Project 2: CHESS

2016-12299 컴퓨터공학부
박용주

알고리즘

1)말을 클릭할 때 분홍색 타일을 표시하고 분홍색 타일을 눌렀을 때 이동하는 부분

```
252     else if(getIcon(x,y).type == PieceType.king ){
253         Click_king(x, y ,PlayerColor.black);
254     }
255     else if(getIcon(x,y).type == PieceType.queen ){
256         Click_rook(x, y, PlayerColor.black, PlayerColor.white);
257         Click_bishop(x, y ,PlayerColor.black, PlayerColor.white);
258         Click_king(x, y ,PlayerColor.black);
259     }
260
261     }
262     else if((chessBoardSquares[y][x].getBackground() != Color.pink) && ((getIcon(x,y).color == PlayerColor.none || getIcon(x,y).type == PieceType.none)
263             || getIcon(x,y).color == PlayerColor.white)){   //검은색 말이나 핑크타일을 누르지 않을 경우 모든 마킹 해제
264         unmarkAll();
265     }
266     else if(chessBoardSquares[y][x].getBackground() == Color.pink){ //핑크타일을 눌렀을 때
267         if(chessBoardStatus[y][x].color == PlayerColor.white && chessBoardStatus[y][x].type == PieceType.king){ //만약 왕을 잡았을 경우
268             setIcon(x, y, getIcon(selX, selY));
269             setIcon(selX, selY, new Piece(PlayerColor.none, PieceType.none));   //누른 곳으로 말 이동
270             unmarkAll();    //핑크마크는 해제
271             setStatus("BLACK WINS!");   //검은색 승리
272             turn = TURN.GAME_OVER;  //게임 끝
273         }
274         else{
275             setIcon(x, y, getIcon(selX, selY));
276             setIcon(selX, selY, new Piece(PlayerColor.none, PieceType.none));   //말 이동
277             unmarkAll();
278             Checkmate_On_White_Turn();
279             if(checkmate){
280                 for(int i=0;i<8;i++)
281                     for(int j=0;j<8;j++){
282                         if(chessBoardStatus[j][i].color == PlayerColor.white && chessBoardStatus[j][i].type == PieceType.king){
283                             chessBoardSquares[j][i].setBackground(Color.red);   //체크메이트 당한 king을 빨간색으로 표시
284                         }
285                 }
286                 setStatus("WHITE'S TURN | CHECKMATE");
287                 turn = TURN.GAME_OVER;  //게임 끝
```

```
378     void Click_rook(int x, int y, PlayerColor Player_color, PlayerColor Enemy_color){
379         for(int i=1; x+i<8; i++){
380             if(getIcon(x+i,y).color == Player_color) break;
381             else if(getIcon(x+i,y).color == Enemy_color){
382                 markPosition(x+i, y);
383                 break;
384             }
385             else markPosition(x+i, y);
386         }
387         for(int i=1; x-i>=0; i++){
388             if(getIcon(x-i,y).color == Player_color) break;
389             else if(getIcon(x-i,y).color == Enemy_color){
390                 markPosition(x-i, y);
391                 break;
392             }
393             else markPosition(x-i, y);
394         }
395         for(int i=1; y+i<8; i++){
396             if(getIcon(x,y+i).color == Player_color) break;
397             else if(getIcon(x,y+i).color == Enemy_color){
398                 markPosition(x, y+i);
399                 break;
400             }
401             else markPosition(x, y+i);
402         }
403         for(int i=1; y-i>=0; i++){
404             if(getIcon(x,y-i).color == Player_color) break;
405             else if(getIcon(x,y-i).color == Enemy_color){
406                 markPosition(x, y-i);
407                 break;
408             }
409             else markPosition(x, y-i);
410         }
411         selX = x; selY = y;
412     }
```

먼저 말을 눌렀을 때 핑크색으로 표시할 이동 or 공격가능 범위를 함수로 구현했다. 예를 들어 click_rook함수를 보면 자신을 기준으로 상하좌우로 같은색 말이 나오면 표시를 중단하고 상대편 말일 경우 해당 말까지만 표시한다.

(+ queen을 눌렀을 땐 click_rook, click_knight, click_king 3가지 함수를 이용했다.)

그리고 체크메이트가 아닌데도 왕을 잡을 수 있으므로 이 경우에는 WHITE(BLACK) WINS! 로 문구를 변경하였다. 그리고 체크메이트일 때 체크메이트 당한 왕을 빨간색 타일로 표시해서 의미를 강조했다.

```
//=====================================================Implement below=====================================================//
    enum MagicType {MARK, CHECK, CHECKMATE}
    enum TURN {BLACK_TURN, WHITE_TURN, GAME_OVER}    //누구의 턴인지를 표시하는 enum
    TURN turn = TURN.BLACK_TURN;                     //처음은 흑으로 시작.
    private int selX, selY;
    private boolean check, checkmate, end;           //check인지 checkmate인지 상태를 나타내는 boolean
```

또, 게임이 끝났을 때 TURN을 GAME_OVER로 변경하여 BLACK이나 WHITE모두 턴을 시작하지 못하도록 해서 게임을 종료시킨다.

2)체크인지 확인하는 부분

상대편의 말의 공격범위 안에 자신의 왕이 있으면 자신은 체크를 당한 것이므로 이를 함수로 구현했다.

```
475  boolean Check_On_White_Turn(){
476      end = false; check = false; checkmate = false;
477
478      for(int x=0; x<8; x++)
479          for(int y=0; y<8; y++){
480              if(chessBoardStatus[y][x].color == PlayerColor.black){ //검은색 폰의 공격가능 범위에 흰색 킹이 있으면 check
481                  if(chessBoardStatus[y][x].type == PieceType.pawn){
482                      if(x+1 < 8 && y-1>=0 && chessBoardStatus[y-1][x+1].color == PlayerColor.white && chessBoardStatus[y-1][x+1].type == PieceType.king) return
483                      if(x+1 < 8 && y+1<8 && chessBoardStatus[y+1][x+1].color == PlayerColor.white && chessBoardStatus[y+1][x+1].type == PieceType.king) return c
484                  }
485                  else if(chessBoardStatus[y][x].type == PieceType.rook ){ //검은색 룩의 공격가능 범위에 흰색 킹이 있으면 check
486                      for(int i=1; x+i<8; i++){
487                          if(chessBoardStatus[y][x+i].color == PlayerColor.black) break;
488                          else if(chessBoardStatus[y][x+i].color == PlayerColor.white){
489                              if(chessBoardStatus[y][x+i].type == PieceType.king) return check = true;
490                              else break;
491                          }
492                      }
493                      for(int i=1; x-i>=0; i++){
494                          if(chessBoardStatus[y][x-i].color == PlayerColor.black) break;
495                          else if(chessBoardStatus[y][x-i].color == PlayerColor.white){
496                              if(chessBoardStatus[y][x-i].type == PieceType.king) return check = true;
497                              else break;
498                          }
499                      }
500                      for(int i=1; y+i<8; i++){
501                          if(chessBoardStatus[y+i][x].color == PlayerColor.black) break;
502                          else if(chessBoardStatus[y+i][x].color == PlayerColor.white){
503                              if(chessBoardStatus[y+i][x].type == PieceType.king) return check = true;
504                              else break;
505                          }
506                      }
507                      for(int i=1; y-i>=0; i++){
508                          if(chessBoardStatus[y-i][x].color == PlayerColor.black) break;
509                          else if(chessBoardStatus[y-i][x].color == PlayerColor.white){
510                              if(chessBoardStatus[y-i][x].type == PieceType.king) return check = true;
511                              else break;
```

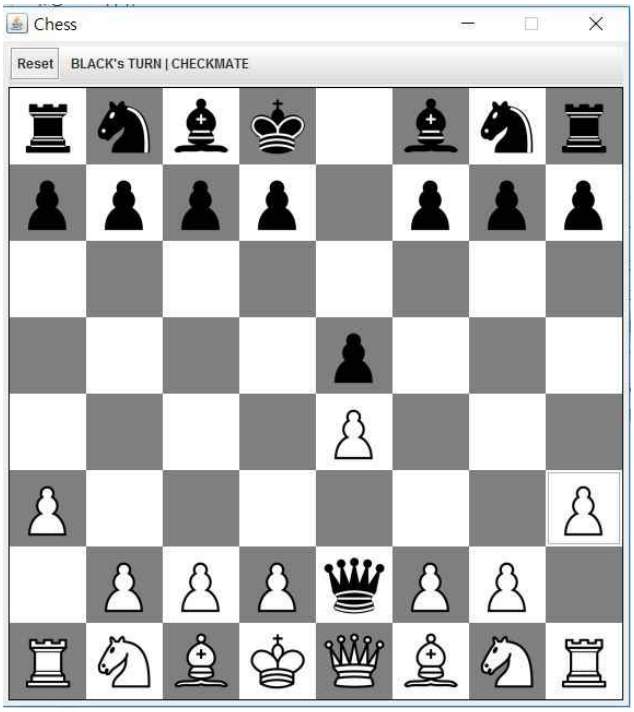
3)체크메이트인지 확인하는 부분

체크를 당한 후 자신의 말을 이동시킬 수 있는 모든 경우의 수로 이동시켜도 모든 이동에 대해 여전히 체크이면 체크메이트 이므로 체크인지 확인 -> 말 이동 -> 체크인지 확인 ->반복으로 알고리즘을 구현했다.

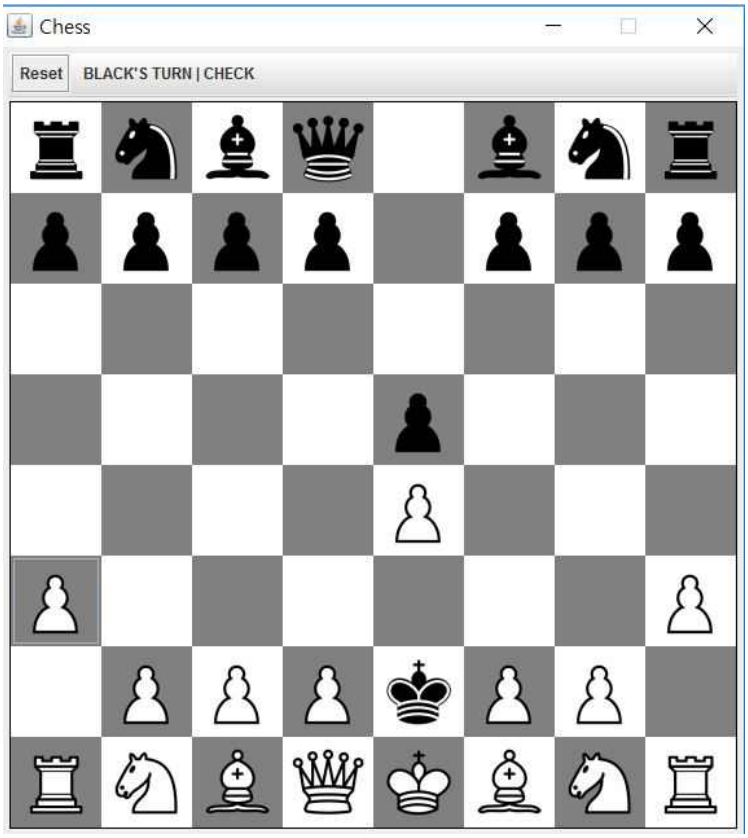
```
803  boolean Checkmate_On_White_Turn(){  //흰색 말을 이동시키고 모든 말에 대해서 여전히 check이면 checkmate이다.
804      Check_On_White_Turn();
805      if(!check) return false;    //체크가 아니라면 체크메이트도 아니므로 함수 종료.
806
807      end = false; check = false; checkmate = false;
808      Piece memory_Piece = new Piece(PlayerColor.none, PieceType.none);
809
810      for(int x=0; x<8; x++)
811          for(int y=0; y<8; y++){
812              if(chessBoardStatus[y][x].color == PlayerColor.white){
813                  if(chessBoardStatus[y][x].type == PieceType.pawn){
814                      if(x-1 >= 0 && getIcon(x-1,y).color == PlayerColor.none ){
815                          if(Checkmate_On_White_Type(-1, 0, x, y, memory_Piece) == false ) return false;
816                      }
817                      if(x-1 >= 0 && y-1>=0 && getIcon(x-1,y-1).color == PlayerColor.black ){
818                          if(Checkmate_On_White_Type(-1, -1, x, y, memory_Piece) == false ) return false;
819                      }
820                      if(x-1 >= 0 && y+1<8 && getIcon(x-1,y+1).color == PlayerColor.black ){
821                          if(Checkmate_On_White_Type(-1, 1, x, y, memory_Piece) == false ) return false;
822                      }
823
824                      if(x == 6 && getIcon(x-1,y).color == PlayerColor.none && getIcon(x-2,y).color == PlayerColor.none){
825                          if(Checkmate_On_White_Type(-2, 0, x, y, memory_Piece) == false ) return false;
826                      }
827                  }
828                  else if(chessBoardStatus[y][x].type == PieceType.rook){
829                      for(int i=1; x+i<8; i++){
830                          if(chessBoardStatus[y][x+i].color == PlayerColor.white) break;      //막혀있을 시 break
831                          else{                                                               //이동 가능할 시
832                              if(Checkmate_On_White_Type(i, 0, x, y, memory_Piece) == false ) return false;
833                              else {
834                                  if(chessBoardStatus[y][x+i].color == PlayerColor.none) continue;    //none이면 continue
835                                  else if(chessBoardStatus[y][x+i].color == PlayerColor.black) break; //black이면 break
836                              }
837                          }
838                      }
```

```
1353  boolean Checkmate_On_White_Type(int x_addition, int y_addition, int x, int y, Piece memory_Piece){  //Checkmate_On_White_Turn에서  말 하나를 옮기고 체크인지 확인하는 함수
1354      memory_Piece = getIcon(x+x_addition, y+y_addition);
1355      if(memory_Piece.type == PieceType.king) return false;    //만약 자신의 턴에 상대의 왕을 잡을 수 있다면 자신은 체크메이트를 당한 것이 아니므로 false를 리턴한다.
1356      setIcon(x+x_addition, y+y_addition, getIcon(x, y));
1357      setIcon(x, y, new Piece(PlayerColor.none, PieceType.none));      //말 이동
1358      Check_On_White_Turn();                                           //체크인지 확인
1359      setIcon(x, y, getIcon(x+x_addition, y+y_addition));
1360      setIcon(x+x_addition, y+y_addition, memory_Piece);               //원위치
1361      if(!check) return false;                                         //체크가 아니면 체크메이트가 아니므로 false 리턴
1362      else return true;                                                //체크면 넘어감
1363  }
```

그리고 바이너리의 체스게임에는



위와 같은 경우를 체크메이트로 구현되어있고 제 코드도 마찬가지로 저런 오류가 있어서 어디가 문제인지 찾아본 결과 checkmate_on 함수에서 자신이 상대말을 잡을 수 있는 경우를 고려하지 않음을 발견했습니다. 그래서 왕을 직접 잡을 수 있는 경우에는 checkmate_on 함수를 빠져나옴으로써 checkmate가 아니도록 구현했습니다.



```
else{
    setIcon(x, y, getIcon(selX, selY));
    setIcon(selX, selY, new Piece(PlayerColor.none, PieceType.none));    //말 이동
    unmarkAll();
    Checkmate_On_White_Turn();
    if(checkmate){
        for(int i=0;i<8;i++)
            for(int j=0;j<8;j++){
                if(chessBoardStatus[j][i].color == PlayerColor.white && chessBoardStatus[j][i].type == PieceType.king){
                    chessBoardSquares[j][i].setBackground(Color.red);    //체크메이트 당한 king을 빨간색으로 표시
                }
        }
        setStatus("WHITE'S TURN | CHECKMATE");
        turn = TURN.GAME_OVER;  //게임 끝
    }else{
        Check_On_White_Turn();
        if(end){
            setStatus("WHITE'S TURN");
            turn = TURN.WHITE_TURN;
        }
        else if(check){
            setStatus("WHITE'S TURN | CHECK");
            turn = TURN.WHITE_TURN;
        }
    }
}
```

마지막으로 자신의 턴에서 말을 이동시킨 직후 checkmate인지 확인한 후 checkmate(checkmate = true)이면 게임을 끝내고 아니면 다시 한 번 check인지 확인한 후 check(check = true)이면 check출력, 아니면(end = true) 그냥 턴을 넘겨주는 형식으로 구현했다.