바꾼 부분 1.

Class Piece 의 정의.

Polymorphism 을 보다 잘 활용하기 위해서,

Piece 에 좌표를 나타내는 field 와, 현재 자신이 마킹 되었는지 되지 않았는지를 판별하는 변수를 추가하였다.

또한 다음 메소드를 구현하여 Derived Class 에서 메소드를 오버라이드 하게 하였다.

setLocation : piece 의 위치를 해당 x,y 좌표로 바꾼다.

Marktiles : 이 piece 가 공격하거나 이동할 수 있는 타일을 마킹한다.

Unmarktiles : 이 piece 가 마킹했던 타일들의 마킹을 해제한다.

doActionTo : 해당 x,y 좌표에 이동 / 공격을 실행한다.

canMove : 해당 x,y 좌표로 이동할 수 있으면 true, 아니면 false 리턴.

canAttack : 해당 x,y 좌표를 공격할 수 있으면 true, 아니면 false 리턴.

```
class Piece{        //modified have more methods and current position.
    PlayerColor color;
    PieceType type;
    int x;                          // this piece's x position
    int y;                          // this piece's y position.
    private boolean marked;         // for check this piece is marked

    Piece(int x, int y){
        color = PlayerColor.none;
        type = PieceType.none;
        setLocation(x, y);
        marked = false;
    }
    Piece(PlayerColor color, PieceType type, int x, int y){
        this.color = color;
        this.type = type;
        setLocation(x, y);
        marked = false;
    }

    public void setLocation(int x, int y)
    {
        if(x < 0) this.x = 0;
        else if(x > 7) this.x = 7;
        else this.x = x;

        if(y < 0) this.y = 0;
        else if(y > 7) this.y = 7;
        else this.y = y;
    }
```

```
    public PlayerColor getColor()
    {
        return this.color;
    }

    public PieceType getType()
    {
        return this.type;
    }

    public boolean getMarked()
    {
        return this.marked;
    }

    public int getX()
    {
        return this.x;
    }

    public int getY()
    {
        return this.y;
    }

    public void mark()
    {
        this.marked = true;
    }

    public void unmark()
    {
        this.marked = false;
    }


    // mark tiles this piece could action.
    public void markTiles()
    {}

    // unmark tiles this piece could action.
    public void unmarkTiles()
    {}

    // do action to other piece. (by location)
    public void doActionTo(int x, int y)
    {}

    // return this piece can move to location.
    public boolean canMove(int x, int y)
    {
        return false;
    }

    // return this piece can attact to location.
    public boolean canAttack(int x, int y)
    {
        return false;
    }
}
```

바꾼 부분 2.

InitiateBoard 메소드.

앞에서 Piece 를 재정의 한 것 때문에, Piece 를 처음 생성할 때 좌표 정보를 입력해주어야 한다.

또한 코드에서 Piece class 를 상속받는 derived class 들을 이용하므로,

해당 기물의 class 들로 Piece 를 선언하여 초기화 하였다.

EmptyPiece : 아무것도 없는 공간을 나타내는 derived class.

xxxPiece : 해당 기물을 나타내는 derived class. (룩, 비숍, 폰 ...)

```
public void initiateBoard(){        //modified : for make derived class instead of base class Piece.
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++) setIcon(i, j, new EmptyPiece(i, j));          //modified for make derived class empty Piece.
    }
    setIcon(0, 0, new RookPiece(PlayerColor.black, PieceType.rook, 0, 0));
    setIcon(0, 1, new KnightPiece(PlayerColor.black, PieceType.knight, 0, 1));
    setIcon(0, 2, new BishopPiece(PlayerColor.black, PieceType.bishop, 0, 2));
    setIcon(0, 3, new QueenPiece(PlayerColor.black, PieceType.queen, 0, 3));
    setIcon(0, 4, new KingPiece(PlayerColor.black, PieceType.king, 0, 4));
    setIcon(0, 5, new BishopPiece(PlayerColor.black, PieceType.bishop, 0, 5));
    setIcon(0, 6, new KnightPiece(PlayerColor.black, PieceType.knight, 0, 6));
    setIcon(0, 7, new RookPiece(PlayerColor.black, PieceType.rook, 0, 7));
    for(int i=0;i<8;i++){
        setIcon(1, i, new PawnPiece(PlayerColor.black, PieceType.pawn, 1, i));
        setIcon(6, i, new PawnPiece(PlayerColor.white, PieceType.pawn, 6, i));
    }
    setIcon(7, 0, new RookPiece(PlayerColor.white, PieceType.rook, 7, 0));
    setIcon(7, 1, new KnightPiece(PlayerColor.white, PieceType.knight, 7, 1));
    setIcon(7, 2, new BishopPiece(PlayerColor.white, PieceType.bishop, 7, 2));
    setIcon(7, 3, new QueenPiece(PlayerColor.white, PieceType.queen, 7, 3));
    setIcon(7, 4, new KingPiece(PlayerColor.white, PieceType.king, 7, 4));
    setIcon(7, 5, new BishopPiece(PlayerColor.white, PieceType.bishop, 7, 5));
    setIcon(7, 6, new KnightPiece(PlayerColor.white, PieceType.knight, 7, 6));
    setIcon(7, 7, new RookPiece(PlayerColor.white, PieceType.rook, 7, 7));
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++) unmarkPosition(i, j);
    }
    onInitiateBoard();
}
```