최종 결과 보고서

< 문제 해결 및 실습 : JAVA >

10조. No Pain No Game
19011566 박진우
18011582 유경진
19011476 이수진
19011460 이유재

2020. 12. 11

1) 제목 및 개요

제목 : **OTHELLO GAME**

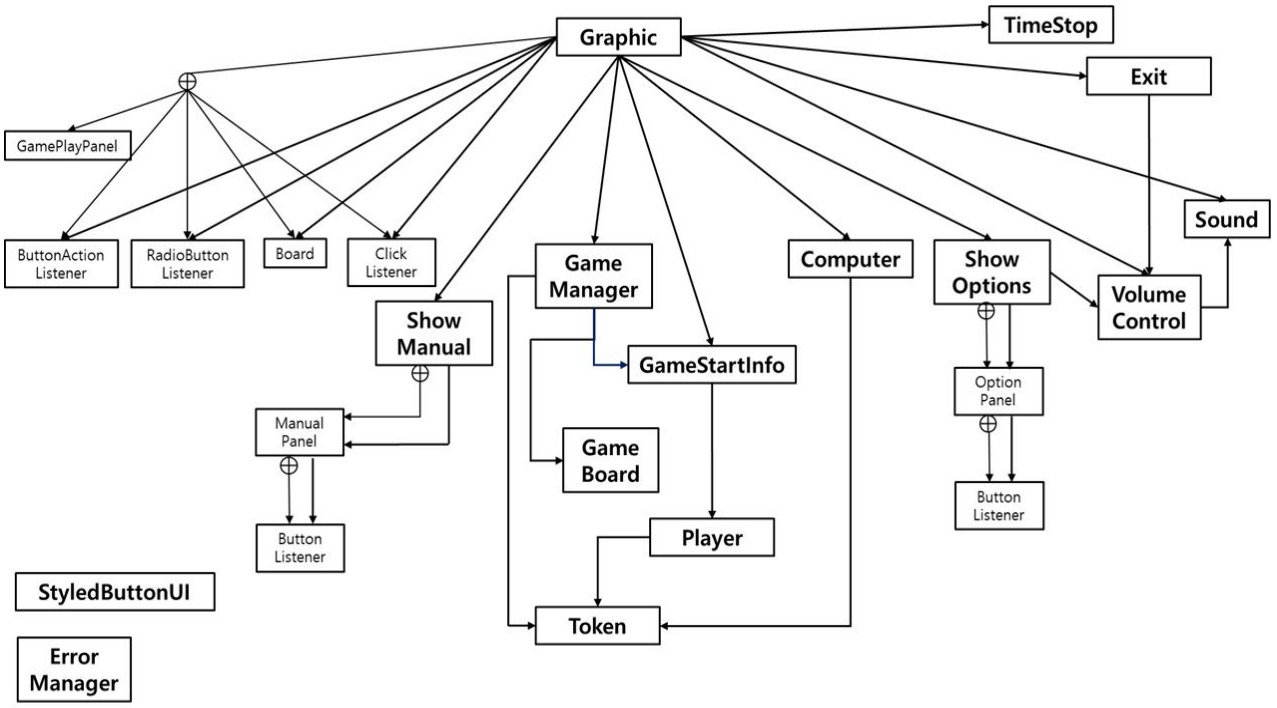
개요 : 오델로(Othello)는 보드 게임의 한 종류이며 리버시(Reversi)라고도 불린다. 가로 세로가 8칸인 8x8 게임판 위에서 한쪽은 검은색, 다른 한쪽은 흰색의 돌을 번갈아 놓으며 진행된다. 이때 어느 한 쪽의 돌이 다른 한쪽의 돌을 직선 상으로 감싸기만 하면 감싸진 돌들은 모두 감싼 돌의 색으로 교체되며 돌의 개수가 각 플레이어의 점수로 여겨진다. 이런 오델로 혹은 리버시라고 불리는 게임은 인터넷 검색으로 간편하게 즐길 수 있다. 다만 아쉬운 점이 몇 가지 있다.

첫째, 모드의 자율성이 없다. 즉, 컴퓨터와 대결하는 1인모드, 사용자와 대결하는 2인 모드의 선택이 사용자에게는 없다는 것이다. 이런 구성은 친구와 같이 플레이할 마음으로 접속한 사용자의 요구를 만족시키지 못할 뿐만 아니라, 사용자에게 실망을 줄 가능성도 존재한다.

둘째, 오델로에 익숙하지 않은 이들의 경우 자신의 돌을 위치시킬 수 있는 위치를 찾느라 주의를 집중해야 할 것이다. 그런데 이때 게임 자체적으로 현재 차례에서 가능한 위치들을 모두 알려준다면 사용자, 특히 초심자 입장에서는 부담이 줄어들어 최적의 판단을 가능케 해줄 것이다. 또한 이런 이유로 위치 표시 서비스가 오델로의 진입장벽을 낮추는데 큰 도움을 줄 것이다.

앞에서 파악한 문제점 2가지를 토대로 이를 보완하고자 우리의 프로젝트에서는 사용자에게 어떤 게임을 즐길 것인지에 대한 선택권을 부여하여 다양한 사용자의 요구를 충족시켰다. 또한, 현재 차례의 플레이어가 돌을 놓을 수 있는 위치를 알려주는 위치 표시 기능을 제공하여 사용자에게 주어지는 부담감을 최소화시켜 우리 프로그램의 진입장벽을 낮췄다. 이외에도 흥미로운 요소들을 삽입하여 사용자에게 재미를 제공하고자 하였다.

2) 전체 클래스 구조



주요 클래스는 볼드체로 작성하였다.

실선으로 연결되어있는 클래스는 해당 클래스 내부에서 멤버변수로 선언된 클래스이며, ⊕기호와 함께 실선으로 연결되어있는 것은 내부 클래스이다.

3) 개별 클래스 설명

접근 지정자를 표현하는 방식은 다음과 같다. 대시(-)는 private, 해시(#)는 protected, 더하기(+)는 public을 나타낸다. 멤버 변수의 자료형이나 함수의 반환형은 이름 뒤에 콜론(:)을 쓰고 표기하였다. 그리고 static은 밑줄을 그어 표현하였다.

1. StyledButtonUI

StyledButtonUI
None
+ StyledButtonUI() + installUI(JComponent) : void + paint(Graphics, Jcomponent) : void - paintBackground(Graphics, Jcomponent, int) : void

❑ **클래스 설명**

- **StyledButtonUI :** 버튼의 모양과 색상을 마음대로 커스텀 할 수 있다.

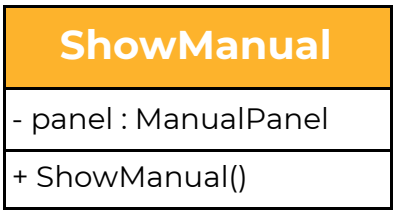
❑ **멤버 변수**

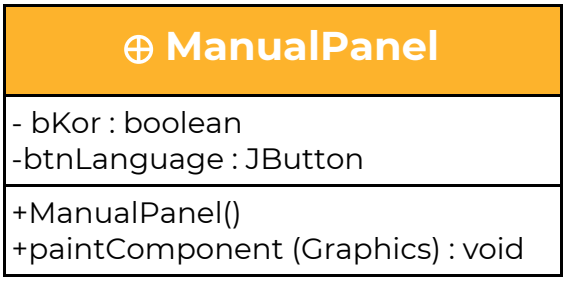
- 없음

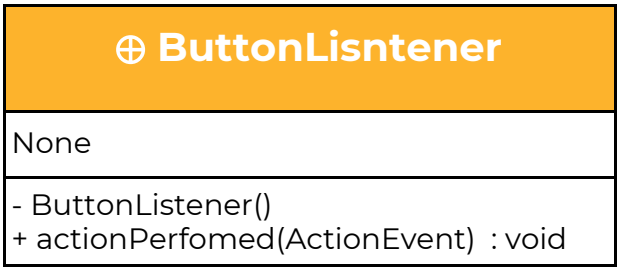
❑ **메서드**

- **installUI :** 버튼의 모양을 설정
- **paint :** 버튼의 press유무에 따라 모양을 바꿀 수 있도록 painBackground로 매개변수 값을 반환
- **paintBackground :** 둥근 직사각형 형태의 버튼 생성

2. ShowManual







❑ **클래스 설명**

- **ShowManual :** 게임 방법과 게임 승리 전략이 적힌 매뉴얼을 호출한다.
- **ManualPanel :** 두 가지 언어로 적힌 매뉴얼 패널을 생성한다.
- **ButtonListener :** 버튼에 관한 이벤트와 해당 이벤트에서 수행하는 메소드와 작업을 나타낸다.

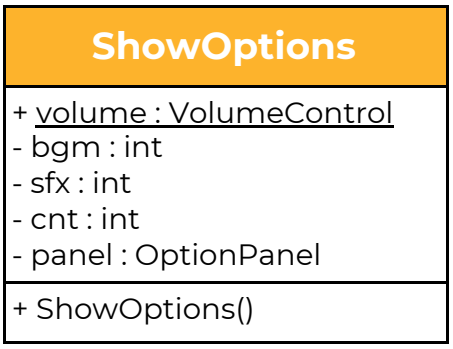
❑ **멤버 변수**

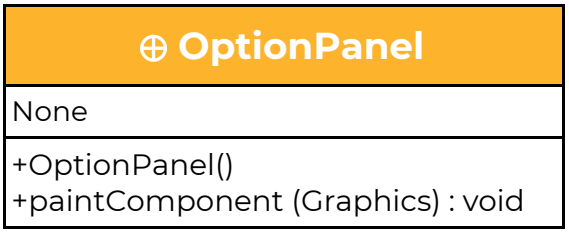
- **panel :** 매뉴얼에 관한 정보가 담긴 패널

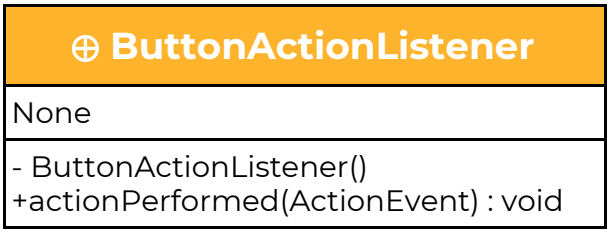
❑ **메서드**

- **ShowManual :** 매뉴얼에 관한 정보가 담긴 ManualPanel 클래스를 생성

3. ShowOptions







❑ **클래스 설명**

- **ShowOptions :** 게임 화면에서 옵션 버튼을 클릭했을 때 배경 음악과 효과음의 음량을 조절 할 수 있다.
- **OptionPanel :** 내부 클래스로 패널을 생성한다.
- **ButtonActionListener :** 버튼에 관한 이벤트와 해당 이벤트에서 수행하는 메소드와 작업을 나타낸다.

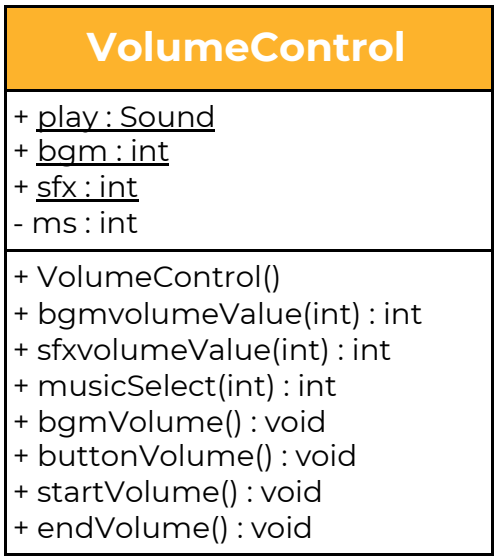
❑ **멤버 변수**

- **volume :** 볼륨 값이 변경되면 그 값에 맞는 배경음악과 효과음을 출력
- **bgm :** 현재 배경음악의 볼륨 값을 저장
- **sfx :** 현재 효과음의 볼륨 값을 저장
- **cnt :** OK가 몇 번 눌렸는지 그 값을 저장
- **panel :** 패널 선언 및 생성

❑ **메서드**

- **ShowOptions :** 프레임 안의 패널을 설정
- **OptionPanel :** 패널 안에 레이블과 슬라이더, 버튼을 구성
- **paintComponent :** 패널의 배경을 설정
- **actionPerformed :** 버튼을 클릭했을 때 이벤트를 처리

4. VolumeControl



❑ 클래스 설명

- **VolumeControl :** 배경 음악과 효과음을 재생해주고 음량이 변경되면 그에 맞게 변경된 배경 음악과 효과음을 출력해준다.

❑ 멤버 변수

- **play :** 배경 음악과 효과음을 재생
- **bgm :** 배경 음악의 볼륨 값 저장
- **sfx :** 효과음의 볼륨 값 저장
- **ms :** 배경 음악을 바꾸기 위한 변수

❑ 메서드

- **bgmvolumeValue :** 배경 음악의 볼륨 값을 저장하고 반환
- **sfxvolumeValue :** 효과음의 볼륨 값을 저장하고 반환
- **musicSelect :** 배경 음악을 바꾸기 위해 OK가 몇 번 눌렸는지 값을 저장하고 반환
- **bgmVolume :** 반환된 볼륨 값에 따라 배경 음악을 재생
- **buttonVolume :** 반환된 볼륨 값에 따라 버튼 효과음을 재생
- **startVolume :** 반환된 볼륨 값에 따라 게임 시작 효과음을 재생
- **endVolume :** 반환된 볼륨 값에 따라 게임 엔딩 효과음을 재생

5. Sound

Sound
+ clip : Clip
+ Sound() + playBackgroundSound1() : void + stopBackgroundSound() : void + buttonSpecialEffect() : void + startSpecialEffect() : void + endSpecialEffect() : void + volume_playBackgroundSound1(int) : void + volume_playBackgroundSound2(int) : void + volume_playBackgroundSound3(int) : void + volume_buttonSpecialEffect(int) : void + volume_startSpecialEffect(int) : void + volume_endSpecialEffect(int) : void

❑ **클래스 설명**

- **Sound :** 음악 클립을 가져와서 배경 음악 또는 효과음을 재생해준다.

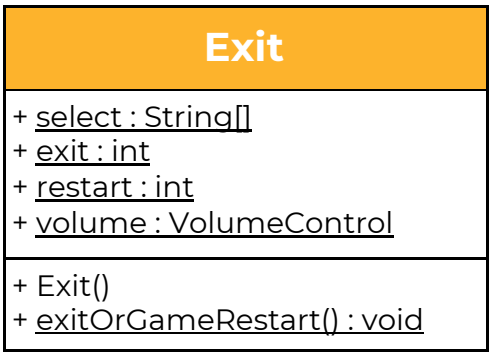
❑ **멤버 변수**

- **clip :** 음악 파일을 재생하기 위한 클립

❑ **메서드**

- **playBackGroundSound1 :** 프로그램을 실행시킬 때 배경 음악을 재생
- **stopBackGroundSound :** 배경 음악을 정지
- **buttonSpecialEffect :** 버튼을 클릭할 때 재생되는 효과음
- **startSpecialEffect :** 게임이 시작될 때 재생되는 효과음
- **endSpecialEffect :** 게임이 끝날 때 재생되는 효과음
- **volume_playBackgroundSound1 :** 바뀐 볼륨 값에 따라 첫번째 배경 음악을 재생
- **volume_playBackgroundSound2 :** 바뀐 볼륨 값에 따라 두번째 배경 음악을 재생
- **volume_playBackgroundSound3 :** 바뀐 볼륨 값에 따라 세번째 배경 음악을 재생
- **volume_buttonSpecialEffect :** 바뀐 볼륨 값에 따라 버튼 효과음을 재생
- **volume_startSpecialEffect :** 바뀐 볼륨 값에 따라 게임 시작 효과음을 재생
- **volume_endSpecialEffect :** 바뀐 볼륨 값에 따라 게임 엔딩 효과음을 재생

6. Exit



❑ 클래스 설명

- **Exit :** 게임 도중 게임을 재시작 하거나 나갈 수 있게 해준다.

❑ 멤버 변수

- **select :** 버튼에 들어갈 String 배열
- **exit :** 게임을 나가는지 판단하는 변수
- **restart :** 게임을 재시작하는지 판단하는 변수
- **volume :** VolumeControl 객체

❑ 메서드

- **exitOrGameRestart :** 게임을 나가거나 재시작 할건지 묻는 옵션 창을 띄우고 버튼 이벤트에 맞는 상황을 처리

7. Game Manager

GameManager
- currentBoard : GameBoard - info: GameStartInfo + currentTurnToken : Token + successiveCntSkip : int
+ GameManager() + setGameInfo(GameStartInfo) : void + setCurrentToken(Token) : void + isRange(int, int) : boolean + findPossiblePosition(GameBoard, int) : boolean[][] + isPossible(Point) : boolean + canSkip() : boolean + isFull() : boolean + getGameCurrentBoard() : GameBoard

❑ 클래스 설명

- **GameManager :** 게임 진행시 플레이어와 컴퓨터에게 도움을 주는 클래스이다.

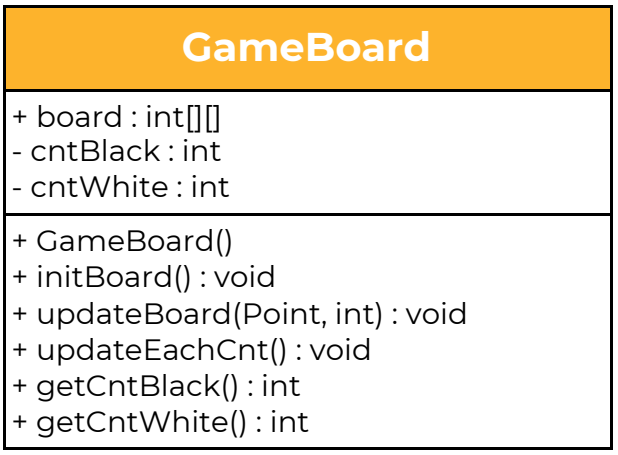
❑ 멤버 변수

- **currentBoard :** 게임 감독을 위해 현재 보드판 객체를 저장하는 변수
- **info :** 게임 설정정보를 담아두기 위한 변수
- **currentTurnToken :** 현재 차례를 분별하기 위한 변수
- **successiveCntSkip :** 연속적으로 skip된 횟수를 저장하기 위한 변수

❑ 메서드

- **GameManager :** GameManager객체를 생성하는 생성자
- **setGameInfo :** GameManager의 info를 설정
- **setCurrentToken :** currentTurnToken을 설정
- **isRange :** 지정한 위치 인덱스가 게임 보드 내부인지를 판단
- **findPossiblePosition :** 현재의 보드를 나타내는 객체를 매개변수로 받아 해당 상태에서 현재 차례의 토큰을 위치시킬 수 있는지 아닌지를 판별하여 가능한 위치들에 대해서는 2차원 배열에 boolean 값으로 저장하여 반환
- **isPossible :** 지정한 위치의 인덱스가 토큰을 위치할 수 있는 지를 반환
- **canSkip :** 스킵 가능 여부 반환
- **isFull :** 게임판 모든 칸에 토큰이 놓여있는지 여부 반환
- **getGameCurrentBoard :** GameBoard 객체인 currentBoard를 반환

8. GameBoard



❑ 클래스 설명

- **GameBoard :** 게임 보드를 구현한 클래스이다. 현재 게임판의 상태에 대한 정보를 가지며 게임판 위에서 변화에 맞춰 현재 게임판 상태를 엡데이트해준다.

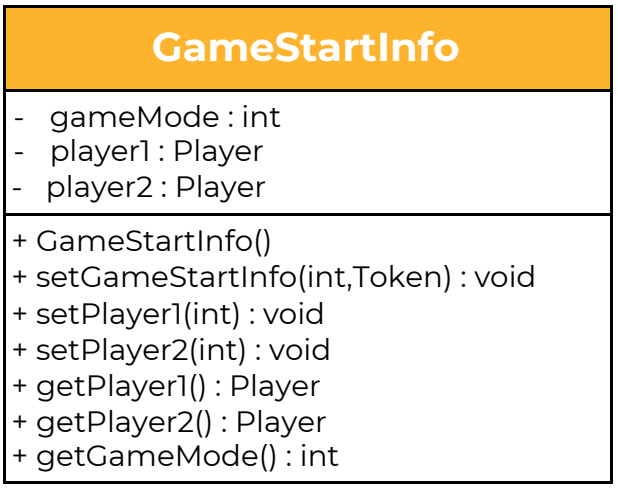
❑ 멤버 변수

- **board :** 2차원 배열 형태로, 검은색 토큰이 있는 곳은 1, 흰색 토큰이 있는 곳은 -1로 하여 게임판의 상태를 저장.
- **cntBlack :** board에 있는 검은색 토큰의 수를 저장
- **cntWhite :** board에 있는 흰색 토큰의 수를 저장

❑ 메서드

- **GameBoard :** GameBoard 객체를 생성하는 생성자
- **initBoard :** board를 게임 초기상태로 초기화
- **updateBoard :** 토큰을 위치함으로써 변화하는 주변 토큰에 대해서 업데이트
- **updateEachCnt :** board가 업데이트됨에 따라 cntBlack과 cntWhite를 업데이트
- **getCntBlack :** cntBlack을 반환
- **getCntWhite :** cntWhite를 반환

9. GameStartInfo



❑ **클래스 설명**

- **GameStartInfo :** 게임 시작에 필요한 정보를 저장한다.

❑ **멤버 변수**

- **gameMode :** 게임모드 정보를 저장
- **player1 :** 플레이어1 의 정보를 저장
- **player2 :** 플레이어2 의 정보를 저장

❑ **메서드**

- **GameStartInfo :** 게임모드와 플레이어를 생성하는 생성자
- **setGameStartInfo :** 게임모드와 플레이어의 정보를 설정
- **setPlayer1 :** Player1의 정보를 설정
- **setPlayer2 :** Player2의 정보를 설정
- **getPlayer1 :** Player1의 정보를 반환
- **getPlayer2 :** Player2의 정보를 반환
- **getGameMode :** 게임모드를 반환

10. Player

Player
- myToken : Token - myCharacter : int
+ Player(Token) + Player(int) + Player(Token, int) + setMyToken(Token) : void + setMyCharacter(int) : void + getMyToken() : Token + getMyCharacter() : int + convertPosToIdx(Point) : Point

❑ 클래스 설명

- **Player :** 플레이어에 대한 정보를 담는다. 이 플레이어가 어떤 색을 가졌는지, 캐릭터는 무엇인지에 대한 정보를 담는다.

❑ 멤버 변수

- **myToken :** 토큰의 정보를 저장
- **myCharacter :** 캐릭터의 정보를 저장

❑ 메서드

- **Player(Token) :** 토큰의 정보를 인자로 받아 Player을 생성하는 생성자
- **Player(int) :** 캐릭터의 정보를 인자로 받아 Player을 생성하는 생성자
- **Player(Token, int) :** 토큰의 정보와 캐릭터의 정보를 인자로 받아 Player을 생성하는 생성자
- **setMyToken :** 토큰의 정보를 설정
- **setMyCharacter :** 캐릭터의 정보를 설정
- **getMyToken :** 토큰의 정보를 반환
- **getMyCharacter :** 캐릭터의 정보를 반환
- **convertPosToIdx :** 좌표를 보드의 인덱스로 변환

11. Token

Token
- State : int
+ Token(int) + getState() : int

❑ **클래스 설명**

- **Token :** 게임 보드위에 올라가는 토큰의 정보를 담는다. state가 1이면 검은색, -1이면 흰색이다.

❑ **멤버 변수**

- **State :** 토큰의 정보를 저장

❑ **메서드**

- **Token :** 토큰의 정보를 설정하는 생성자
- **getState :** 토큰의 정보를 반환

12. Computer

Computer
- myPoint : Point - myToken : Token
+ Computer(int) + getMyToken() : Token - copyElement(int [][], int [][]) : void - sol(GameBoard, int, int, int) : int + searchBestPosition(GameBoard, int, int, int) : Point

❑ **클래스 설명**

- **Computer :** 플레이어와 대결을 하는 클래스이며 최적의 위치를 탐색한다.

❑ **멤버 변수**

- **myPoint :** 컴퓨터가 토큰을 놓을 위치 정보를 저장
- **myToken :** 컴퓨터에게 부여된 토큰에 대한 정보를 저장

❑ **메서드**

- **Computer :** Computer객체를 생성하는 생성자
- **getMyToken :** 컴퓨터가 가지고 있는 토큰을 반환
- **copyElement :** 두 개의 int형 2차원배열을 매개변수로 받아 한 배열의 원소를 다른 하나의 배열에 복사
- **sol :** minimax알고리즘을 활용하여 myPoint에 적절한 위치를 저장
- **searchBestPosition :** sol를 호출하며 myPoint를 반환

13. TimeStop

TimeStop
None
+ TimeStop() + run() : void

❑ **클래스 설명**

- **TimeStop :** 일정 시간 동안 Thread를 멈춘다.

❑ **멤버 변수**

- 없음

❑ **메서드**

- **run :** 일정 시간 Thread를 중지

14. Error Manager

ErrorManager
None
+ ErrorManager() + positionError(JFrame) : void + skipMsg(JFrame, int) : void

❑ **클래스 설명**

- **ErrorManager :** 게임 도중 오류 메시지 및 알림 메시지를 띄운다.

❑ **멤버 변수**

- 없음

❑ **메서드**

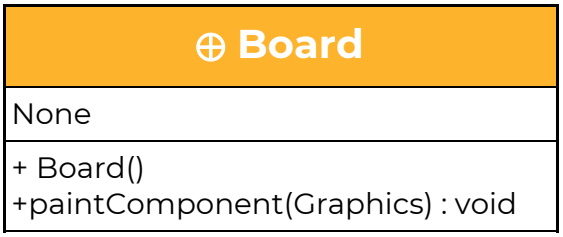
- **postionError :** 토큰을 놓을 수 없는 곳에 클릭했을 때 알려주는 메시지 출력
- **skipMsg :** 턴이 스킵되어 상대방에게 넘어갔을 때 알려주는 메시지 출력

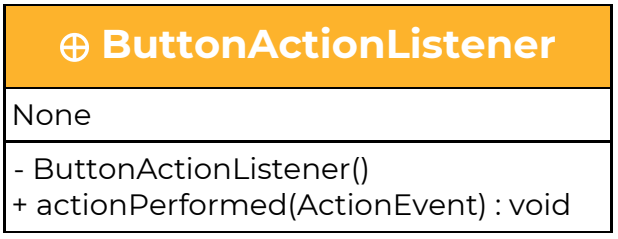
15. Graphic

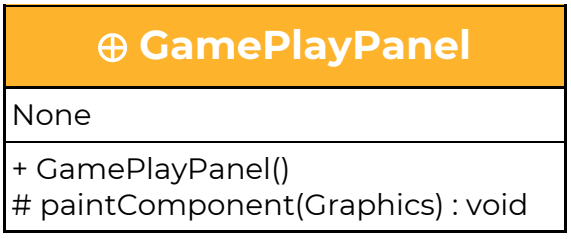
Graphic	
+ mainFrame : JFrame + winFrame : JFrame + initPanel : JPanel + gameSettingPanel : JPanel + characterSelectPanel : JPanel + winPanel : JPanel + gamePlayPanel : JPanel + gameInfo : GameStartInfo - sf : ShowManual - option : ShowOptions - gm : GameManager - com : Computer - posPossible : Boolean[][] - boardPanel : Board - volume : VolumeControl - play : Sound - ts : TimeStop - score_1 : int - score_2 : int - exitOrRestart : Exit	+ Graphic() - mySetPanel(JPanel) : void - getLabelLogo() : JLabel - getButtonHome() : JButton - setInitPanel() : void - setGameSettingPanel() : void - setCharacterSelectPanel() : void - setGamePlayPanel() : void - setWinPanel() : void - showInitPage() : void - showGameSettingPage() : void - showCharacterSelectPanel() : void - showGamePlayPage() : void - showWinPage() : void - updateGameBoardG() : void - updateColorInfo(int) : void - updateScore() : void + restartGame(int) : void + main(String[]) : void

⊕ ClickListener
None
- ClickListener() + mouseClicked(MouseEvent) :void + mousePressed(MouseEvent) :void + mouseReleased(MouseEvent) : void + mouseEntered(MouseEvent) : void + mouseExited(MouseEvent): void

⊕ RadioButtonListener
None
- RadioButtonListener() + actionPerformed(ActionEvent) :void







❑ **클래스 설명**

- **Graphic :** 프로그램 실행 시 시각적으로 보여지는 모든 부분을 표현하기 위한 메서드와 멤버 변수를 갖고 있다.
- **ClickListener :** 마우스의 입력에 관한 이벤트와 해당 이벤트에서 수행하는 메소드와 작업을 나타낸다.
- **RadioButtonListener :** 라디오버튼에 관한 이벤트와 해당 이벤트에서 수행하는 메소드와 작업을 나타낸다.
- **Board :** 게임화면 내 보드판 이미지를 보여준다.
- **ButtonActionListener :** 버튼에 관한 이벤트와 해당 이벤트에서 수행하는 메소드와 작업을 나타낸다.
- **GamePlayPanel :** 게임화면의 배경화면을 설정한다.

❑ **멤버 변수**

- **mainFrame :** 게임이 실행될 주된 프레임
- **winFrame :** 게임이 끝났을 때 게임 결과를 알려주는 패널을 포함한 프레임
- **initPanel :** 프로그램을 시작 했을 때 보여지는 초기 패널
- **gameSettingPanel :** 게임 설정을 하는 화면을 보여주는 패널
- **characterSelectPanel :** 캐릭터 선택화면을 보여주는 패널
- **winPanel :** 게임 결과를 알려주는 패널
- **gamePlayPanel :** 게임 플레이시 게임 화면을 보여주는 패널
- **gameInfo :** 게임 설정 정보를 담아두는 객체

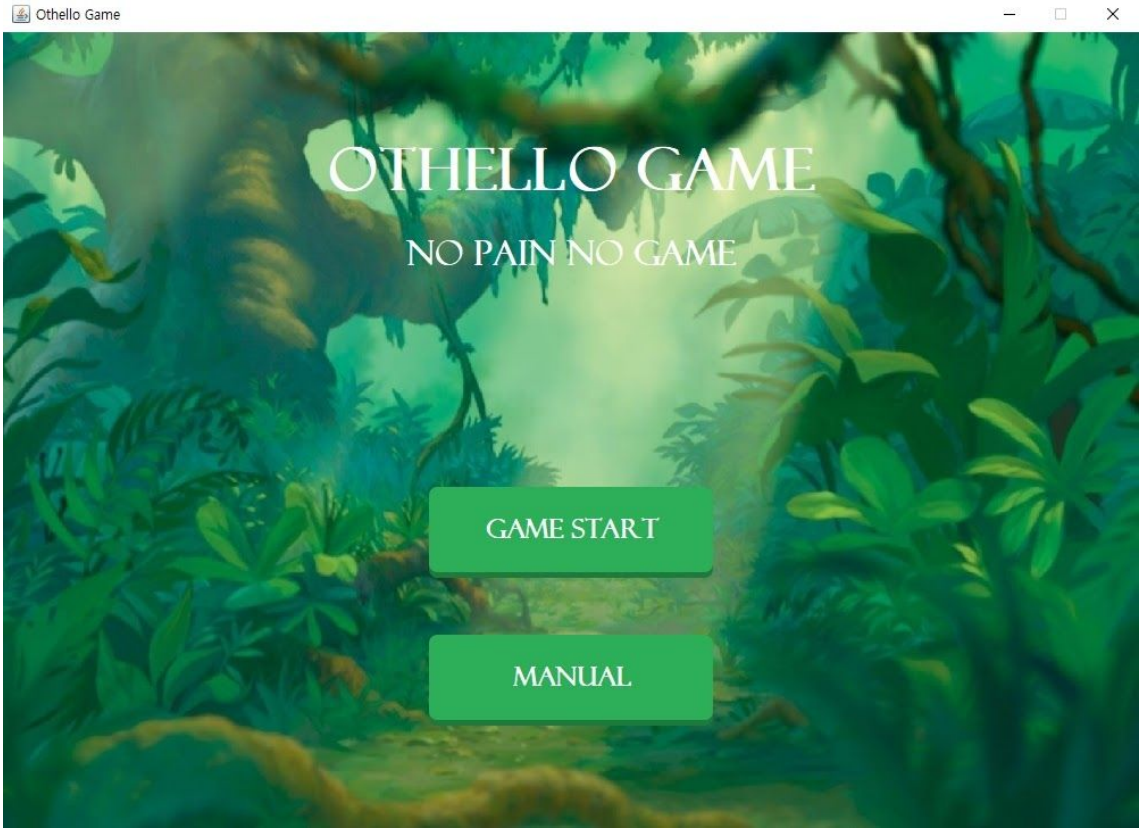
- **sf :** ShowManual 객체
- **option :** 게임 도중 옵션 창을 띄우기 위한 변수
- **gm :** GameManager 객체
- **com :** Computer 객체
- **posPossible :** 현재 차례의 객체가 위치가능한 곳을 저장하는 2차원 boolean 배열
- **boardPanel :** 게임판을 보여주는 패널
- **volume :** VolumeControl 객체
- **play :** Sound 객체
- **ts :** TimeStop 객체
- **score_1 :** 1번 플레이어에 대한 점수
- **score_2 :** 2번 플레이어에 대한 점수
- **exitOrRestart :** 게임 도중 종료 창을 띄우기 위한 변수

❑ **메서드**

- **Graphic :** mainFrame과 패널, 이미지 파일 경로 선언을 포함하는 생성자
- **mySetPanel :** Panel에서 공통적으로 설정해야 되는 것들을 설정
- **getLabelLogo :** 게임의 로고 레이블 생성 및 반환
- **getButtonHome :** Home Button을 생성 및 반환
- **setInitPanel :** initPanel을 설정
- **setGameSettingPanel :** 게임 설정 화면을 보여주는 패널을 설정
- **setCharacterSelectPanel :** 캐릭터 선택 화면을 보여주는 패널을 설정
- **setGamePlayPanel :** 게임 플레이 화면을 보여주는 패널을 설정
- **setWinPanel :** 게임 결과를 보여주는 패널을 설정
- **showInitPage :** mainFrame의 내부 패널을 게임 초기 패널로 설정
- **showGameSettingPage :** mainFrame의 내부 패널을 게임 설정 패널로 설정
- **showCharacterSelectPanel :** mainFrame의 내부 패널을 캐릭터 선택 패널로 설정
- **showGamePlayPage :** mainFrame의 내부 패널을 게임 플레이 패널로 설정
- **showWinPage :** winFrame에 게임결과를 넣고 winFrame을 생성
- **updateGameBoardG :** 게임화면 내 게임보드 이미지 업데이트
- **updateColorInfo :** 현재 누구 차례인지에대 알려주는 txtFiled내용을 업데이트
- **updateScore :** 현재 점수를 업데이트
- **restartGame :** 게임을 다시 시작하기 위해 showGameSettingPage 호출

4) UI 기능 정의 내용 및 실행 화면

1. 초기 화면



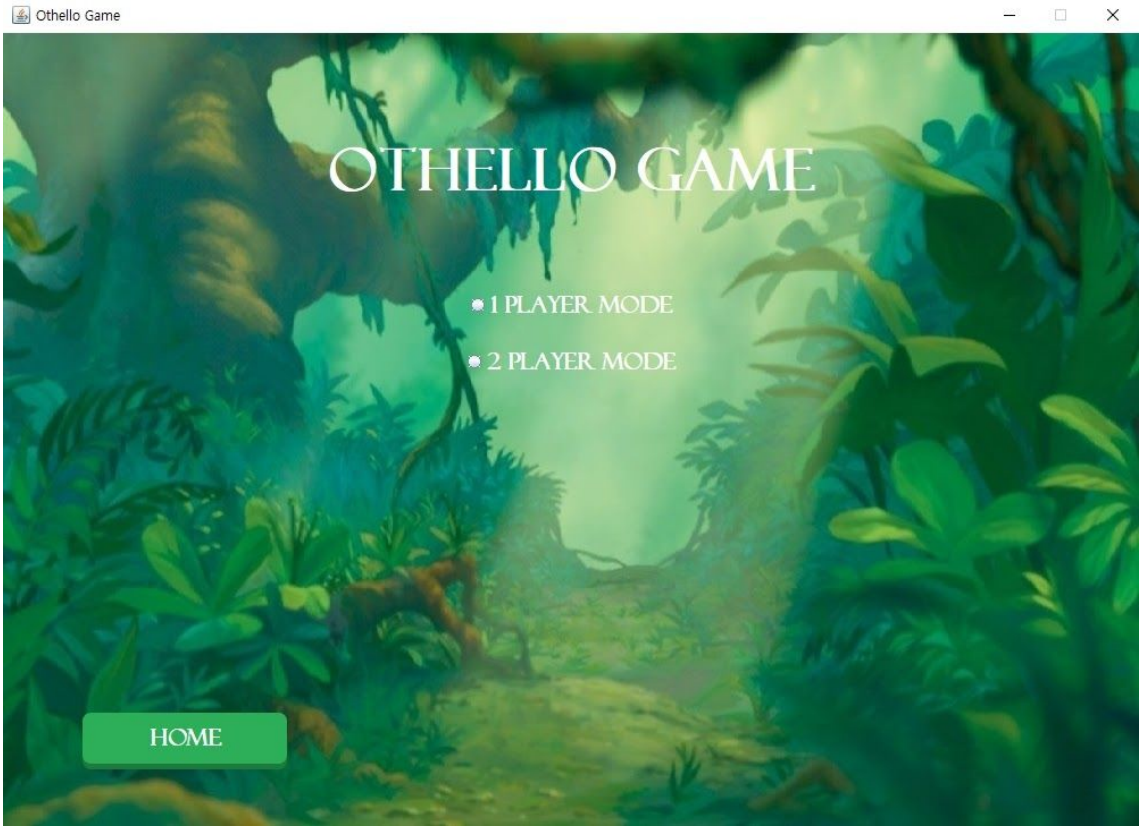
❑ **UI 화면 설명 / 실행화면 설명**

- 프로그램을 실행했을 때 제일 먼저 보여지는 화면이다.

❑ **기능**

- 게임 시작 : ‘GAME START’ 버튼을 누르면 게임 모드를 선택하는 화면으로 넘어간다.
- 매뉴얼 : ‘MANUAL’ 버튼을 누르면 게임 방법과 승리 전략을 한국어와 영어, 두가지 버전으로 확인할 수 있다.

2. 게임 모드 선택



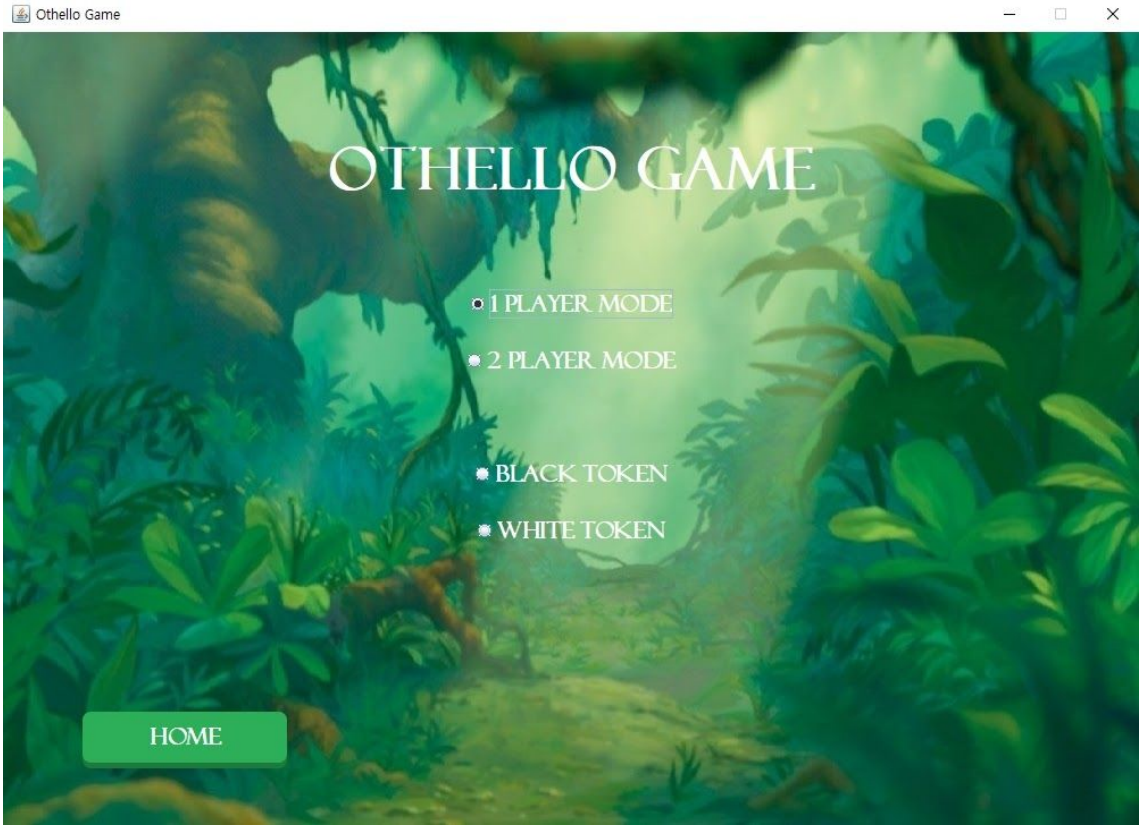
❑ **UI 화면 설명 / 실행화면 설명**

- 1인 플레이로 할지 2인 플레이로 할지 게임 모드를 선택하는 화면이다.

❑ **기능**

- 게임 모드 선택 : 1인 플레이를 하고 싶다면 '1 PLAYER MODE' 를 클릭하고, 2인으로 게임을 플레이하고 싶다면 '2 PLAYER MODE' 를 클릭하면 된다.
- 홈 버튼 : 'HOME' 버튼을 누르면 초기 화면으로 돌아간다.

2 - 1. 1인 모드



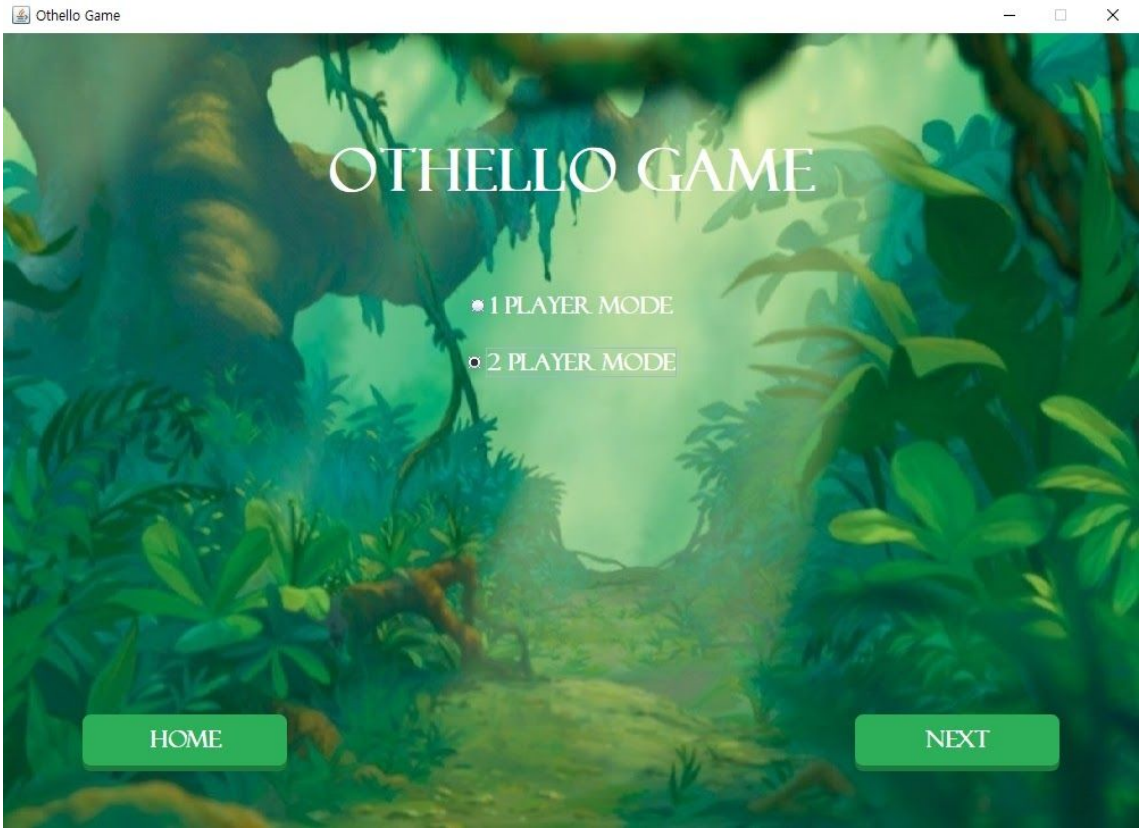
❑ **UI 화면 설명 / 실행화면 설명**

- '1 PLAYER MODE' 를 클릭했을 때 보여지는 화면이다.

❑ **기능**

- 검정색 토큰 : 'BLACK TOKEN' 을 클릭하면 게임이 시작되었을 때 먼저 토큰을 놓게 된다.
- 흰색 토큰 : 'WHITE TOKEN' 을 클릭하면 게임이 시작되었을 때 컴퓨터가 먼저 토큰을 놓게 되고 다음 차례에 토큰을 놓을 수 있다.
- 홈 버튼 : 'HOME' 버튼을 누르면 초기 화면으로 돌아간다.

2 - 2. 2인 모드



❑ **UI 화면 설명 / 실행화면 설명**

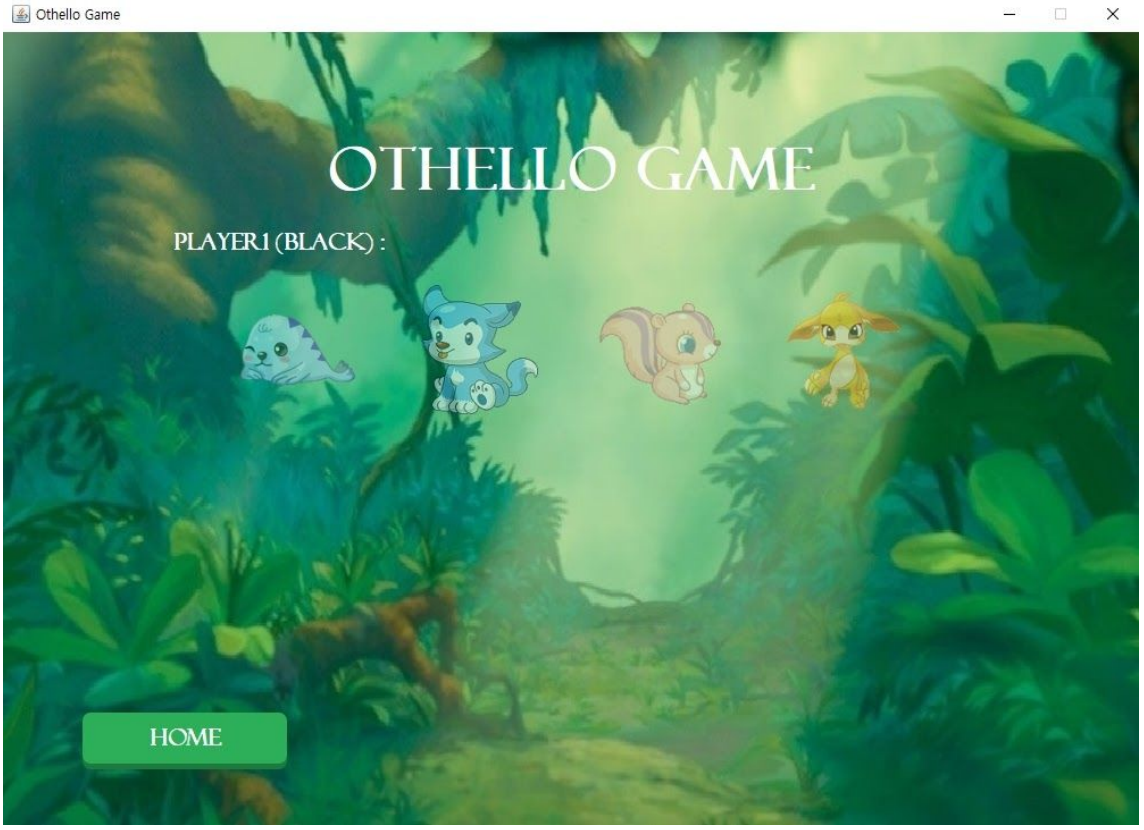
- '2 PLAYER MODE' 를 클릭했을 때 보여지는 화면이다.

❑ **기능**

- 다음 버튼 : 'NEXT' 버튼을 누르면 캐릭터를 설정할 수 있는 화면으로 넘어간다.
- 홈 버튼 : 'HOME' 버튼을 누르면 초기 화면으로 돌아간다.

3. 게임 캐릭터 선택

3 - 1. 1인 모드



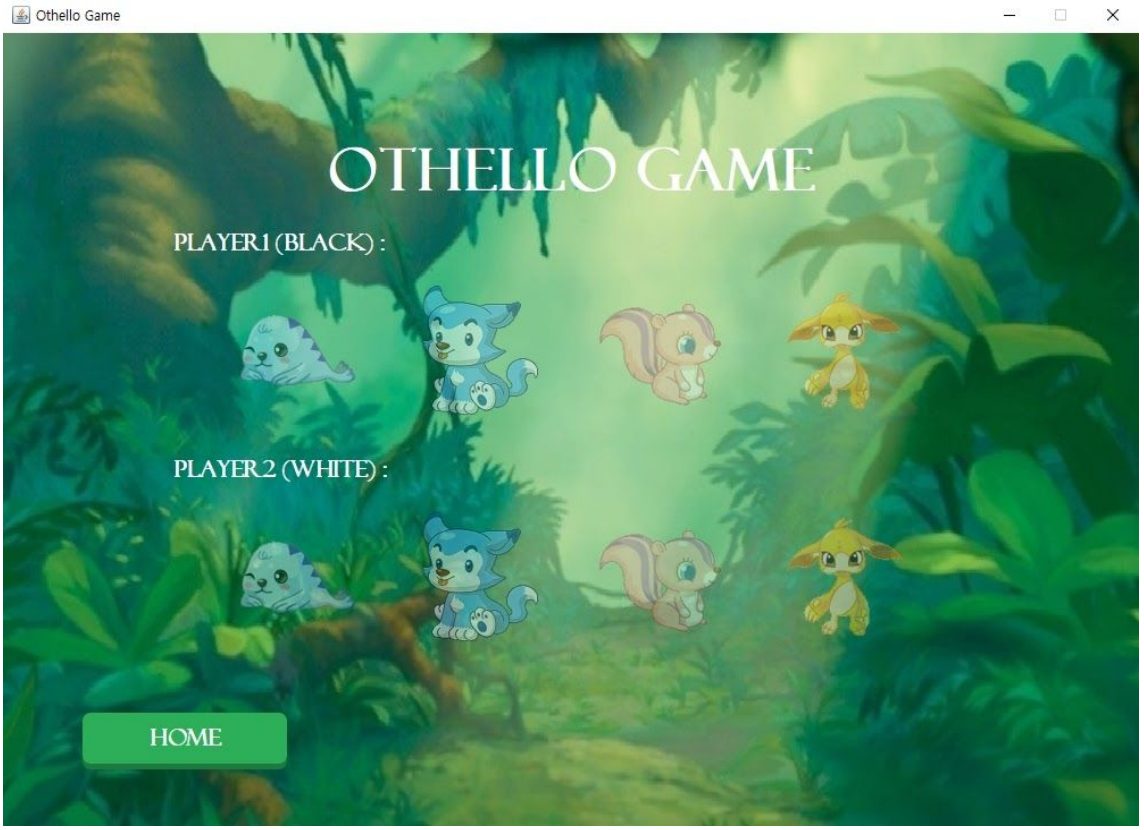
❑ **UI 화면 설명 / 실행화면 설명**

- '1 PLAYER MODE' 일 때 캐릭터를 선택할 수 있는 화면이다.

❑ **기능**

- 캐릭터 선택 : 캐릭터 위에 플레이어와 토큰의 색이 텍스트로 보여지고 플레이어는 4개의 캐릭터 중, 원하는 캐릭터를 선택할 수 있다. 흰색 토큰을 선택했을 경우에는 *PLAYER1(WHITE):* 로 보여진다.
- 홈 버튼 : 'HOME' 버튼을 누르면 초기 화면으로 돌아간다.

3 - 2. 2인 모드



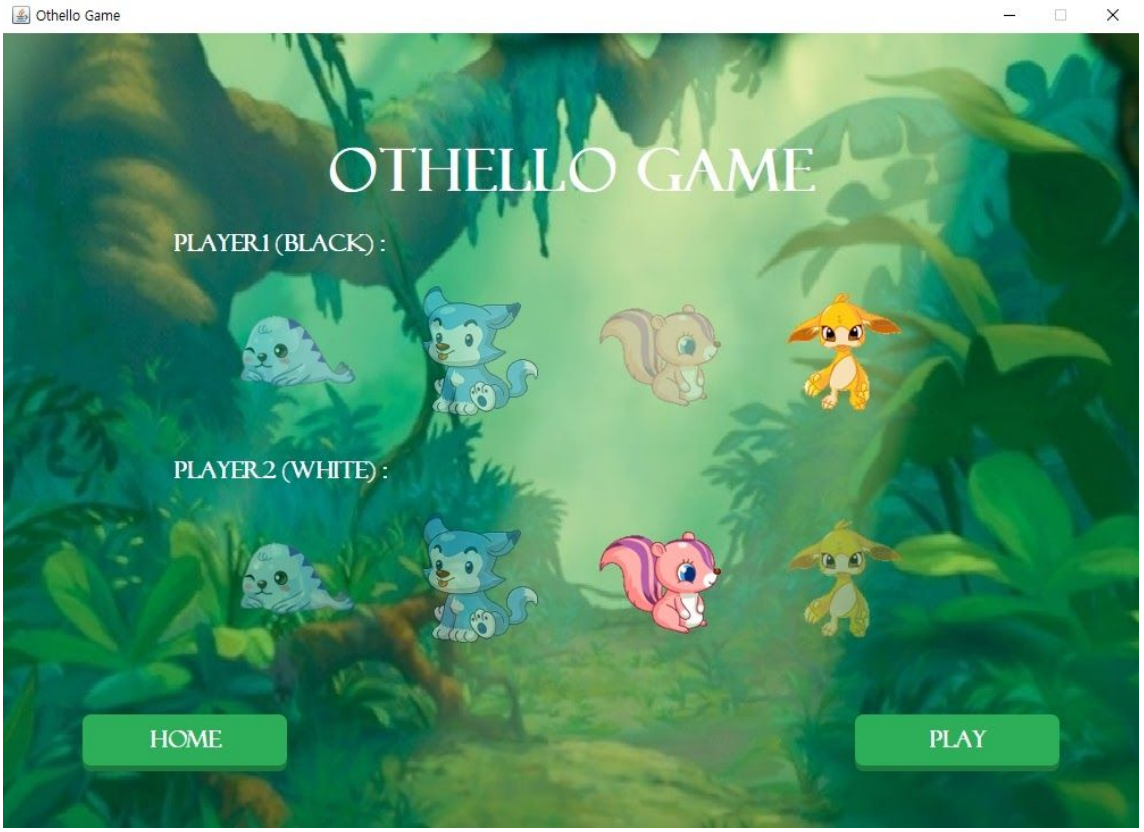
❑ **UI 화면 설명 / 실행화면 설명**

- '2 PLAYER MODE' 일 때 캐릭터를 선택할 수 있는 화면이다.

❑ **기능**

- 캐릭터 선택 (BLACK) : 플레이어가 4개의 캐릭터 중, 원하는 캐릭터를 선택할 수 있고 게임화면에서는 캐릭터가 왼쪽에 나타나며 검정색 토큰이므로 토큰을 먼저 놓게 된다.
- 캐릭터 선택 (WHITE) : 플레이어가 4개의 캐릭터 중, 원하는 캐릭터를 선택할 수 있고 게임화면에서는 캐릭터가 오른쪽에 나타나며 흰색 토큰이므로 'PLAYER 1' 이 먼저 토큰을 놓은 다음 토큰을 놓게 된다.
- 홈 버튼 : 'HOME' 버튼을 누르면 초기 화면으로 돌아간다.

3 - 3. 캐릭터 선택 화면



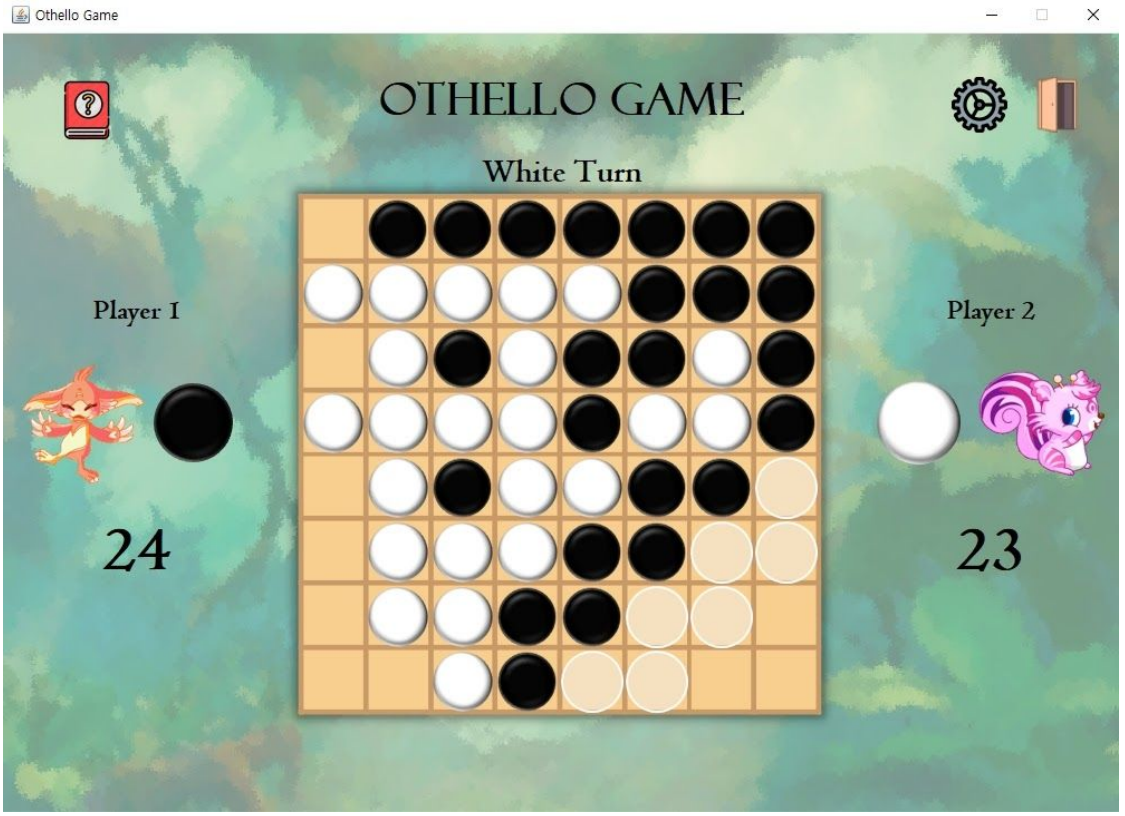
❑ **UI 화면 설명 / 실행화면 설명**

- 캐릭터를 선택했을 때 (클릭했을 때) 보여지는 화면이다.

❑ **기능**

- 선택한 캐릭터 : 플레이어가 캐릭터를 선택하게 되면 투명하게 보이던 캐릭터가 선명하게 보인다. 플레이어의 캐릭터를 모두 선택하면 플레이 버튼이 나타나게 된다.
- 플레이 버튼 : ‘PLAY’ 버튼을 누르면 게임 화면으로 넘어간다.
- 홈 버튼 : ‘HOME’ 버튼을 누르면 초기 화면으로 돌아간다.

4. 게임 화면



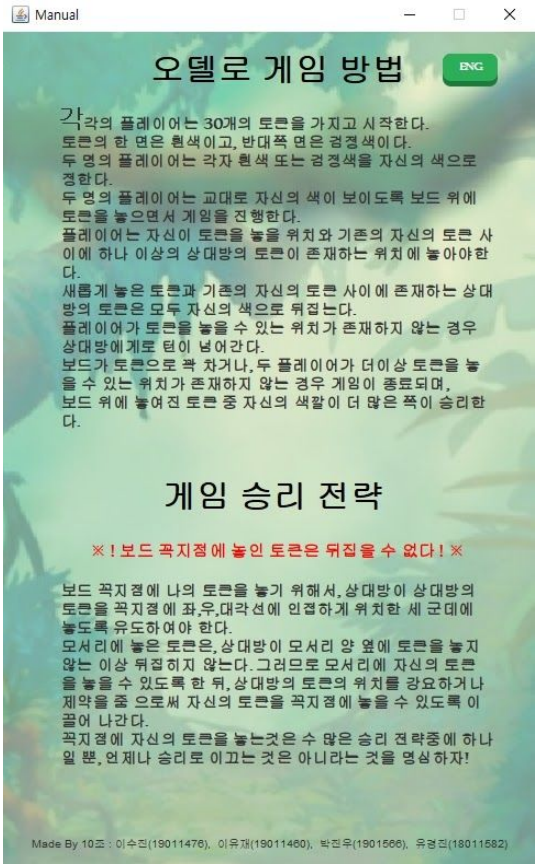
❑ **UI 화면 설명 / 실행화면 설명**

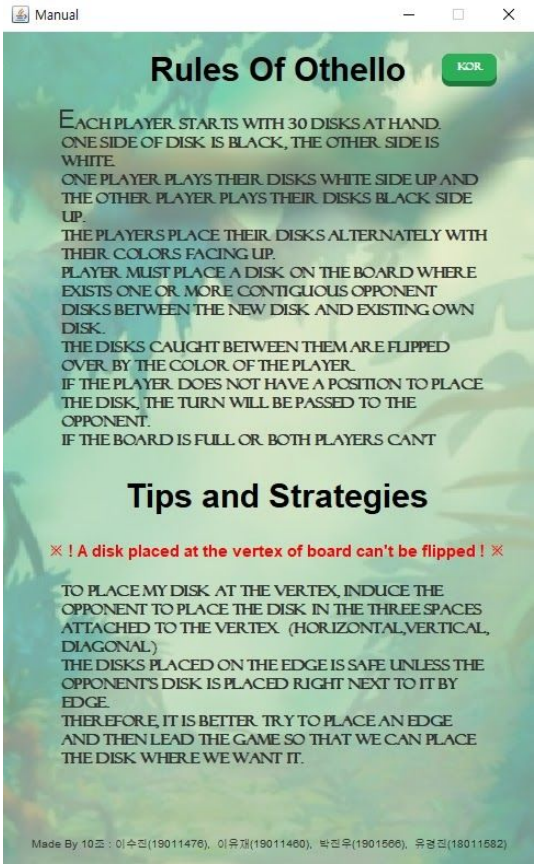
- 각종 옵션들과 캐릭터, 토큰, 점수 등으로 구성되어 있다.

❑ **기능**

- 보드판 : 검정색 토큰과 흰색 토큰을 번갈아 가면서 놓게 되고 보드판 위에 있는 차례가 'White Turn' 이라면 흰색 샤도우 토큰이, 'Black Turn' 이라면 검정색 샤도우 토큰이 보여지게 된다.
- 점수 : 검정색 토큰과 흰색 토큰의 개수를 보여준다. 플레이어가 토큰을 놓게 되면 토큰의 개수가 달라지는데 이때 해당 플레이어의 점수가 업데이트된다.
- 캐릭터 : 왼쪽과 오른쪽에 있는 캐릭터는 점수에 따라 모습이 바뀌며 진화하거나 퇴화한다.
- 매뉴얼 : 왼쪽 위 빨간색 버튼을 누르면 초기 화면에서처럼 게임 방법과 승리 전략을 한국어와 영어, 두가지 버전으로 확인할 수 있다.
- 옵션 : 오른쪽 위 톱니바퀴 모양 버튼을 누르면 배경 음악과 효과음을 조절할 수 있다.
- 종료 : 옵션 버튼 옆 문 모양 버튼은 게임을 하는 도중 게임을 다시 시작하고 싶거나 종료하고 싶을 때 이를 처리해준다.

5. 게임 매뉴얼





❑ UI 화면 설명 / 실행화면 설명

- 초기 화면에서 'MANUAL' 버튼을 클릭하거나 게임 화면에서 왼쪽 위 빨간색 버튼을 클릭했을 때 나오는 매뉴얼이다.

❑ 기능

- 한국어 : 매뉴얼의 처음 화면은 한국어로 되어있고 게임 방법과 승리 전략에 대해 자세히 나와있다.
- 영어 전환 버튼 : 'ENG' 버튼을 누르면 영어로 게임 방법과 승리 전략을 볼 수 있다.
- 한글 전환 버튼 : 'KOR' 버튼을 누르면 다시 한국어로 게임 방법과 승리 전략을 볼 수 있다.
- 팀원 : 매뉴얼의 맨 아래에는 개발자의 이름(팀원명)이 적혀있다.

6. 게임 옵션

❑ **UI 화면 설명 / 실행화면 설명**

- 게임 화면에서 오른쪽 위 톱니바퀴 모양의 버튼을 눌렀을 때 나오는 화면이다.

❑ **기능**

- 음량 조절 : 배경 음악(BGM)과 효과음(SFX)을 조절 할 수 있다.
- 확인 : ‘OK’ 버튼을 누르면 배경 음악(BGM) 세가지가 번갈아 가면서 들리게 되고 조절한 음량에 맞게 볼륨이 설정된다.
- 닫기 : ‘CLOSE’ 버튼을 누르면 옵션 창이 닫힌다.

7. 게임 나가기

❑ **UI 화면 설명 / 실행화면 설명**

- 게임 화면에서 오른쪽 위 문 모양 버튼을 누르게되면 나오는 화면이다.

❑ **기능**

- 게임 재시작 : ‘GAME RESTART’ 버튼을 누르면 게임을 정말 다시 시작할지 묻는 창이 나타난다.
- 게임 종료 : ‘EXIT’ 버튼을 누르면 게임을 정말 종료할지 묻는 창이 나타난다.
- 취소 : ‘CANCEL’ 버튼을 누르면 하고 있던 게임 화면으로 돌아간다.

7 - 1. ‘GAME RESTART’ 버튼 클릭

❑ **UI 화면 설명 / 실행화면 설명**

- 위의 화면에서 ‘GAME RESTART’ 버튼을 눌렀을 때 보여지는 화면이다.

❑ **기능**

- 예 버튼 : ‘Yes’ 버튼을 누르면 게임을 다시 시작하게 되고 게임 모드 선택 화면으로 돌아간다.
- 아니오 버튼 : ‘No’ 버튼을 누르면 하고 있던 게임 화면으로 돌아간다.

7 - 2. ‘EXIT’ 버튼 클릭

❑ **UI 화면 설명 / 실행화면 설명**

- 위의 화면에서 ‘EXIT’ 버튼을 눌렀을 때 보여지는 화면이다.

❑ **기능**

- 예 버튼 : ‘Yes’ 버튼을 누르면 프로그램이 종료된다.
- 아니오 버튼 : ‘No’ 버튼을 누르면 하고 있던 게임 화면으로 돌아간다.

8. 게임 오류 및 알림 메시지

8 - 1. 오류 메시지

❑ **UI 화면 설명 / 실행화면 설명**

- 게임 도중 보드판 내에서 섀도우 토큰 외에 다른 곳을 클릭했을 때 보여지는 오류 화면이다.

❑ **기능**

- 오류 메시지 : 보드판 내에서 섀도우 토큰 이외에 다른 곳을 클릭했을 때 'Cannot be placed' 라는 오류 메시지가 뜬다. 'OK' 버튼을 누르면 하고 있던 게임 화면으로 돌아간다.

8 - 2. 알림 메시지

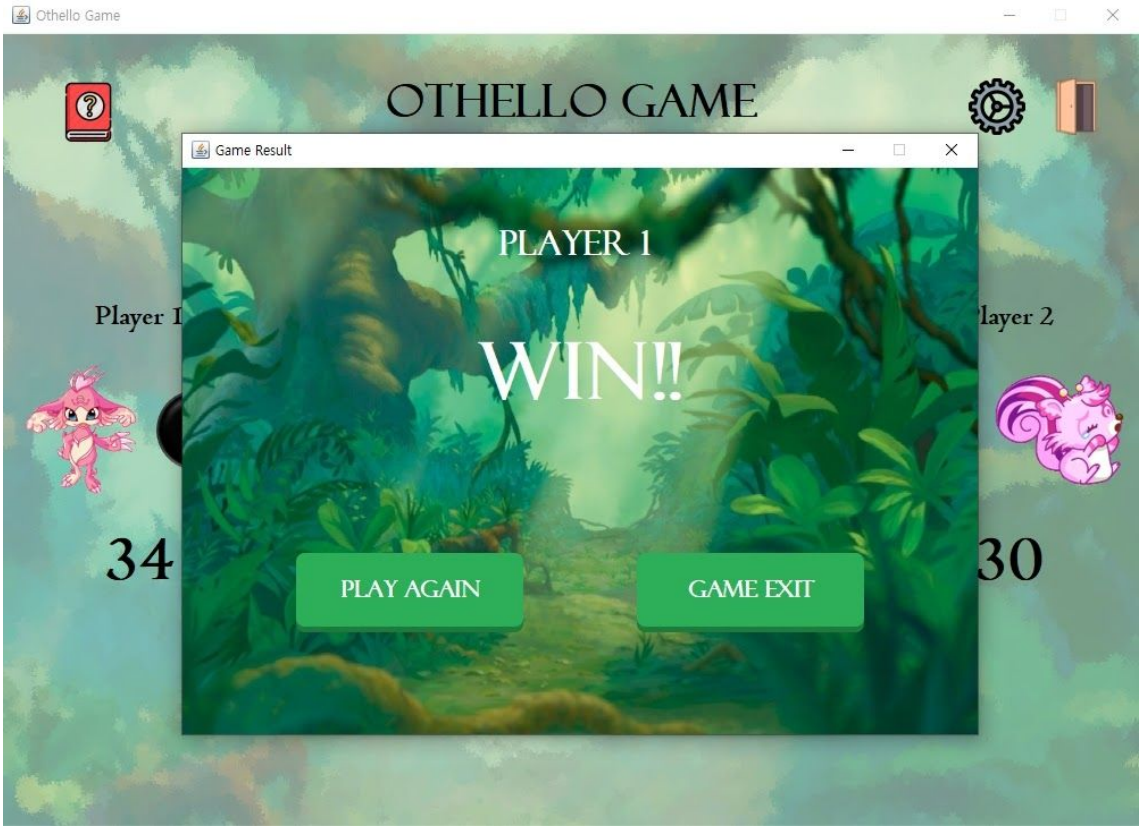
❑ **UI 화면 설명 / 실행화면 설명**

- 차례가 상대방에게 넘어갔다는 것을 알려주는 알림 메시지 화면이다.

❑ **기능**

- 알림 메시지 : 토큰을 놓을 수 있는 곳이 없을 경우 상대방에게 차례가 넘어가게 되는데 이를 알려주기 위해 'Turn skip : current Turn is (White/Black)!' 라는 알림 메시지가 뜬다. 'OK' 버튼을 누르면 하고 있던 게임 화면으로 돌아간다.

9. 게임 엔딩



❑ **UI 화면 설명 / 실행화면 설명**

- 게임이 끝났을 때 보여지는 화면이다.

❑ **기능**

- 게임 다시하기 : ‘PLAY AGAIN’ 버튼을 누르면 게임을 다시 시작할 수 있고 게임 모드 선택 화면으로 돌아간다.
- 게임 종료 : ‘GAME EXIT’ 버튼을 누르면 프로그램이 종료된다.

5) 발표 시 질문사항 및 답변 정리 내용

Q1. ai는 어떤 식으로 구현하셨는지 설명 해주실 수 있나요?

A. 이 알고리즘은 상대방 차례와 내 차례가 번 갈아가며 진행되는 게임방식과 상대의 이득은 나의 손해라는 제로섬 특성을 지닌 메커니즘에서 사용할 수 있고 상대방은 나의 이익을 최소로 하는 방향으로 게임을 진행할 것이며 나는 내 이익을 최대로 하는 방향으로 진행한다는 것을 가정한 알고리즘입니다. 내 차례에서 내가 게임 보드에 어떤 행동을 할 수 있는 경우의 수가 여러가지 일 수도 있는데, 모든 경우에 대해 재귀적으로 깊이 우선 탐색을 적용합니다. 점점 깊어지다 미리 지정한 어떤 깊이까지 가면 부모 노드로 반환 값들을 전달하며 값을 전달받은 노드는 자식 노드 간의 비교를 진행하고, 나의 차례일때는 가장 큰 값을, 상대의 차례일때는 가장 작은 값을 선택하는 방식으로 구현됩니다.

Q2. 클래스가 상당히 커보였는데 awt.Graphics 상속받으신건가요?

A. Graphic이라는 클래스는 따로 상속을 받지는 않았지만 내부에서 페인트를 위해 awt.Graphics 클래스를 상속받은 것이 있습니다.

Q3. 그렇다면 게임에 새롭게 등장하는 버튼이 생긴다면 그래픽 클래스를 수정해야하나요?

A. 새로운 버튼을 추가하려고 한다면 그래픽 클래스를 수정해야 되겠죠. 그리고 저희는 페이지별로 panel의 형태로 화면을 관리했기 때문에 그리 어렵지 않게 관리할 수 있습니다.

6) 팀구성 및 역할분담

박진우	❑ **ManualPanel** ● ShowManual의 JFrame에 추가할 패널이 담긴 내부 클래스인 ManualPanel 작성 ● ManualPanel의 한글-영어 전환 버튼에 추가할 ButtonListener 클래스 와 actionPerformed메소드 작성 ❑ **Graphic** ● 프로그램 실행과 게임모드 선택화면에 들어가는 컴포넌트 생성 및 배치 ● 캐릭터 선택 화면 구현 및 게임에 관한 정보를 GameStartInfo 클래스에 저장 할 수 있도록 구현 ❑ **Token** ● 토큰의 정보를 저장하는 변수 생성 ● 토큰의 정보를 반환하는 메서드 작성 ❑ **GameStartInfo** ● 게임모드와 플레이어의 정보가 담긴 멤버변수 생성 ● 게임모드와 토큰의 정보를 설정하는 setGameStartInfo 메소드 작성 ● 플레이어의 정보를 설정하고 가져오는 setPlayer, getPlayer 메소드 작성 ● 게임모드를 반환하는 getGameMode 메소드 작성
유경진	❑ **Graphic** ● 게임 화면에서 음량 조절, 게임을 exit 할 수 있는 버튼 생성 ● 모든 버튼과 토큰을 놓을 때, 게임이 시작할 때, 끝날 때마다 효과음이 나도록 구현 ❑ **ShowOptions** ● 게임 화면에서 음량 조절을 할 수 있는 옵션을 구현 ● 지정된 사운드를 반복 재생하도록 구현 ❑ **Exit** ● 게임 화면에서 게임을 재시작하거나 게임을 종료할 수 있는 옵션 창을 구현 ❑ **VolumeControl** ● 배경 음악이 조절될 때 다른 음악으로 바뀌도록 구현 ● 배경 음악과 효과음의 음량이 조절되도록 구현 ❑ **Sound** ● 배경 음악이 재생되고 멈추도록 구현 ● 버튼이나 토큰을 놓을 때, 게임을 시작하거나 끝날 때의 효과음이 재생되도록 구현 ❑ **ErrorManager** ● 게임 오류 메시지와 알림 메시지를 구현

이수진 (팀장)	❑ **StyledButtonUI** ● 버튼의 모양과 색을 커스텀할 수 있는 클래스 설계 ● 버튼의 press 유무에 따라 버튼의 형태를 다르게 구현 ❑ **Graphic** ● 프로그램 실행 시 가장 먼저 보여지는 홈 화면을 구성 ● 각 플레이어의 점수를 점수 라벨에 업데이트 ● 게임 페이지의 오른쪽 끝과 왼쪽에 캐릭터를 위치하며 토큰의 점수에 따라 모습이 변하도록 구현 ● 게임 화면의 배경이미지와 배치 구성 ❑ **Board** ● 현재 보드판의 상황을 token이미지와 shadow token이미지를 사용하여 매번 업데이트해주고 paint해주도록 설계
이유재	❑ **GameManager** ● 현재 차례의 객체가 어디에도 토큰을 놓을 수 없음을 판단하는 메서드 구현 ● 8x8게임판에서 입력으로 들어오는 위치에 token을 놓을 수 있는지를 판단하는 메서드 구현 ● 8x8게임판에서 현재 차례의 객체가 놓을 수 있는 위치를 boolean[][]으로 반환하는 메서드 구현 ● 8x8게임판이 모두 토큰으로 찼는지 판단하는 메서드 구현 ❑ **GameBoard** ● 초기 상태로 초기화 시켜주는 메서드 구현 ● 게임판을 기준으로 각 토큰의 개수를 업데이트 하는 메서드 구현 ● 위치를 입력받아 그 위치에 놓음으로써 발생되는 변화를 게임판에 반영하는 메서드 구현 ❑ **Player** ● 사용자가 지정한 마우스의 위치를 보드판의 인덱스로 맵핑 시켜주는 메서드 작성

	❑ **TimeStop** ● 지정한 시간만큼 쓰레드를 정지시켜 시간을 지연하는 메서드 작성 ❑ **Computer** ● 게임승리를 위한 최적의 수를 탐색하는 알고리즘 구현 ● 2차원 배열의 내용을 복사해주는 메서드 작성 ❑ **Graphic** ● 게임 플레이시 사용자가 적절한 위치가 아닐 때 대응 코드 작성 ● 게임 플레이시 사용자의 차례때 놓을 수 있는 칸이 존재하지 않아 스킵되는 로직 구현 ● 게임이 끝났음을 알아내는 로직 구현 ● 한 패널로 모든 페이지를 보여주던 것을, 각각의 고유한 패널로 분리

일자	내용	비고
9/29	**[진우, 경진, 수진, 유재]** ❑ 주제 정하기 ● 주제: 종합보드게임 -> 쿼리도 -> 오델로 ● 오델로 확정 ❑ 관련 오픈소스 검색 및 공유 ❑ UML 관련 부가적인 개념 공부하기	● 9/29 온라인 회의 20:00 => 오델로의 구성 클래스 구조도 생각해오기 (멤버 변수 + 멤버 함수)
10/3	**[진우, 경진, 수진, 유재]** ❑ 클래스 분류 ● 각 클래스별 멤버 변수와 메서드 작성(접근 제어자, 반환 타입, 매개변수, 기능)	● 10/3 온라인 회의 15:00, 20:00 => 각자 역할분담한 부분 작성
10/6	**[진우, 경진, 수진, 유재]** ❑ 자신이 맡은 클래스를 바탕으로 역할 분담 내용 작성 **[진우]** ❑ 주요 기능 **[경진]** ❑ 팀프로젝트 제목 및 개요 작성 **[수진]** ❑ 전체 클래스 구조도 - 각 클래스별 간략한 설명 **[유재]** ❑ 전체 클래스 구조도 - UML 표기	
10/7 ~ 10/8	**[진우, 경진, 수진, 유재]** ❑ 팀프로젝트 제안서 수정 및 제출	● 10/7 온라인 회의 14:00 => 계획서 확인 및 수정 ● 10/8 제안서 제출

10/9 ~ 10/11	**[진우, 경진, 수진, 유재]** ❏ 클래스에 사용될 주요 기능 및 구현 방법에 대해 생각하고 참고될 만한 자료 찾기(오픈소스, 강의, 영상)	
10/12~11/1 클래스 구현 작업		
10/12 ~ 10/18	**[진우]** ❏ ShowManual 클래스 구현 ❏ ShowManual의 JFrame에 추가할 패널이 담긴 내부 클래스인 ManualPanel 작성 ❏ ManualPanel의 한글-영어 전환 버튼에 추가할 ButtonListener 클래스 와 actionPerformed메소드 작성 **[경진]** ❏ Sound 클래스 구현 ❏ 프로그램을 실행 시켰을 때 배경 음악이 적절한 음량으로 나오도록 구현 ❏ 버튼을 클릭했을 때 효과음이 적절한 음량으로 나오도록 구현 ❏ 배경 음악이 반복 재생되도록 구현 ❏ 게임이 시작될 때, 끝날 때 효과음이 다르게 나오도록 구현 ❏ 배경 음악 3개가 재생 되도록 구현 **[수진]** ❏ Graphic 클래스 구현 ❏ 1인,2인 모드 선택 메서드 (selectMode) 작성 ❏ 1인 모드일 경우 토큰의 색을 정하고 정한 색을 반환해주는 메서드 (selectToken) 작성 ❏ GUI 디자인 생각하기 ❏ 초기화면 ❏ 선택모드 ❏ 캐릭터 **[유재]** ❏ GameManager 클래스 일부 구현 ❏ 특정 위치에 대해 현재 턴의 객체가 접근 가능한 위치인지를 판단하는 메서드 구현(isPossible) ❏ 모든 위치에 대해 현재 턴의 객체가 접근 가능한 위치를 판단하여 boolean[][]으로 반환 해주는 메서드 구현(findPossiblePosition) ❏ GameBoard 클래스 구현 ❏ 게임판을 초기 상태로 초기화 해주는 메서드 구현(initBoard) ❏ 게임판을 기준으로 검은색, 흰색 토큰의 개수를 업데이트 하는 메서드 구현(updateEachCnt) ❏ 위치를 입력받아 그 위치에 놓음으로써 발생되는 변화를 게임판에 반영하는 메서드 구현(updateBoard)	● 10/16 중간 점검 & 온라인 회의 => 구현 방법과 동작 원리 설명 및 의논할 부분 제시, 오류나 예외 사항 생각해보기

	❑ Computer algorithm 구현 생각 ❑ Greedy ❑ Graph ❑ reversi algorithm	
10/19 ~ 10/25	**[진우]** ❑ Token 클래스 구현 ❑ 토큰의 정보를 반환하는 메서드(getState) 작성 ❑ Graphic 클래스 일부 구현 ❑ 프로그램 실행과 게임모드 선택화면에 들어가는 컴포넌트 생성 및 배치 ❑ 캐릭터 선택 화면 구현 및 게임에 관한 정보를 GameStartInfo 클래스에 저장 할 수 **[경진]** ❑ ShowOptions 클래스 구현 ❑ 게임 화면에서 버튼을 클릭하면 출력되는 프레임을 구성 ❑ 프레임 안에 패널을 구성하고 음량을 조절 할 수 있게 옵션을 구성 ❑ VolumeControl 클래스 구현 ❑ 옵션에서 조절된 음량이 재생 될 수 있도록 구현 ❑ Graphic 클래스 구현 ❑ 게임 화면에서 옵션 버튼 생성 ❑ 모든 버튼과 토큰을 놓을 때, 게임이 시작할 때, 끝날 때마다 효과음이 나도록 구현 **[수진]** ❑ Graphic 클래스 구현 ❑ 시작화면에 대한 메서드 작성 ❑ 점수에 따라 서로 다른 캐릭터를 보여주는 메서드작성 **[유재]** ❑ GameManager클래스 구현 ❑ 현재 차례의 객체가 놓을 수 있는 자리가 있는지 여부를 판단하는 메서드 구현(canSkip) ❑ 객체가 어떤 수를 놓음으로써 발생하는 변화에 맞게 board를 업데이트 하는 메서드 구현(updateBoard) ❑ 게임판이 다 찼는지 판단하는 메서드 구현 (isFull) ❑ Player 클래스 구현 ❑ 사용자의 마우스 위치를 게임판의 인덱스로 변환하는 메서드 작성 (convertPosToIdx) ❑ Computer algorithm 구현 ❑ TimeStop 클래스 구현 ❑ 쓰레드를 정지 시키는 메서드 작성(run)	● 10/23 중간 점검 & 온라인 회의 => 구현 방법과 동작 원리 설명 및 의논할 부분 제시, 오류나 예외 사항 생각해보기

10/26 ~ 11/1	**[진우]** ❏ GameStartInfo ❏ 게임모드와 플레이어의 정보가 담긴 멤버변수 생성 ❏ 게임모드와 토큰의 정보를 설정하는 setGameStartInfo 메소드 작성 ❏ 플레이어의 정보를 설정하고 가져오는 setPlayer, getPlayer 메소드 작성 ❏ 게임모드를 반환하는 getGameMode 메소드 작성 **[경진]** ❏ Exit 클래스 구현 ❏ 게임 도중 재시작을 하거나 게임을 나갈 수 있는 옵션창을 띄우도록 구현 ❏ ErrorManager 클래스 구현 ❏ 게임 도중 오류 메시지나 알림 메시지를 출력하도록 구현 **[수진]** ❏ Graphic 클래스 구현 ❏ 여러가지 플레이 모드 중 하나를 선택하는 옵션 페이지 구성 및 메서드(showPlayOptions) 작성 ❏ 매뉴얼 작성 내용 구상 ❏ 시작화면 내 버튼 구성 및 매뉴얼 관련 메서드(showManual) 작성 **[유재]** ❏ Graphic 클래스 구현 ❏ 어떤 객체의 차례가 skip 되었을 때 사용자에게 공지해주는 메서드 호출 시점을 판단하는 로직 작성 ❏ 게임이 끝났을 때를 판단하는 로직 작성 ❏ Computer algorithm 검증 및 최적화 모색	● 10/30 중간 점검 & 온라인 회의 => 구현 방법과 동작 원리 설명 및 의논할 부분 제시, 오류나 예외 사항 생각해보기
11/2 ~ 11/22 클래스 병합, 추가 사항 및 개선 사항		
11/2 ~ 11/4	**[진우, 경진, 수진, 유재]** ❏ 클래스 병합 후 프로그램 실행시켜보기	● 11/3 온라인 회의 => 추가 사항 의논, 문제점 및 개선 사항 의논
11/5	**[진우, 경진, 수진, 유재]** ❏ 문제점 및 수정사항 의논 ❏ 해결방법에 관한 아이디어 제시	● 11/5 온라인 회의 => 병합 후 프로그램 실행해보고 수정 및 개선 사항 의논 => 추가 기능에 대한 아이디어 공유
11/6 ~ 11/8	**[진우, 경진, 수진, 유재]** ❏ 문제점 해결 및 개선사항 ❏ 추가 사항 의논 ❏ 추가적인 기능에 대한 아이디어 ❏ 필요 클래스 ❏ 새로운 클래스 구현 및 해결 방법에 참고할 만한 자료 찾기(오픈소스, 강의, 영상)	● 11/7 온라인 회의 => 문제해결과 수정된 부분 재결합 => 추가할 기능과 그에 따른 클래스 상의, 역할 분담

11/9 ~ 11/22	**[진우, 경진, 수진, 유재]** ❑ 이미 구현된 부분에 대한 개선 및 오류 수정 ❑ 각자 맡은 클래스에 대한 구현	● 11/20 온라인 회의 => 구현 방법과 동작 원리 설명, 오류나 예외 사항 생각해보기
11/23 ~ 11/25 모든 클래스 병합		
11/23 ~ 11/25	**[진우, 경진, 수진, 유재]** ❑ 클래스 병합 후 프로그램 실행시켜보기 ❑ 문제점 및 수정사항 의논 ❑ 해결방법에 관한 아이디어 제시	● 11/23 온라인 회의 => 병합 후 프로그램 실행해보고 수정 및 개선 사항 의논, 오류나 예외 사항 생각해보기
11/26 ~ 11/29 최종 검토 및 프로그램 실행		
11/26	**[진우, 경진, 수진, 유재]** ❑ 프로그램 실행시켜보기 ❑ 지인이나 가족을 통한 사전테스트해보기 ❑ 계속해서 확인 및 수정 ❑ 발표 자료 내용 구상	● 11/29 프로그램 완성 => 발표 내용 구상해오기
11/27 ~ 11/28	**[진우,경진, 수진, 유재]** ❑ 발표 자료 준비 ❑ 각자 클래스 및 메서드 설명 **[진우,유재]** ❑ 클래스 구조도 작성 **[경진]** ❑ GUI 기능 및 실행화면 **[수진]** ❑ 제목 및 개요 ❑ 역할분담 내용	● 11/27 온라인 회의 => 발표자 선정 및 ppt 만들기 => 진우: 시연 및 화면 공유 경진, 수진: 발표 유재: 질문 담당
11/29	**[진우, 경진, 수진, 유재]** ❑ 발표 자료 내용 추가 보완 ❑ 발표자 및 시연자 발표 연습	● 11/29 온라인 회의 => 발표 연습
12/1	**[진우, 경진, 수진, 유재]** ● 발표 및 프로그램 시연 및 QnA ● 최종 보고서 작성	● 12/1 온라인 회의 => 최종보고서 작성
12/11	**[진우, 경진, 수진, 유재]** ● 최종 제출	

7) 프로젝트에 대한 개인의견

박진우	기초 지식도 없는 과목을 한 학기만에 프로젝트를 구현하려니까 막막했다. 그러나 수업시간에 배운 내용과 구글링에 기반하여 많은 시간과 노력을 쏟아부은 덕분에 만족스러운결과물을 만든 것 같아서 뿌듯하다. 힘들고 막막할 때 조원들과 토론하며 같이 문제점을 해결해 나간게 큰 도움이 되었던 것 같다. 프로그램 완성 이후에도 추가로 보완이나 개발하고픈 사항을 생각하며 프로그램의 완성도를 더 높이는 방안을 생각하며 이번 프로젝트에 상당한 노력을 기울이며 좋은 경험이 되었다고 생각한다.
유경진	기존에 프로젝트 제출 날짜가 29일이었어서 마감날까지 최대한 구현하려고 중간고사 끝나고부터 11월 거의 한달동안 프로젝트에 시간과 노력을 쏟았다. 처음이라 아무래도 힘든 부분도 있었지만 조원들이 다함께 적극적으로 참여하고 모르는 부분이나 안되는 부분있으면 같이 고민해주고 주저없이 말해달라고 해서 조금은 부담을 덜고 할 수 있었던 것 같다. 그래도 다 만들고 보니 너무나도 뿌듯했고 자바 수업이 끝난 후에도 더 나아가 exe파일로 만들어보거나 앱으로 만들어보면 어떨까 하는 생각도 들었다.
이수진	처음 접해보는 언어라 프로젝트 주제를 정하면서도 매우 조심스러웠다. 하지만 초반에 계획서를 작성한 게 큰 도움이 됐다. 클래스의 구현 방법이나 구조도, GUI 구성, 어느 부분에도 팀원들의 고민이 안들어간 부분이 없다. 고민을 하고 서로 의견을 나누는 과정에서 점점 프로그램의 모습이 구체적으로 그려졌다. 마지막에는 모든 팀원이 계속해서 오류를 수정하고 조금 더 좋은 결과물을 내고자 신중을 기울였다. 그 과정에서 모든 팀원이 협조적으로 임해줘서 고마웠고 노력한 만큼 만족스러운 프로젝트를 완성할 수 있어 뿌듯하다
이유재	개발자로서 취직을 하려면 다양한 프로젝트를 하라는 조언을 직접 들을 수도 있고 인터넷을 통해 볼 수 있다. 이러한 조언을 여러 번 접한 거 같은데, 이번 JAVA프로젝트가 인생 첫번째 프로젝트여서 더욱 의미 있었고, 느낀 것이 많아서 특별한 경험이었다. **1. 알고리즘 활용** 이전까지는 Online Judge사이트에서 알고리즘 문제를 푸는 방식으로 개발자로서 역량을 키워왔었고 이런 것들을 실생활에 적용시킬 수 있을지 의문이었다. 그러나 이번 OTELLO GAME프로젝트를 진행하면서 그래픽 일부분과 게임 진행같은 프로그램상 로직, 그리고 1인 모드 플레이시 컴퓨터 역할을 하는 알고리즘을 만들었는데 이것은 실생활에서 사용될 서비스하고 알고리즘을 잇는 첫 순간이었다. 기존 알고리즘에 대한 지식과 경험이 없었더라면 구현하는데 오래 걸리거나 구현을 하지 못했을 거라는 생각이 든다. **2. 절차지향과 객체지향** 현재 학교 수업에서 주로 절차지향 언어인 C언어를 사용하고 알고리즘 문제를 풀 때 C++를 사용하기는 했지만 STL을 위해서 사용한 것이기 때문에 스타일은 절차지향과 유사하게 코드를 작성하곤 했다. 프로젝트 제안서를 제출할 때 나름대로 논리 흐름들을 생각해보고 클래스들의 관계를 작성했지만, 앞서 말한 이유로 인해 절차지향 사고가 깊게 박힌 클래스들이었다. 그래서 객체지향

	언어인 JAVA로 프로젝트를 진행할 때 기존에 작성했던 것과 달리 추가되거나 삭제되는 메서드 혹은 클래스들이 존재하게 되었고, 절차 지향과 객체 지향의 차이를 몸소 느낄 수 있었던 순간이었다. **3. 팀플** 대학교에 들어오면서 느낀 것 중하나가 고등학교와 다르게 팀플이 많다는 것인데, 기업과 사회에서는 혼자서 일하기 보다는 협업을 통해 진행된다는 것을 생각해보면 당연히 팀플이 많아야 된다. 그리고 이번 프로젝트도 팀플로 진행되었고 팀플이었기에 좋은 점도 있었으며 팀플이었기에 안 좋은 점도 있었다. 우선 좋았던 점은, 내가 생각하지 못했던 부분을 다른 이는 생각하고 있다 라는 것이다. 그래서 내가 부족한 부분은 다른 팀원이 채워줄 수 있으며 이것이 팀플의 묘미라 생각한다. 하지만 이것은 나와 팀원의 의견이 다를 수 있다는 것을 암시하며 이것은 좋지 않은 방향으로 영향을 끼친다. 서로의 의견이 달라지게 되면 서로의 의견을 들어보면서 해결해야 되는데 이때 시간이 꽤 지나가버리는 문제가 발생하는데 이것이 팀플의 단점이라고 생각한다. 물론 이번 프로젝트는 팀플이었기에 가능했다.