //panel 2가 안보이게 됩니다

**new** Janggi(); //Janggi로 넘어가서 화면이 바뀌게 됩니다.

}

}

);

}

});

setVisible(**true**);

main\_button1.setBounds(380, 480, 220, 100); //십이장기 버튼 위치, 폭, 넓이 지정

// b.setBounds(x좌표, y좌표, 가로길이, 세로길이); //장기버튼 위치

}

**public** **static** **void** main(String[] args)

{

Mainpage G = **new** Mainpage();

G.setTitle("더 지니어스 - 프로토타입");

}

}

**구현하려 했던 화면 –** 세 번째 화면의 장기말 움직임 **구현에 사용된 클래스** -4개 클래스 중 **MovePieces  
구현한 내용의 중점 설명 –** 게임 화면 중 말이 움직일 수 있는 위치를 설정하는 내용입니다. 배열로 작성하였습니다.

**public** **class** MovePieces {

**public** **boolean** CheckMalMove(**int**[][] board, **int** i, **int**[] preposition, **int**[] nowposition)

{

//움직이려는 말이 왕 일 경우

**if** (i % 10 == 1)

{ **if** (King\_move(preposition, nowposition))

// 만약 전에 있던 위치와 현재 위치에 관해서 King\_move 함수?를 적용시킨다

{ **if**(preposition[0] <5 && preposition[1] <4 )

**return** **true**;

} **else**

**return** **false**;

}

//움직이려는 말이 상일 경우

**else** **if** (i % 10 == 2 || i % 10 == 3 )

{

**if** (sang\_move(preposition, nowposition))

**if**(preposition[0] <5 && preposition[1] <4 )

**return** **true**;

**else**

**return** **false**;

}

//움직이려는 말이 빨간 자일 경우

**else** **if** (i == 4 || i == 5) {

**if** ( R\_ja\_move(preposition, nowposition))

**if**(preposition[0] <5 && preposition[1] <4 )

**return** **true**;

**else**

**return** **false**;

}

//움직이려는 말이 녹색 자일 경우

**else** **if** (i == 14 || i == 15) {

**if** ( G\_ja\_move(preposition, nowposition))

**return** **true**;

**else**

**return** **false**;

}

//움직이려는 말이 장일 경우

**else** **if** (i % 10 == 6 || i % 10 == 7){

{

**if** (jang\_move(preposition, nowposition))

**return** **true**;

**else**

**return** **false**;

}

//움직이려는 말이 빨간 후일 경우

} **else** **if** (i == 8 || i == 9) {

**if** ( R\_who\_move(preposition, nowposition))

**return** **true**;

**else**

**return** **false**;

}

//움직이려는 말이 녹색 후일 경우

**else** **if** (i == 18 || i == 19) {

**if** ( G\_who\_move(preposition, nowposition))

**if**(preposition[0] <5 && preposition[1] <4 )

**return** **true**;

**else**

**return** **false**;

}

**return** **false**;

}

//ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

/\*

왕 [i][j]에서 [i][j+1], [i][j-1], [i+1][j], [i-1][j], [i][j-1], [i][j+1], [i+1][j+1], [i-1][j-1]로 이동가능

자 [i][j]에서 [i][j+1]로 이동가능 빨/녹 다름

상 [i][j]에서 [i+1][j+1], [i+1][j-1], [i-1][j+1], [i-1][j-1]로 이동가능

장 [i][j]에서 [i+1][j] ,[i-1][j] , [i][j-1], [i][j+1]로 이동가능

후 [i][j]에서 [i-1][j-1], [i-1][j],[i-1][j+1], [i][j+1], [i][j-1], [i+1][j]로 이동가능 빨/녹 다름

\*/

//왕 이동가능 방향

**boolean** King\_move(**int**[] preposition, **int**[] nowposition) {

**if**(preposition[0] == nowposition[0]) //

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0] + 1 == nowposition[0])

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1] == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0] - 1 == nowposition[0])

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1] == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else**

**return** **false**;

}

//상 이동가능 방향

**boolean** sang\_move(**int**[] preposition, **int**[] nowposition) {

**if**(preposition[0] + 1 == nowposition[0])

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0] - 1 == nowposition[0])

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else**

**return** **false**;

}

//빨간 자 이동가능방향

**boolean** R\_ja\_move(**int**[] preposition, **int**[] nowposition) {

**if**(preposition[1] == nowposition[1]) //x좌표가 같음

{

**if**(preposition[0]+1 == nowposition[0])

**return** **true**;

**else**

**return** **false**;

}

**else**

**return** **false**;

}

//녹색 자 이동 가능 방향

**boolean** G\_ja\_move(**int**[] preposition, **int**[] nowposition) {

**if**(preposition[1] == nowposition[1]) //x좌표가 같음

{

**if**(preposition[0]-1 == nowposition[0])

**return** **true**;

**else**

**return** **false**;

}

**else**

**return** **false**;

}

//장 이동가능방향

**boolean** jang\_move(**int**[] preposition, **int**[] nowposition)

{

**if**(preposition[0] == nowposition[0]) //x좌표가 같음

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[1] == nowposition[1]) //x좌표가 같음

{

**if**(preposition[0]+1 == nowposition[0])

**return** **true**;

**if**(preposition[0]-1 == nowposition[0])

**return** **true**;

**else**

**return** **false**;

}

**else**

**return** **false**;

}

//빨간 후 이동 가능 방향

**boolean** R\_who\_move(**int**[] preposition, **int**[] nowposition)

{

**if**(preposition[0]-1 == nowposition[0])

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1] == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0] == nowposition[0])

{

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0]+1 == nowposition[0])

{

**if**(preposition[1] == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**return** **false**;

}

//초록색 후 이동가능 방향

**public** **boolean** G\_who\_move(**int**[] preposition, **int**[] nowposition)

{

**if**(preposition[0]+1 == nowposition[0])

{

**if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else** **if**(preposition[1] == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0] == nowposition[0])

{

**if**(preposition[1]-1 == nowposition[1])

**return** **true**;

**if**(preposition[1]+1 == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**else** **if**(preposition[0]-1 == nowposition[0])

{

**if**(preposition[1] == nowposition[1])

**return** **true**;

**else**

**return** **false**;

}

**return** **false**;

}

}

**구현하려 했던 화면 –** 세 번째 화면 게임판 옆 정보를 알려주는 menu판 **구현에 사용된 클래스 -** 4개 클래스 중 **Information  
구현한 내용의 중점 설명 –** 게임 화면 중 옆에 표시되는 턴의 남은 시간 / 플레이 총 시간 / 현재 턴 표시용 플레이어패널 / 먹힌 장기말 알려주는 보드 / 시작버튼과 끝내기 버튼. 한 수 물리기 버튼이 있습니다.

**package** Prototype;

**import** java.awt.Color;

**import** java.awt.FlowLayout;

**import** java.awt.Font;

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.ImageIcon;

**import** javax.swing.JButton;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**import** javax.swing.JToggleButton;

**import** javax.swing.SwingConstants;

**import** javax.swing.border.LineBorder;

**import** javax.swing.border.TitledBorder;

@SuppressWarnings("serial")

**public** **class** Information **extends** JPanel {

JLabel RemainTime, PlayTime;

JButton Startbu; //시작 버튼

JButton Endbu; //끝내기 버튼

JPanel jp2;

JButton onesuB; //시작버튼이 눌린 후 한

JToggleButton[] Player;

PlayTimer PlayTimer; //총 플레이 시간

RemainTimer RemainTimer; //현재 턴 남은 시간

**static** JPanel[] *PlayerPanel*;

**int** hours, mins, secs, hours1, mins1, secs1; //시간 표시 변수

**double** oo, xx, oo1, xx1; //시간 표시할 때 쓰일 시간 계산용 변수

**boolean** loop, first = **true**, TurnIsChange = **false**, imageIsChange = **false**;

ImageIcon icon;

**boolean** IsStart;

**void** setTurnIsChangeToTrue() {

// TurnIsChange를 True로 만듬.턴이바뀌면 남은시간 초기화를 위해서

TurnIsChange = **true**;

SetTurn();

}

**void** setimageIsChangeToTrue() {

imageIsChange = **true**;

}

**void** SetTurn(){//턴이 바뀌면 토글버튼도 바뀌게

**if**(Player[0].isSelected())

{

Player[0].setSelected(**false**);

Player[1].setSelected(**true**);

}

**else**

{

Player[1].setSelected(**false**);

Player[0].setSelected(**true**);

}

}

**void** ResetTimer(){

PlayTime.setText("00:00:00");

RemainTime.setText("00:00:00");

onesuB.setVisible(**false**);

Startbu.setVisible(**true**);

PlayTimer.~~suspend~~();

RemainTimer.~~suspend~~();

IsStart=**false**;

}

**public** Information() {

setLayout(**null**);

icon = **new** ImageIcon("images/게임화면/board.png");

jp2 = **new** JPanel();

//time = new JPanel();

jp2.setLayout(**null**);

add(jp2);

//제한시간 표시 설정

TitledBorder TB = **new** TitledBorder(**new** LineBorder(Color.***white***));

Font font = **new** Font("Arial", Font.***ITALIC***, 20);

RemainTime = **new** JLabel();

RemainTime.setBounds(0, 0, 300, 70);

RemainTime.setHorizontalAlignment(SwingConstants.***CENTER***);

RemainTime.setForeground(Color.***red***);

RemainTime.setText("00:00:00");

RemainTime.setBorder(TB);

RemainTime.setFont(font);

PlayTime = **new** JLabel(){

**public** **void** paintComponents(Graphics g){

//super.paintComponents(g);

g.drawImage(icon.getImage(), 0, 0, 50, 80, **null**);

}

};

//플레이시간 표시 설정

PlayTime.setBounds(0, 70, 300, 70);

PlayTime.setHorizontalAlignment(SwingConstants.***CENTER***);

PlayTime.setForeground(Color.***yellow***);

PlayTime.setText("00:00:00");

PlayTime.setBorder(TB);

PlayTime.setFont(font);

//턴 넘기는 버튼 설정

onesuB = **new** JButton();

onesuB.setBounds(0, 560, 150, 100);

onesuB.setVisible(**false**);

onesuB.setText("턴 넘기기");

onesuB.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

Janggi.*changeTurn*();

setTurnIsChangeToTrue();

}

});

//player 1, 2 나타내는 패널

Player = **new** JToggleButton[2];

**for** (**int** i = 0; i < 2; i++) {

Player[i] = **new** JToggleButton();

Player[i].setText("Player " + (i + 1));

Player[i].setBorder(TB);

Player[i].setBounds(150 \* i, 140, 150, 100);

Player[i].setEnabled(**false**);

add(Player[i]);

}

Player[0].setSelected(**true**);

//player 1, 2 가 먹은 말을 나타내는 패널

*PlayerPanel* = **new** JPanel[2];

**for** (**int** i = 0; i < 2; i++)

{

*PlayerPanel*[i] = **new** JPanel();

*PlayerPanel*[i].setLayout(**new** FlowLayout());

*PlayerPanel*[i].setBounds(150 \* i, 235, 150, 325);

*PlayerPanel*[i].setBorder(TB);

add(*PlayerPanel*[i]);

}

Startbu = **new** JButton();

Startbu.setBounds(0, 560, 150, 100);

Startbu.setText("시작");

Startbu.addMouseListener(**new** ButtonMouseListener());

Endbu = **new** JButton();

Endbu.setBounds(150, 560, 150, 100);

Endbu.setText("끝내기");

Endbu.addMouseListener(**new** ButtonMouseListener());

add(RemainTime);

add(PlayTime);

add(onesuB);

add(Startbu);

add(Endbu);

setFocusable(**false**);

setLocation(700, 0);// 게임판 옆에 정보나타내주는 영역의 시작지점

setBackground(Color.***BLACK***);//검정배경 만들기

setSize(500, 1000); // 정보 보드 크기

setVisible(**true**);

PlayTimer = **new** PlayTimer();

RemainTimer = **new** RemainTimer();

}

**class** PlayTimer **extends** Thread {

**void** timer1() {// 플레이 시간

xx = (System.*currentTimeMillis*() - oo) / 1000d;

hours = ((**int**) xx % 86400) / 3600;

mins = ((**int**) xx % 3600) / 60;

secs = (**int**) xx % 60;

PlayTime.setText(String.*format*("%02d:%02d:%02d ", hours, mins, secs));

}

**public** **void** run() {

**while** (**true**) {

**try** {

**if** (loop)

{

Thread.*sleep*(100);

timer1();

}

**else**

{

Thread.*sleep*(1000 \* 60 \* 60 \* 24); // 24시간 정지상태

}

} **catch** (InterruptedException e)

{

**break**;

}

}

}

}

**class** RemainTimer **extends** Thread {

**void** timer() {

// 남은시간

xx1 = (oo1 + (30000d) - System.*currentTimeMillis*()) / 1000d;

hours1 = ((**int**) xx1 % 86400) / 3600;

mins1 = ((**int**) xx1 % 3600) / 60;

secs1 = (**int**) xx1 % 60;

RemainTime.setText(String.*format*("%02d:%02d:%02d ", hours1, mins1,

secs1));

// 턴이 바뀌었거나 시간이 초과됐을때

**if** (oo1 + (30000d) - System.*currentTimeMillis*() <= 0d || TurnIsChange)

{

**if** (oo1 + (30000d) - System.*currentTimeMillis*() <= 0d)

// 만약 상태가 시간초과라면

{ Janggi.*changeTurn*();// 턴을 바꾼다

setTurnIsChangeToTrue();

}

oo1 = System.*currentTimeMillis*();

TurnIsChange = **false**;

}

}

**public** **void** run() {

**while** (**true**) {

**try** {

**if** (loop) {

Thread.*sleep*(100);

timer();

} **else** {

Thread.*sleep*(1000 \* 60 \* 60 \* 24); // 24시간 정지상태

}

} **catch** (InterruptedException e) {

**break**;

}

}

}

}

**class** ButtonMouseListener **implements** MouseListener {

@SuppressWarnings("deprecation")

@Override

**public** **void** mouseClicked(MouseEvent e) {

**if** (e.getSource() == Startbu) {

**if** (first) {// 처음 시작 눌렀을때(처음 실행할때)

loop = **true**;

oo = System.*currentTimeMillis*();//기준 시간을 정한다.

oo1 = System.*currentTimeMillis*();//기준 시간을 정함

PlayTimer.start();

RemainTimer.start();

first = **false**;

}

**else** {// 끝내기 누른후 시작 눌렀을때(처음 실행이 아닐때)

oo = System.*currentTimeMillis*();

oo1 = System.*currentTimeMillis*();

PlayTimer.~~resume~~();

RemainTimer.~~resume~~();

}

IsStart=**true**;

onesuB.setVisible(**true**);

Startbu.setVisible(**false**);

} **else** **if** (e.getSource() == Endbu) {

ResetTimer();

IsStart=**false**;

}

}

@Override

**public** **void** mouseEntered(MouseEvent e) { //마우스가 접근했을 때}

@Override

**public** **void** mouseExited(MouseEvent e) { //마우스가 접근안했을 때}

@Override

**public** **void** mousePressed(MouseEvent e) { //마우스 눌렸을 때}

@Override

**public** **void** mouseReleased(MouseEvent e) { //마우스 떼졌을 때}

}

}

**구현하려 했던 화면 –** 세 번째 화면 **구현에 사용된 클래스 -** 4개 클래스 중 **Janggi  
구현한 내용의 중점 설명 –** 게임 시작시 배열에 말 저장 / 장기말이 움직일 경우 움직이는 위치를 콘솔창에 알려주는 기능 & / 왕이 진영에 들어가서 2턴 버티면 이기는 룰 / 이겼을 경우 나오는 알림창과 왕이 상대진영에 진입했을 때 나오는 경고창 / 장기말 중 자가 상대진영에 진입했을 경우 후로 바뀌는 기능 / 장기말이 장기말을 먹을 경우 장기말이 옆의 정보창으로 가는 기능

**package** Prototype;

**import** java.awt.Graphics;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** java.awt.event.MouseMotionListener;

**import** javax.swing.ImageIcon;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

@SuppressWarnings("serial")

**public** **class** Janggi {

**final** **int** NULL = 0; //비어있는 상태 0

**final** **int** r90king = 1; //빨간말은 1부터 9까지

**int** r0sang = 2;

**int** r90sang = 3;

**int** r0ja = 4;

**int** r90ja = 5;

**int** r0cha = 6;

**int** r90cha = 7;

**int** r0who = 8;

**int** r90who = 9;

**final** **int** g90king = 11; //녹색말은 11부터 19까지

**int** g0sang = 12;

**int** g90sang = 13;

**int** g0ja = 14;

**int** g90ja = 15;

**int** g0cha = 16;

**int** g90cha = 17;

**int** g0who = 18;

**int** g90who = 19;

**static** **int** *k\_count1* = 0;

//king이 상대진영에서 2턴 버티면 이기는 것을 판단하는 객체.

**static** **int** *k\_count2* = 0; // k\_count = 2가 되면 끝난다.

**static** **int** *myTurn* = 0; //첫차례는 player1이 먼저 시작한다

**boolean** start = **false**; // start가 false로 시작, true가 되면 게임이 진행됨.

Information information;

CheckStart check;

JFrame mainFrame; //게임 전체 프레임을 만든다.

JPanel panel = **new** JPanel(); //그 안에 넣을 패널을 만듬.

JPanel gameZone = **new** JPanel() {

**public** **void** paintComponent(Graphics g) {

g.drawImage(icon.getImage(), 0, 0, 700, 700, **this**);

setOpaque(**false**);

// background의 알파도를 0으로 만들어서 배경을 투명하게 만드는 함수

**super**.paintComponent(g);

}

};

ImageIcon icon;

MovePieces checkmove = **new** MovePieces();

**static** **int**[][] *janggiBoard*;

**int**[] malIndex, malIndextmp;

**public** Janggi() {

mainFrame = **new** JFrame();

mainFrame.setTitle("십이장기");

mainFrame.setLayout(**null**);

mainFrame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

information = **new** Information();

icon = **new** ImageIcon("images/게임화면/board.png");

check = **new** CheckStart();

panel.setLayout(**null**);

gameZone.setLayout(**null**);

*janggiBoard* = Janggipan(); // Janggipan = 게임 맨처음 시작 말 배치

gameZone = Locate(*janggiBoard*, gameZone);

panel.add(gameZone); //게임 플레이패널 추가

panel.add(information); //시간 및 정보 패널 추가

check.start();

gameZone.setSize(700, 700); // 700x700으로 되어있는 십이장기 말판크기

mainFrame.setContentPane(panel);

mainFrame.setBounds(0, 0, 1000, 690); //게임전체크기

mainFrame.setResizable(**false**);

mainFrame.setVisible(**true**);

}

**private** **int**[][] Janggipan() {//장기판 처음에 말을 저장함

**int**[][] Janggipan = {

{r90sang, r90king, r90cha},

{NULL, r90ja, NULL},

{NULL, g90ja, NULL},

{g90cha, g90king, g90sang} };

**return** Janggipan;

}

// 스레드를 이용해서 게임을 run시키는 것

**class** CheckStart **extends** Thread

{

**int** count = 0;

**public** **void** run(){

**while** (**true**){

**if**(!information.IsStart&& count==0){

*janggiBoard* = Janggipan();// 초기 보드 배열상태를 Janggipan으로 초기화

gameZone.removeAll();

gameZone = Locate(*janggiBoard*, gameZone);

// 말을 게임화면에 배치시킨다

gameZone.repaint();// 화면 갱신

*myTurn* = 0; // 차례는 Player1부터 시작

information.Player[0].setSelected(**true**);

//setSelected가 번갈아가며 true와 false가 되며

information.Player[1].setSelected(**false**); //차례를 나타낸다.

count++;

}

**else** **if**(information.IsStart && count != 0){

count = 0;

}

}

}

}

//말 하나하나의 설정

**public** **class** mal **extends** JButton {

**int** x;

**int** y;

ImageIcon icon;

**public** mal(**int** i, **int** j, String imagePath)

//JPanel Locate에서 for로 말의 정보를 하나하나씩 넣고 여기서는 위치를 정의한다.

{

setSize(90, 90); //말 하나의 크기

setLocation((i \* 130 + 100), (j \* 110 + 205)); //말위치

icon = **new** ImageIcon(imagePath);

addMouseListener(**new** MouseListener() {

@Override

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

**if** (information.IsStart) { //start버튼 누르면

JButton btn = (JButton) e.getSource(); //클릭이 활성화된다.

btn.setSelected(**false**);

x = e.getXOnScreen() - mainFrame.getX()-50; //x좌표를 구한다.

y = e.getYOnScreen() - mainFrame.getY()-50; //y좌표를 구한다.

System.***out***.println("클릭한 좌표의 위치는 "+x + ", " + y);

// 콘솔창에 x,y좌표의 위치가 나옴

malIndex = getIndex(x, y); //getIndex로 x좌표와 y좌표의 값으로 현재 배열의 위치를 구한다.

System.***out***.println("이동할 배열 좌표 위치 = ("+malIndex[0] + "," + malIndex[1]+")"); //이동할 곳의 배열의 위치를 콘솔창에 나타낸다.

System.***out***.println("지금은 player"+(*myTurn*+1)+" 차례"); //콘솔창에 차례를 나타내준다.

System.***out***.println("player1의 kingcount는 "+(*k\_count1*));

//각 플레이어의 kingcount를 콘솔창에 나타내준다.

System.***out***.println("player2의 kingcount는 "+(*k\_count2*));

**if** (*janggiBoard*[malIndextmp[0]][malIndextmp[1]] / 10 == *myTurn*) {//자신턴인 경우에만 자신말을 움직일 수 있다.

**if** (malIndex[0] == malIndextmp[0] && malIndextmp[1] == malIndex[1]); // 가려는 곳에 내 말이 있는경우 아무일도 일어나지 않음

/\*말이 헷갈려서 하나로 정의함. 현재인덱스 = 움직인후 인덱스 , 이전인덱스= 움직이기전 인덱스\*/

**else** **if** (checkmove.CheckMalMove(*janggiBoard*, *janggiBoard*[malIndextmp[0]][malIndextmp[1]], malIndextmp, malIndex))

{ //checkmove를 통해 선택된 이전 인덱스와 현재인덱스를 비교해서 움직일 수 있는 곳인지 판단한다. 움직일 수 없는 곳이면 checkmove에서 false로 반환하여 다시 선택해야한다.

**if** (*janggiBoard*[malIndex[0]][malIndex[1]] == NULL)

{ //현재 인덱스가 비어있는 경우

**if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 5 && malIndex[0] == 3) // 빨간자가 상대 진영에 도착해서 뒤 후로 변할경우

{

*janggiBoard*[malIndex[0]][malIndex[1]] = 9; //현재인덱스는 자에서 후로 바뀌며

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전 인덱스는 NULL값으로 초기화. 아무것도 없는 상태

checkJang();//차례를 끝마침.

}

**else** **if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 15 && malIndex[0]==0) //녹색자가 상대 진영에 도착해서 뒤 후로 변할경우

{

*janggiBoard*[malIndex[0]][malIndex[1]] = 19; //현재인덱스는 자에서 후로 바뀌며

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전 인덱스는 NULL값으로 초기화. 아무것도 없는 상태

checkJang(); //차례를 끝마침.

}

**else** **if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 1 && malIndex[0] == 3) //빨간 킹이 비어있는 녹색진영에 들어가서 카운트가 올라가는 경우

{

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]]; //일단 그곳을 먹고

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL;

*k\_count1* = 1; //카운트가 1이 증가한다.

JOptionPane.*showConfirmDialog*(**null**, "왕이 녹색진영에 들어왔습니다.", "경고", JOptionPane.***WARNING\_MESSAGE***);

checkJang(); //차례를 마침.

}

**else** **if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 11 && malIndex[0] == 0) //녹색 킹이 비어있는 빨간진영에 들어가서 카운트가 올라가는 경우

{

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]]; //그곳을 먹고

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL;

*k\_count2* = 1; //카운트가 1이 증가하고

JOptionPane.*showConfirmDialog*(**null**, "왕이 빨간진영에 들어왔습니다.", "경고", JOptionPane.***WARNING\_MESSAGE***);

checkJang(); //차례를 마친다.

}

**else** { //그냥 비어있는 공간일 경우

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]]; //현재 인덱스에 이전 인덱스값을 넣고

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전인덱스는 NULL로 초기화.

checkJang(); //차례를 마침.

}

}

**else** **if** (*janggiBoard*[malIndex[0]][malIndex[1]] / 10 == *janggiBoard*[malIndextmp[0]][malIndextmp[1]] / 10);

//이동할 곳이 같은 편이 있을 경우 아무일도 일어나지 않음.

**else** //현재 인덱스에 상대방 말이 있어서 이전인덱스가 잡아먹는 경우

{

**if** (*janggiBoard*[malIndex[0]][malIndex[1]] == 11 ) //현재인덱스의 값이 player2의 왕인 경우

{ //player1의 승리로 게임 끝

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]];

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL;

start = **false**;

JOptionPane.*showConfirmDialog*

(**null**, "player1의 승리", "안내", JOptionPane.***WARNING\_MESSAGE***);

information.IsStart = **false**;

information.ResetTimer();

information.IsStart= **false**;

}

**else** **if** (*janggiBoard*[malIndex[0]][malIndex[1]] == 1 ) //현재인덱스의 값이 player1의 왕인경우

{ //player2의 승리로 게임끝

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]];

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL;

start = **false**;

JOptionPane.*showConfirmDialog*

(**null**, "Player2의 승리", "안내", JOptionPane.***WARNING\_MESSAGE***);

information.IsStart = **false**;

information.ResetTimer();

information.IsStart=**false**;

}

**else**

{

**if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 5 && malIndex[0] == 3) // 빨간자가 상대진영에 있는 현재인덱스를 먹고 뒤후가 되는경우

{

setPlayerEatMal(*janggiBoard*[malIndex[0]][malIndex[1]]); //먹은 말(현재인덱스값)을패널에 나타내줌.

*janggiBoard*[malIndex[0]][malIndex[1]] = 9; //현재인덱스에 뒤 후 값을 넣어줌.

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전인덱스는 NULL값이 됨.

checkJang();//차례를 마침

}

**else** **if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 15 && malIndex[0]==0) // 녹색자가 상대진영에 있는 현재인덱스를 먹고 뒤후가 되는경우

{

setPlayerEatMal(*janggiBoard*[malIndex[0]][malIndex[1]]);

//먹은 말(현재인덱스값)을패널에 나타내줌.

*janggiBoard*[malIndex[0]][malIndex[1]] = 19; //현재 인덱스의 녹색 뒤후를 넣어줌

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전 인덱스는 NULL값이 됨.

checkJang();//차례를 마침.

}

**else** **if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 1 && malIndex[0] == 3) //빨간 킹이 녹색진영에 들어가서 카운트가 올라가는 경우

{

setPlayerEatMal(*janggiBoard*[malIndex[0]][malIndex[1]]);

//먹은 말(현재인덱스값)을패널에 나타내줌.

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]]; //현재인덱스에 이전인덱스값을 넣어줌.

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전 인덱스는 NULL값이 됨.

*k\_count1* = 1; //카운트는 증가

JOptionPane.*showConfirmDialog*(**null**, "왕이 녹색진영에 들어왔습니다.", "경고", JOptionPane.***WARNING\_MESSAGE***);

checkJang();//차례를 마침.

}

**else** **if**(*janggiBoard*[malIndextmp[0]][malIndextmp[1]] == 11 && malIndex[0] == 0) //녹색킹이 빨간진영에 들어가서 카운트가 올라가는 경우

{

setPlayerEatMal(*janggiBoard*[malIndex[0]][malIndex[1]]); //먹은 말(현재인덱스값)을패널에 나타내줌.

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]]; //현재인덱스에 이전인덱스값을 넣어줌.

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전 인덱스는 NULL값이 됨.

*k\_count2* = 1; //카운트는 증가

JOptionPane.*showConfirmDialog*(**null**, "왕이 빨간진영에 들어왔습니다.", "경고", JOptionPane.***WARNING\_MESSAGE***);

checkJang(); //차례를 마침.

}

**else** //특수한 경우가 아니고, 이전인덱스가 현재인덱스를 먹는경우

{

setPlayerEatMal(*janggiBoard*[malIndex[0]][malIndex[1]]); //먹은 말(현재인덱스값)을패널에 나타내줌.

*janggiBoard*[malIndex[0]][malIndex[1]] = *janggiBoard*[malIndextmp[0]][malIndextmp[1]]; //현재인덱스에 이전인덱스값을 넣어줌.

*janggiBoard*[malIndextmp[0]][malIndextmp[1]] = NULL; //이전 인덱스는 NULL값이 됨.

checkJang(); //차례를 마침.

}

}

}

}

}

//차례가 끝나고 현재 말판의 상태를 갱신해준다.

gameZone.removeAll();

gameZone = Locate(*janggiBoard*, gameZone);

gameZone.repaint();

System.***out***.println();

}

}

@Override

**public** **void** mousePressed(MouseEvent e) { //그냥 누르기만 했을때도 배열의 위치를 콘솔창을 통해 볼 수 있게 한다.

// **TODO** Auto-generated method stub

JButton btn = (JButton) e.getSource();

btn.setSelected(**false**);

x = e.getXOnScreen() - mainFrame.getX() - 50;//

y = e.getYOnScreen() - mainFrame.getY() - 50;//

malIndextmp = getIndex(x, y); // 좌표 x,y값을 getIndex를 통해 배열의 위치를 나타냄.

System.***out***.println("클릭 한 위치 배열 좌표 "+ malIndextmp[0] + ", " + malIndextmp[1]);

}

@Override

**public** **void** mouseExited(MouseEvent arg0) {

// **TODO** Auto-generated method stub}

@Override

**public** **void** mouseEntered(MouseEvent arg0) {

// **TODO** Auto-generated method stub}

@Override

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

});

addMouseMotionListener(**new** MouseMotionListener()

{

@Override

**public** **void** mouseDragged(MouseEvent e) {

// **TODO** Auto-generated method stub

**if** (information.IsStart) {

JButton btn = (JButton) e.getSource();

x = e.getXOnScreen() - mainFrame.getX();

y = e.getYOnScreen() - mainFrame.getY();

**if** (x < 700 && x > 100 && y < 700 && y > 100) //말이 움직일수있는 범위

btn.setLocation(x -45, y - 80); //움직이는 말을 드래그할때 위치하는 마우스의 위치

}

}

@Override

**public** **void** mouseMoved(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

}

);

}

**public** **void** paintComponent(Graphics g) //

{

g.drawImage(icon.getImage(), 0, 0, 90, 90, **null**);

}

}

**void** setPlayerEatMal(**int** killedMal) { //먹히는말이 여기로 와서 JLabel로 PlayerPanel에 나타나서 보여준다.

String MalImg = **null**;

//빨간색 말을 먹는경우

**if**(*myTurn* == 1) {

**if**(killedMal ==1)

MalImg = "images/십이장기빨간말/r90king.png";

**else** **if**(killedMal ==3)

MalImg = "images/십이장기빨간말/r0sang.png";

**else** **if** (killedMal ==5 || killedMal ==9)

MalImg = "images/십이장기빨간말/r0ja.png";

**else** **if**(killedMal==7)

MalImg = "images/십이장기빨간말/r0cha.png";

}

//초록색 말을 먹는경우

**else** {

**if**(killedMal == 11)

MalImg = "images/십이장기초록말/g90king.png";

**else** **if**(killedMal ==13)

MalImg = "images/십이장기초록말/g0sang.png";

**else** **if** (killedMal ==15 || killedMal ==19)

MalImg = "images/십이장기초록말/g0ja.png";

**else** **if**(killedMal==17)

MalImg = "images/십이장기초록말/g0cha.png";

}

Information.*PlayerPanel*[*myTurn*].add(**new** JLabel(**new** ImageIcon(MalImg)));

//먹은말을 자신의 차례의 PlayerPanel에 추가한다.

}

**public** **int**[] getIndex(**int** ox, **int** oy) {

**int**[] Index = **new** **int**[2]; //index[0]과 index[1]을 생성함

**int** i = 0 , j = 0;

**if** (ox >=50 && ox <= 150) //클릭한 x좌표 위치의 범위에 따라 선택된 x좌표 배열이 결정

i = 0;

**else** **if**(ox >= 170 && ox <= 270) // 다양한 방법들을 고민했지만 좌표값에 따른 배열판단이 가장 정확한 방법일 것 같아서

i = 1; // 하나하나의 좌표를 구하고 좌표값에 나눠서 배열 [x],[y]의 값을 구하는 방법을 선택.

**else** **if**(ox >= 310 && ox <= 410)

i = 2;

**else** **if**(ox >= 440 && ox <= 540)

i = 3;

**if**(oy >= 180 && oy <= 280) //클릭한 y좌표 위치의 범위에 따라 선택된 y좌표 배열이 결정

j = 0;

**else** **if**(oy >= 300 && oy <=400)

j = 1;

**else** **if**(oy >= 410 && oy <= 510)

j = 2;

Index[0] = i;

Index[1] = j;

**return** Index;

}

//board배열에 있는 장기말을 장기판에 이미지로 나타내준다.

**public** JPanel Locate(**int**[][] board, JPanel Janggi) {

**for** (**int** i = 0; i < 4; i++) {

**for** (**int** j = 0; j < 3;j++) {

**if** (board[i][j] == r90king) {

mal r90king = **new** mal(i, j, "images/십이장기빨간말/r90king.png");

r90king.setName("images/십이장기빨간말/r90king.png");

Janggi.add(r90king);

} **else** **if** (board[i][j] == r0sang) {

mal r0sang = **new** mal(i, j, "images/십이장기빨간말/r0sang.png");

r0sang.setName("images/십이장기빨간말/r0sang.png");

Janggi.add(r0sang);

} **else** **if** (board[i][j] == r90sang) {

mal r90sang = **new** mal(i, j, "images/십이장기빨간말/r90sang.png");

r90sang.setName("images/십이장기빨간말/r90sang.png");

Janggi.add(r90sang);

} **else** **if** (board[i][j] ==r0ja) {

mal r0ja = **new** mal(i, j, "images/십이장기빨간말/r0ja.png");

r0ja.setName("images/십이장기빨간말/r0ja.png");

Janggi.add(r0ja);

} **else** **if** (board[i][j] == r90ja ) {

mal r90ja = **new** mal(i, j, "images/십이장기빨간말/r90ja.png");

r90ja.setName("images/십이장기빨간말/r90ja.png");

Janggi.add(r90ja );

} **else** **if** (board[i][j] == r0cha) {

mal r0cha = **new** mal(i, j, "images/십이장기빨간말/r0cha.png");

r0cha.setName("images/십이장기빨간말/r0cha.png");

Janggi.add(r0cha);

} **else** **if** (board[i][j] == r90cha) {

mal r90cha = **new** mal(i, j, "images/십이장기빨간말/r90cha.png");

r90cha.setName("images/십이장기빨간말/r90cha.png");

Janggi.add(r90cha);

} **else** **if** (board[i][j] == r0who) {

mal r0who = **new** mal(i, j, "images/십이장기빨간말/r0who.png");

r0who.setName("images/십이장기빨간말/r0who.png");

Janggi.add(r0who);

} **else** **if** (board[i][j] == r90who) {

mal r90who = **new** mal(i, j, "images/십이장기빨간말/r90who.png");

r90who.setName("images/십이장기빨간말/r90who.png");

Janggi.add(r90who);

}

**else** **if** (board[i][j] == g90king) {

mal g90king = **new** mal(i, j, "images/십이장기초록말/g90king.png");

g90king.setName("images/십이장기초록말/g90king.png");

Janggi.add(g90king);

} **else** **if** (board[i][j] == g0sang) {

mal g0sang = **new** mal(i, j, "images/십이장기초록말/g0sang.png");

g0sang.setName("images/십이장기초록말/g0sang.png");

Janggi.add(g0sang);

}

**else** **if** (board[i][j] == g90sang) {

mal g90sang = **new** mal(i, j, "images/십이장기초록말/g90sang.png");

g90sang.setName("images/십이장기초록말/g90sang.png");

Janggi.add(g90sang);

} **else** **if** (board[i][j] == g0ja) {

mal g0ja = **new** mal(i, j, "images/십이장기초록말/g0ja.png");

g0ja.setName("images/십이장기초록말/g0ja.png");

Janggi.add(g0ja);

} **else** **if** (board[i][j] == g90ja) {

mal g90ja = **new** mal(i, j, "images/십이장기초록말/g90ja.png");

g90ja.setName("images/십이장기초록말/g90ja.png");

Janggi.add(g90ja);

} **else** **if** (board[i][j] == g0cha) {

mal g0cha = **new** mal(i, j, "images/십이장기초록말/g0cha.png");

g0cha.setName("images/십이장기초록말/g0cha.png");

Janggi.add(g0cha);

} **else** **if** (board[i][j] == g90cha) {

mal g90cha = **new** mal(i, j, "images/십이장기초록말/g90cha.png");

g90cha.setName("images/십이장기초록말/g90cha.png");

Janggi.add(g90cha);

} **else** **if** (board[i][j] == g0who) {

mal g0who = **new** mal(i, j, "images/십이장기초록말/g0who.png");

g0who.setName("images/십이장기초록말/g0who.png");

Janggi.add(g0who);

} **else** **if** (board[i][j] == g90who) {

mal g90who = **new** mal(i, j, "images/십이장기초록말/g90who.png");

g90who.setName("images/십이장기초록말/g90who.png");

Janggi.add(g90who);

}

}

}

**return** Janggi;

}

**public** **void** checkJang() { //말의 움직임이 끝나거나 시간이 끝나거나 등등 자신의 차례가 끝나면 턴을 바꾸는 역할을 해준다.

**if** (*myTurn* == 0 && *k\_count2* != 1) { //차례가 0이고 왕카운트가 1이 아니면

*myTurn* = 1; //차례가 1로 바뀌고

information.setTurnIsChangeToTrue();

}

//왕이 상대 진영에서 한차례 버티면 게임이 승리된다. 이를 k\_count를 확인해서 승패를 결정.

**else** **if**(*myTurn* == 1 && *k\_count1* == 1) //player2의 차례에서 player1의 왕이 상대진영에 들어가서 한차례 버틴 상태면

{ //player1의 승리로 끝난다.

start = **false**; //게임 끝

JOptionPane.*showConfirmDialog*(**null**, "Player1의 승리", "안내", JOptionPane.***WARNING\_MESSAGE***);

information.IsStart = **false**; //

information.ResetTimer(); //타이머 리셋

information.IsStart=**false**;

}

**else** **if**(*myTurn* == 0 && *k\_count2* == 1)

//player1의 차례에서 player2의 왕이 상대진영에 들어가서 한차례 버틴 상태면

{ //player2의 승리로 끝난다.

start = **false**;

JOptionPane.*showConfirmDialog*(**null**,

"Player2의 승리", "안내", JOptionPane.***WARNING\_MESSAGE***);

information.IsStart = **false**;

information.ResetTimer();

information.IsStart=**false**;

}

**else**

{ //모든 경우의 수가 아니면 차례가 player1이고 왕카운트가 1이 아닌 경우로 판단

*myTurn* = 0; //player2의 턴으로 바뀐다.

information.setTurnIsChangeToTrue();

}

}

**public** **static** **void** changeTurn() //차례를 바꿔준다.

{

**if** (*myTurn* == 0)

{

*myTurn* = 1;

}

**else**

{

*myTurn* = 0;

}

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**new** Janggi();

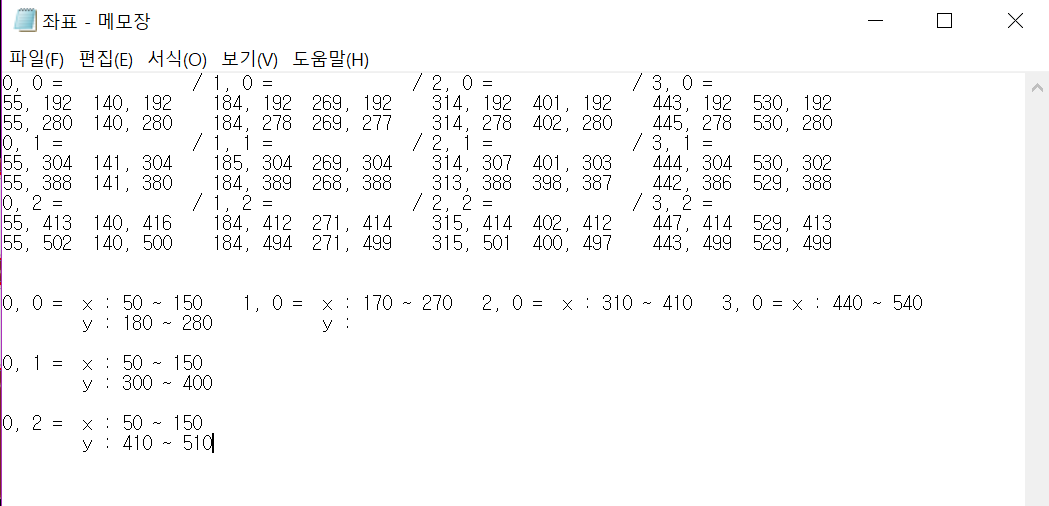
}

}

# 새로 알게된 기능

아래와 같은 기능들을 알게되었습니다.

* **자연스러운 화면 전환에 대한 기능들  
  -** JPanel1과 JPanel2로 전환하는 방법 중 setVisible()기능 활용법  
  **-** 특정 클래스가 끝나면 다음 클래스를 호출하여 불러오는 기능 활용법
* **이미지와 버튼을 활용한 인터페이스 구현  
  -** 버튼을 만든 후 버튼에 이미지를 넣으면 이미지 테두리 밖에 버튼 디자인이 남았는데, 그걸 삭제하는 방법을 배움(이미지 자체가 온전한 버튼이 됨)  
  **-** 버튼 자체를 이미지로 만든 후, 움직이는 장기판 말로 만드는 방법 **-** setbounds( x, y, width, height)를 활용하여 원하는 위치에 이미지 배열 & 버튼 지정법  
  **-** JPanel / JButton / JLabel 의 기능이 얼마나 자주 쓰이는지 그 위력을 알 수 있었음. 사용하는 빈도수와 동일하게 그만큼 각각의 기능들에 대해 명확하게 깨달을 수 있었음
* **게임 규칙에 대한 기능들  
  -** 각 말마다 움직일 수 있는 방향, 거리가 다른데 각 말의 움직임을 배열 수식 계산방법(오류를 방지하기 위해서 절대 위치를 전부 계산해서 만들어야했음.**)** 절대위치 계산결과는 아래와 같음.

  
**-** 장기말 중 ‘왕’이 상대방의 진영에 들어갔을 경우 나오는 경고창 제작 방법 & 이겼을 경우 나오는 경고창 제작 방법  
**-** 장기말 중 ‘자’가 상대방의 진영에 들어갔을 경우 ‘후’의 모습으로 뒤바뀌는 이미지 교환법  
**-** 장기말이 먹혔을 경우 옆의 정보창으로 이동하는 방법  
**-** 시간의 부족 때문에 구현하지 못한 장기판 말 재사용법은 panel을 하나 더 제작하여 이미지를 저장할 수 있게 했어야 했음.

* **2인 플레이 구현  
  -** 같은 화면을 공유하면서 플레이할 수 있게 하였음. 턴이 돌아가면 player 1(빨간 말)과 player 2(초록 말)순으로 말을 움직일 수 있음.

* **게임 플레이 시 플레이어가 알아야 할 정보를 구현   
  -** 시작 버튼을 누르면 게임이 시작되게 함. 그 이후 시작버튼은 ‘턴 넘기기’ 버튼으로 바뀜.  
  **-** 끝내기 버튼을 누르면 게임이 끝나고, 모든 시간이 초기화됨. **-** 플레이 총 시간을 화면에 표시함   
  **-** 현재 턴 남은 시간(30초에서 점점 줄어듦)을 화면에 표시함  
  **-** 턴이 돌아가게 만들어서 현재 플레이해야 하는 플레이어를 알 수 있음 **-** 턴이 돌아가게 만들어서 현재 플레이하는 사람의 말이 아닌 상대방의 말이 움직일 경우 말의 원래 위치로 다시 돌아오게 만듦  
  **-** 장기말이 먹혔을 경우 옆의 정보창으로 이동하게 하는 방법  
  **-** 먹힌 장기말이 정보창에서 서로 겹치지 않게 위치 조정(수식이용)

* **마우스 이벤트 기능   
  -** 장기판 말 마우스로 드래그하여 움직일 수 있게 만듦  
  **-** 마우스로 버튼 클릭하면 다음 화면으로의 전환 또는 특정 기능을 하게 만듦  
   (마우스가 접근했을 때 / 접근하지 않았을 때 / 클릭되었을 때 / 클릭되지 않았을 때) 4가지로 표현

* **스레드를 이용한 시간제어  
  -** timer클래스를 이용해 수식 계산을 통해서 시간에 맞게 시간정보를 표현함   
  **-** 만약 시간이 0보다 작아지면 턴이 바뀌게 했고, 총 시간은 계속해서 증가하게 만듦
* **상속의 무서움  
  -** 그동안셀 수 없을 정도로 많은 오류가 발생했고, 모든 문제는 상속 때문에 오는 것이었음. 수업 때 배운 ‘결합도가 높아질수록 수정해야하는 것들이 어마어마하게 발생한다’를 뼈저리게 알 수 있었던 계기였음. 상속은 최대한 배제하고 결과물을 만듦

# 프로젝트를 끝내면서

이선민 : 자바를 기초부터 처음 배우는 입장에서부터 작지만 처음으로 프로젝트라는 것을 완성하고 나니 정말 신기한 기분 밖에 들지 않습니다. 아직 많은 부분들이 부족한 프로그램이었지만 직접 응용이라는 것을 하며 프로그램에 대한 알고리즘 적인 부분들을 생각하며 게다가 직접 GUI까지 만들어야 한다는 것이 정말 매우 힘든 일인지 깨닫게 되었고 배워야할 점이 많다는 것을 다시 한 번 깨닫게 되었습니다. 그리고 팀으로 작업하면 일의 효율성과 보안점을 보충할 수 있다는 것이 프로그래밍 프로젝트에서 얼마나 중요한 지 많이 알게 되었습니다.

김채윤 : 구현하는 내내 오류의 지옥에서 살았습니다. 한 가지를 추가하면 다른 기능이 오류를 일으켰고, 프로젝트를 진행하면서 적어도 5일 이상은 밤을 샜던 것 같습니다. 시험기간과 겹쳐져서 심적부담이 심했지만 대학교 입학 후 처음 해보는 '진짜' 프로젝트였고, 비교적 자유로운 분위기에서 진행할 수 있어서 행운이었습니다. 올해 교수님 한 분께서 "올해 또는 내년 즈음 실력의 부재가 얼마나 뼈저리게 아픈지 알게되는 시점이 올 것"이라는 말씀을 하셨는데 이렇게 빨리 느낄줄은 미처 알지 못했습니다. 코딩은 배우고 배워도 늘지 않는 느낌이지만 꾸준히 하다보면 3주 가까이 걸린 오늘의 프로젝트를 이틀 만에 작업할 날이 오겠지요. 좋은 경험하게 해주셔서 감사합니다.

### 참고문헌, 참고자료 정리

<https://play.google.com/store/apps/details?id=org.cocos2dx.AnimalShogi&hl=ko>

<https://m.blog.naver.com/PostView.nhn?blogId=highkrs&logNo=220283709171&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F>

<https://stackoverflow.com/questions/4286759/how-to-show-hide-jpanels-in-a-jframe>

<https://blog.naver.com/woncaesar/70142537621>

<http://hoonjeck.tistory.com/entry/%EC%93%B0%EB%A0%88%EB%93%9CThread-%EC%A2%85%EB%A3%8C>