

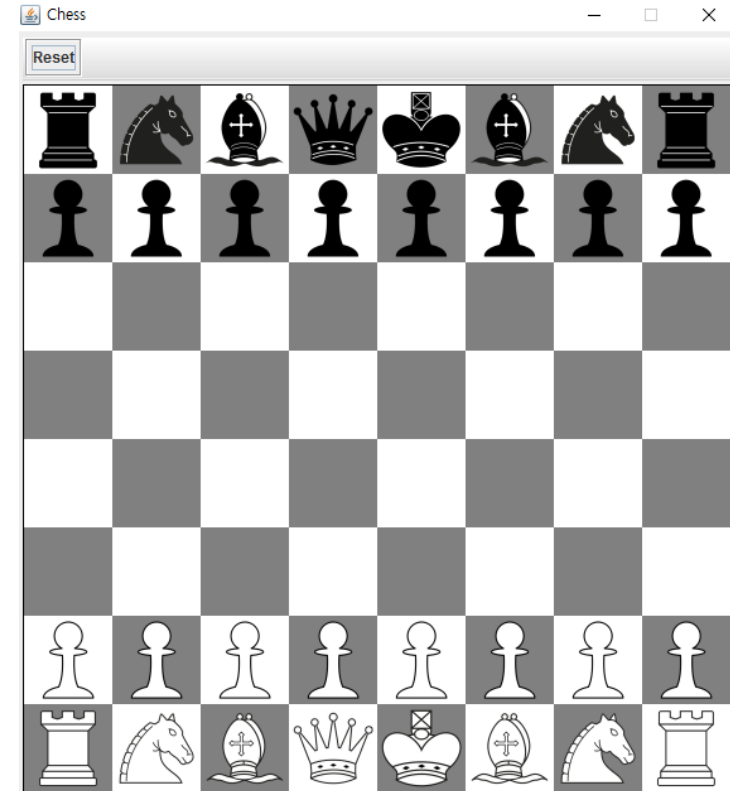
Spring 2019

# 컴퓨터프로그래밍

Project 2: Chess

## 개요

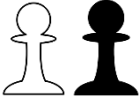
- 체스 게임을 Java GUI로 구현
  - Piece(말)들이 규칙대로 행동
  - Turn / Check / Checkmate 감지 기능

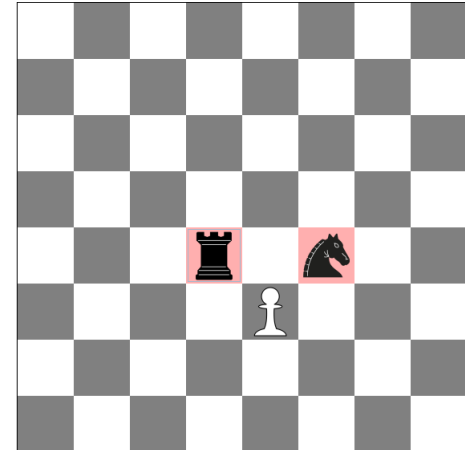
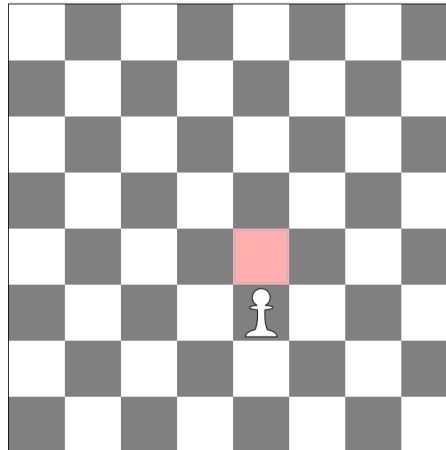
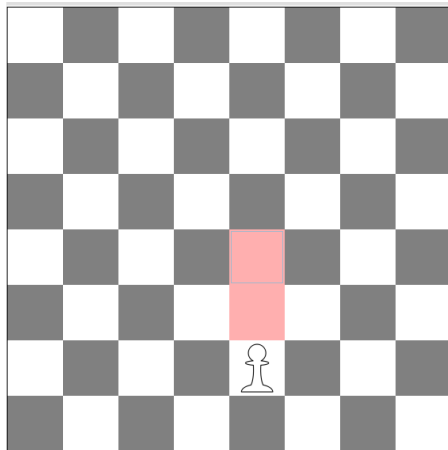


## 기본



- 흑/백 Player가 말을 번갈아가며 움직여 서로의 말을 잡는 게임
  - 흑은 위에서, 백은 아래에서 시작 / 흑이 선수(First Turn)
  - 한 턴당 하나의 말을 이동/공격
- 말의 종류에는 총 6가지가 있음
  - Pawn, Rook, Knight, Bishop, Queen, King
  - 말의 초기 배치는 이전 슬라이드의 그림 참고
- 말은 이동이나 공격이 가능함
  - 이동: 말이 없는 빈 칸으로 이동
  - 공격: 상대방 말이 있는 칸으로 이동하여 상대방 말을 제거
  - Pawn을 제외하고 이동 가능한 영역과 공격 가능한 영역이 같음
  - Knight를 제외하고는 이동 경로에 있는 말(아군/상대)을 뛰어넘는 것은 불가능

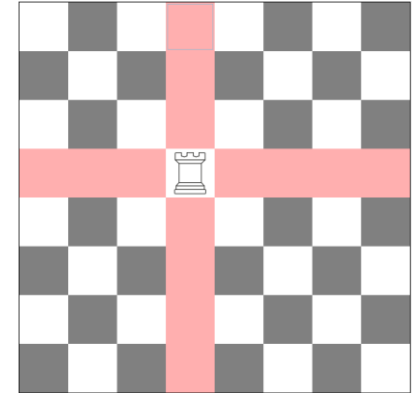
## 행마

- Pawn 
  - 앞으로 한 칸 이동 (앞: 흑 - 아래방향 / 백 - 윗방향)
  - 앞 방향의 대각선의 말 공격
  - Pawn이 처음으로 움직이는 경우, 두 칸 앞으로 이동이 가능하다

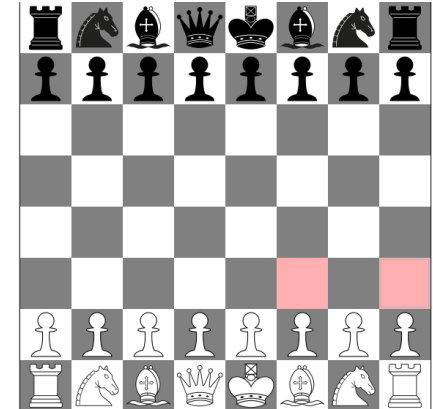
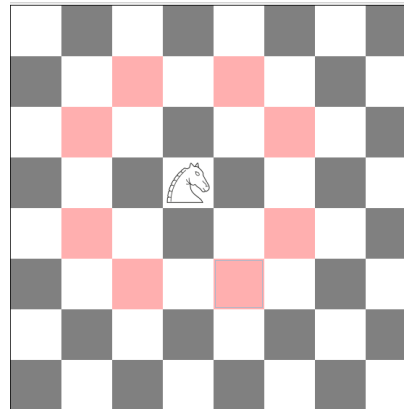


## 행마

- Rook  
  - 상하좌우(십자가) 방향에 있는 칸으로 이동 가능

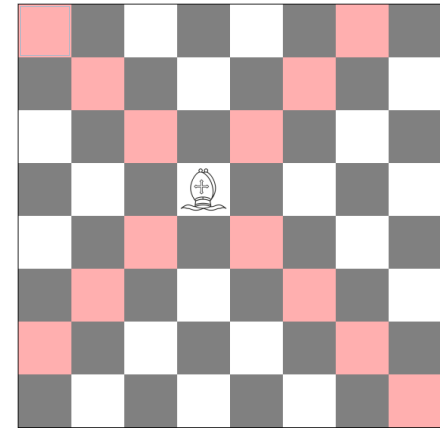


- Knight  
  - L 모양(2칸 + 1칸) 으로 이동
  - 경로상의 말을 뛰어넘을 수 있음

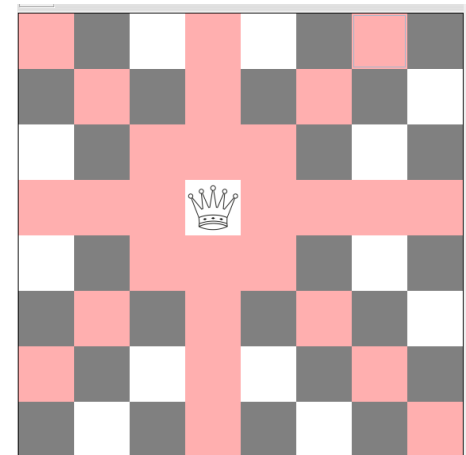


## 행마

- Bishop  
  - 대각 방향에 있는 칸으로 이동 가능

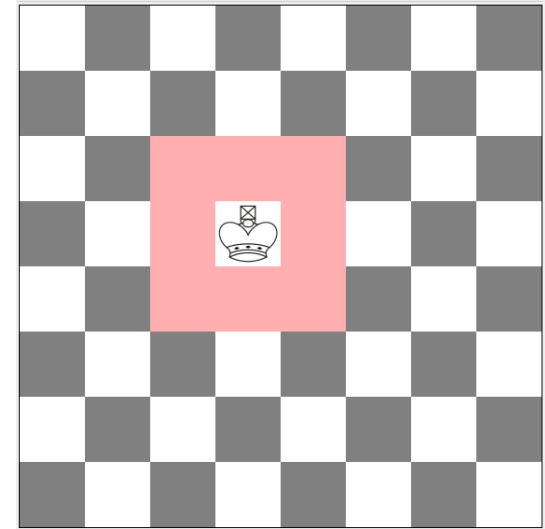


- Queen  
  - 대각방향 + 십자방향의 칸으로 이동 가능



## 행마

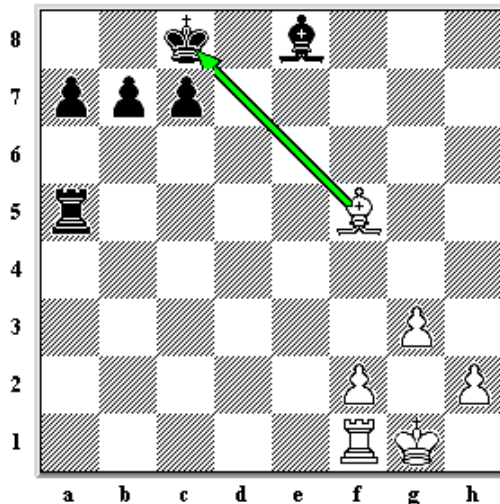
- King  
  - 모든 방향으로 1칸 이동 가능



- 슬라이드에 명시된 행마 이외의 체스 게임에서 정의된 다른 행마 (Promotion, Castling, En Passant 등)은 생각하지 않는다
- 기타 등등의 규칙은 알아서 검색

## Check

- 말을 상대방 왕이 잡힐 수 있는 위치에 도달시킨 경우, 이를 check 상태라고 한다

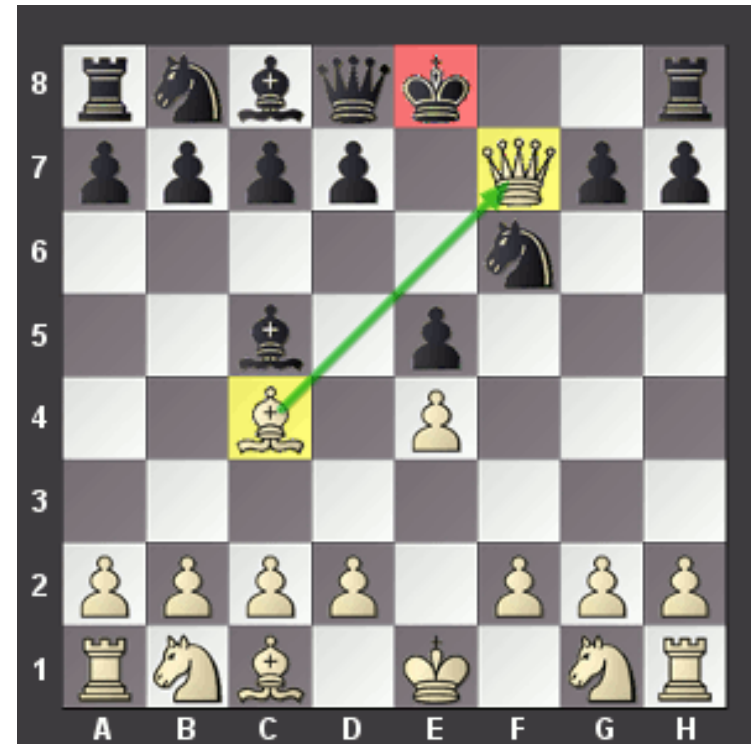
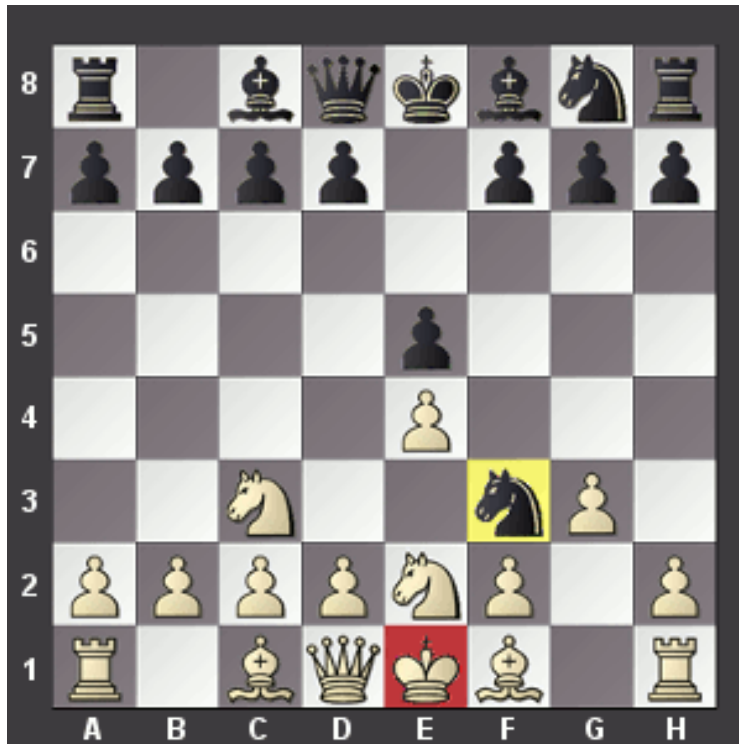


- Check 상태가 된 경우 상대는 아래와 같은 방법으로 check 상태에서 벗어날 수 있다.
  - 왕을 잡히지 않는 곳으로 이동시킨다
  - Check를 발생시킨 말을 제거한다
  - Check를 발생시킨 말을 왕에게 도달하지 못하도록 경로를 막는다



## Checkmate

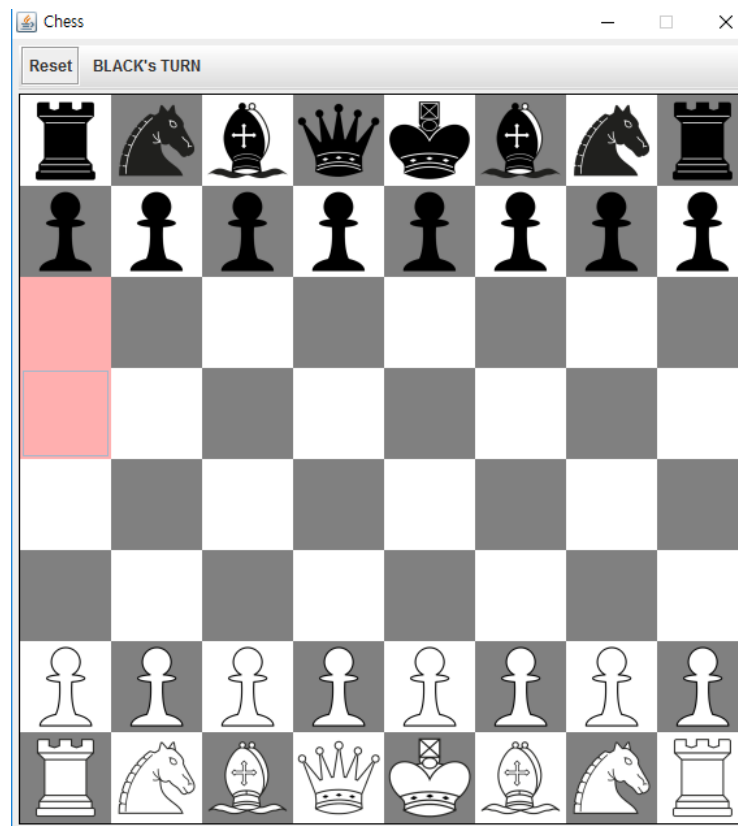
- 어떠한 경우에도 다음 턴에 왕이 잡힐 수 밖에 없는 경우, 즉 check 상태가 해소되지 않는 경우를 checkmate라고 한다.



## 구현

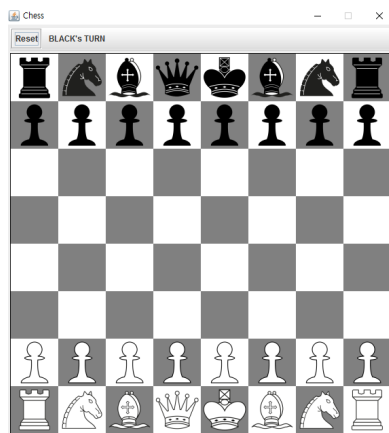
- 앞에서 설명한 규칙들이 올바르게 적용되는 GUI 체스 게임
  - 마우스로 말을 이동 시킬 수 있다
  - 현재 상태(turn 및 check)를 출력한다

\*편의상 첫 턴은 무조건 흑으로 한다



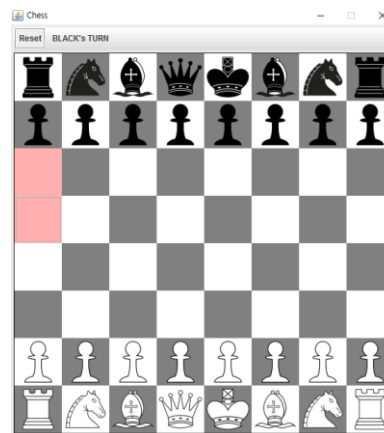
## 구현

- 자신의 턴에 자신의 말을 클릭한 경우
  - 그 말이 갈 수 있는 영역을 mark 해준다
    - 그 영역을 클릭하면 그 영역으로 말을 이동시키고, 턴을 종료 및 순서를 바꾼다
  - 그 말이 갈수 없는 영역을 선택하는 경우, mark를 해제한다
- 그 외에 판의 영역을 클릭했을 때, 어떤 반응도 하지 않는다
  - 게임이 끝난 경우(Checkmate나 왕이 죽은 경우), 모든 판의 영역을 클릭했을 때 반응하지 않도록 한다

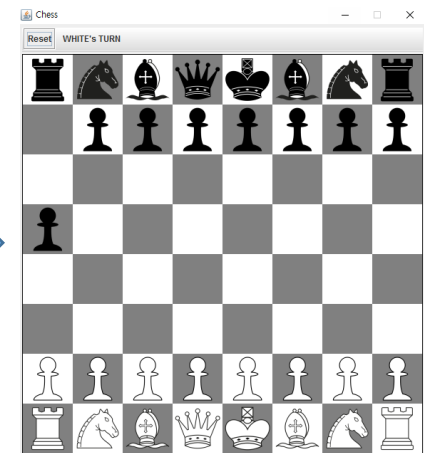


빨간 영역  
외 클릭

(1, 0) 클릭

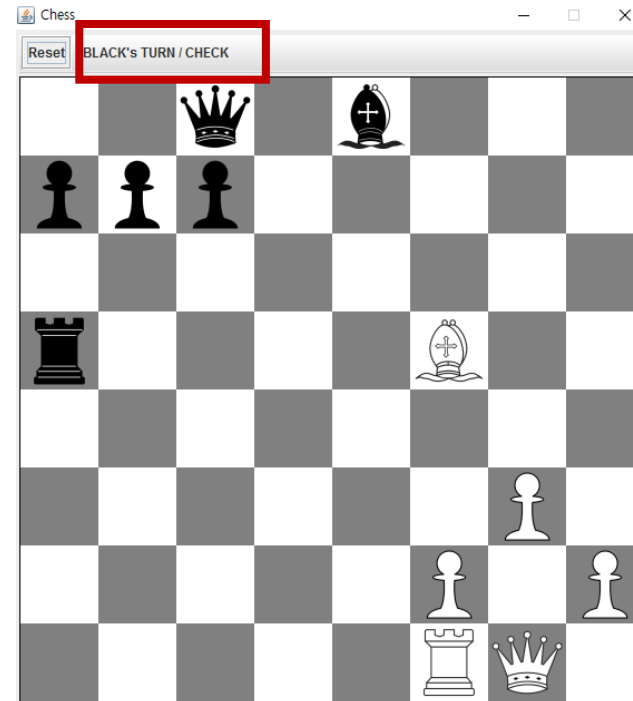
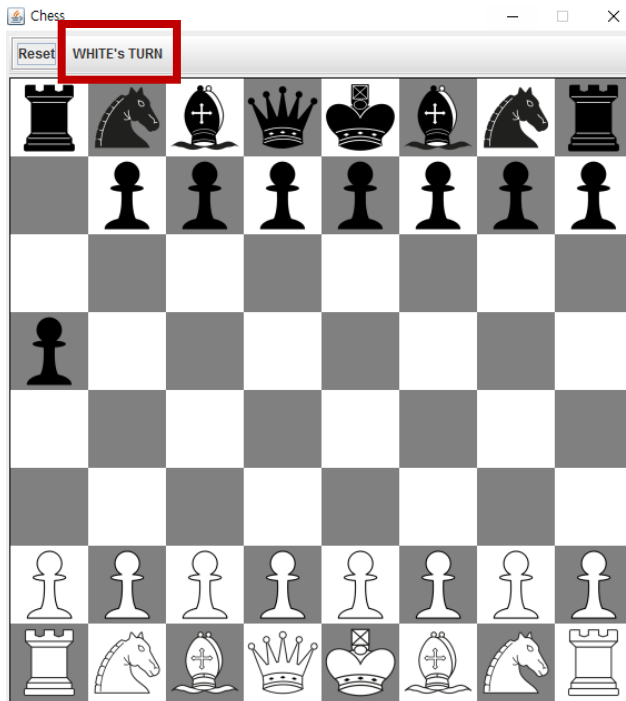


(3, 0) 클릭



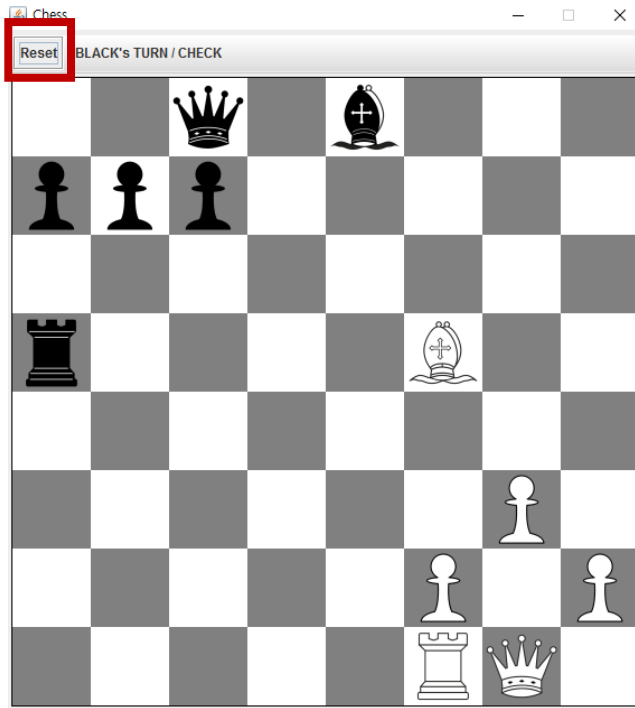
## 구현

- 위의 상태창에 turn과 check status(check/checkmate)를 나타내는 문자열을 출력한다
  - 출력하는 문자열의 제한은 없지만, turn과 check 상태는 명확히 알 수 있어야한다.

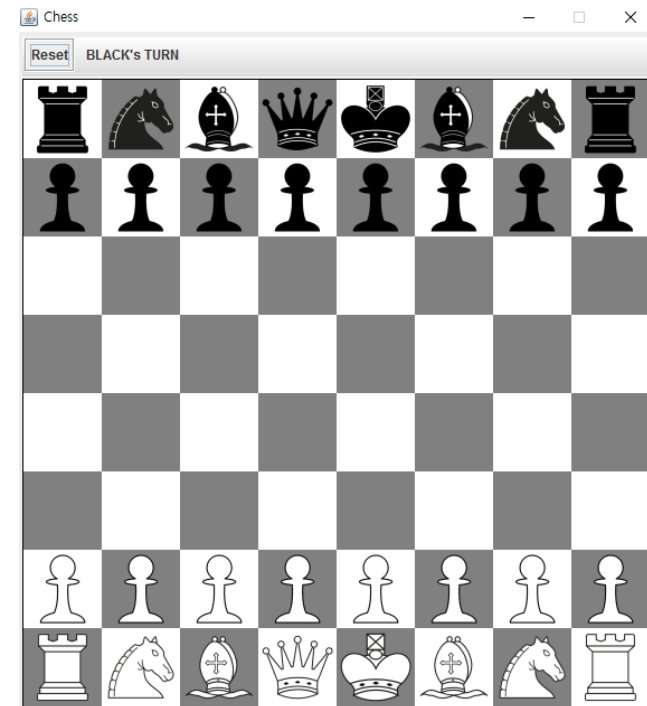


## 구현

- Reset을 클릭하는 경우 판을 다시 처음 상태로 되돌려야 함
  - 이미 구현되어 있음.



Reset 버튼 누름



## 뼈대코드

## • 뼈대 코드 제공

- GUI가 구현되어 있음 (Java Swing으로 구현)
- 뼈대 코드 내의 함수를 활용하여 GUI 조작
- 뼈대 코드는 마음대로 바꿀 수 있으나
  - Line 7~Line 199 사이의 코드는 가급적이면 바꾸지 말 것(주석으로 표시되어 있음)
  - 바꾼다면, 바꾼 부분 / 그 이유 / 바뀐 사용법 등을 보고서로 제출할 것

### IntelliJ에서 실행시키기

- 프로젝트 안에 chess라는 이름의 패키지를 만든다
- 패키지 안에 java 파일을 넣어준다
- img 폴더는 프로젝트 폴더 내에서 bin과 같은 위치에 넣어둔다.

## GUI 조작하기

- 코드에는 2 종류의 enum이 정의되어 있다
  - 말의 색과 종류에 관련 / none은 빈칸을 위해 사용

```
enum PieceType {king, queen, bishop, knight, rook, pawn, none}  
enum PlayerColor {black, white, none}
```

- 이들 enum을 활용하는 Piece라는 이름의 class가 있다
  - Piece는 말의 색과 종류에 대한 정보를 담고 있다
  - Piece에는 2개의 constructor가 존재한다.

```
class Piece{  
    PlayerColor color;  
    PieceType type;  
  
    Piece(){  
        color = PlayerColor.none;  
        type = PieceType.none;  
    }  
    Piece(PlayerColor color, PieceType type){  
        this.color = color;  
        this.type = type;  
    }  
}
```



## GUI 조작하기

- Chess Board는 다음과 같은 좌표계를 가진다

(0, 0)	(0, 1)	(0, 2)	..	..	..	..	(0, 7)
(1, 0)	(1, 1)	..	..	..	..	..	(1, 7)
(2, 0)	..	..	..	..			(2, 7)
..	..	..					(3, 7)
..	..						..
..	..						..
..	..	..					..
(7, 0)	(7, 1)	(7, 2)	..	..	..	..	(7, 7)

## GUI 조작하기

- 좌표계를 활용하여 원하는 칸의 위치에 원하는 그림을 그릴 수 있다.

```
public void setIcon(int x, int y, Piece p);  
public Piece getIcon(int x, int y)  
public void markPosition(int x, int y);  
public void unmarkPosition(int x, int y);
```

- setIcon: (x, y)에 p 에 해당하는 말 그림을 그림
  - p의 property중 하나라도 none이면 빈칸을 그림
- getIcon: (x, y)에 그려져 있는 Piece를 return함
  - 빈칸일 경우 p의 property중 하나 이상이 none임
- markPosition: (x, y)의 칸을 분홍색으로 칠함
  - 갈 수 있는 영역을 표시하기 위함
- unmarkPosition: (x, y)의 칸을 분홍색에서 원래 색으로 되돌림

## GUI 조작하기

- setStatus: 상태창의 message를 inpt로 바꿔 줌 `public void setStatus(string inpt);`

- 각종 행동에 대한 callback함수 또한 정의되어 있다

```
public void setStatus(string inpt);  
class ButtonListener implements ActionListener{  
    int x;  
    int y;  
    ButtonListener(int x, int y){  
        this.x = x;  
        this.y = y;  
    }  
    public void actionPerformed(ActionEvent e){           // Only modify inside here  
        // (x, y) is where the click event occurred  
    }  
}  
void onInitiateBoard(){  
}
```

- actionPerformed: (x, y)좌표를 클릭했을 때 발생하는 callback 함수
- onInitiateBoard : Reset 버튼을 클릭한 뒤 발생하는 callback 함수

### 제출 기한 및 참고사항

- 제출 기한 및 딜레이
  - 6월 14일 자정까지
  - 6월 16일 자정까지 딜레이 허용, 하루에 20%씩 감점
- java class 파일 etl에 업로드
  - 학번\_ChessBoard.java