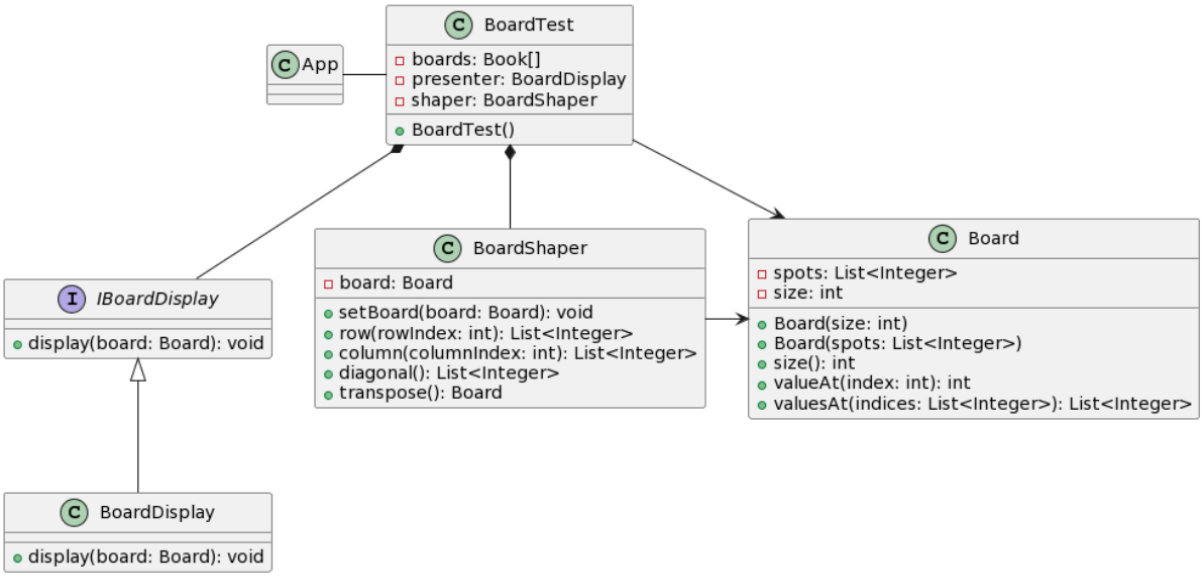
LAB1

소프트웨어학과 32201817 박정운

Lab1에 기재된 클래스 다이어그램에 맞춰 구현하였습니다.



[Board]

```
public class Board {

    private int size;
    private List<Integer> spots = new ArrayList<>();
```

필드 값인 size(=n)와 spots를 private로 선언하고, 이를 불러오거나 처리하는 메소드를 public으로 구현하여 캡슐화 하였습니다.

```
//constructor - size
public Board(int size){
    this.size = size;
    for(int i=1; i<=size*size; i++){
        this.spots.add(i);
    }
}

//constructor - spots(List<Integer>)
public Board(List<Integer> spots){
    this.size = (int)Math.sqrt((double)spots.size());
    this.spots = spots;
}
```

생성자는 size를 받아오는 것과 spots를 직접 받아오는 것이 있습니다. Size를 받을 경우 for loop를 이용해 spots를 초기화하고, spots를 받아오는 경우 size값을 초기화합니다.

```java
// return size
public int size(){ return size; }

// return spots.get(index)
public int valueAt(int index){ return this.spots.get(index); }

//return spots
public List<Integer> spots() { return this.spots; }
```

size와 spots를 return하는 getter 함수들과 index 값에 있는 value를 받아오는 함수를 public으로 구현하였습니다. 이를 통해 board의 필드에 접근하는 연산을 처리합니다.

[BoardShaper]

```java
public class BoardShaper {

    private Board board;

    public void setBoard(Board board){ this.board = board; }
```

BoardShaper는 필드 값으로 board를 가집니다. 이 때, BoardShaper 내부에서 새로운 Board 객체를 생성하는 것이 아니라 생성자를 통해 board를 주입 받아 초기화합니다. 주입 받는 객체는 어떤 size의 board라도 모두 가능합니다.

```java
public List<Integer> row(int rowIndex){
    List<Integer> rowList = new ArrayList<>();

    for(int i=0; i<board.size(); i++){
        rowList.add(board.valueAt(board.size()*(rowIndex-1)+i));
    }

    return rowList;
}

public List<Integer> column(int columnIndex){
    List<Integer> columnList = new ArrayList<>();

    for(int i=0; i<board.size(); i++){
        columnList.add(board.valueAt((columnIndex-1)+board.size()*i));
    }

    return columnList;
}

public List<Integer> diagonal(){
    List<Integer> diagonalList = new ArrayList<>();

    for(int i=0; i<board.size(); i++){
        diagonalList.add(board.valueAt(i*(board.size()+1)));
    }

    return diagonalList;
}
```

row, column, diagonal에 있는 값들을 저장해 하나의 List로서 반환하는 메소드입니다. Board의 size와 파라미터 값을 변수로 하는 수식을 통해 결과값을 저장하기 때문에 board의 size에 상관없이 3X3, 4X4 등의 board를 모두 처리할 수 있습니다.

```java
public Board transpose(){
    List<Integer> transposeList = new ArrayList<>();
    for(int i=1; i<=board.size(); i++){
        for(int j=0; j<board.size(); j++){
            transposeList.add(i+j*board.size());
        }
    }
    return new Board(transposeList);
}
```

Board를 transpose 하는 함수입니다. i+j*size라는 수식을 통해 값을 transpose 하고 이를 이용해 새 board 객체를 return 합니다.

[IBoardDisplay & BoardDisplay]

```java
public interface IBoardDisplay {
    public void display(Board board);
}
```

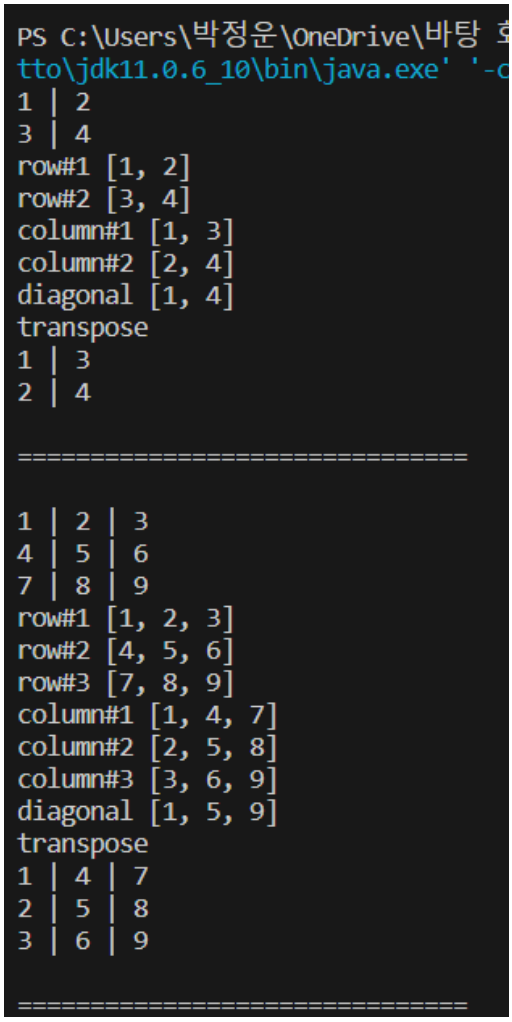
```java
public class BoardDisplay implements IBoardDisplay{

    @Override
    public void display(Board board) {
        for(int i=0; i<board.size()*board.size(); i++){
            System.out.print(board.spots().get(i));
            if((i+1)%board.size()==0) System.out.println();
            else System.out.print(s:" | ");
        }
    }
}
```

인터페이스 IBoardDisplay를 implements한 BoardDisplay class입니다. Board의 spots 값이 2차원이 아닌 1차원이기 때문에 2중 for문 대신 (i+1) 값이 size의 배수일 때 한 칸 내려간다는 조건을 두고 값을 출력합니다.

[출력 결과]

출력 결과는 다음과 같습니다.

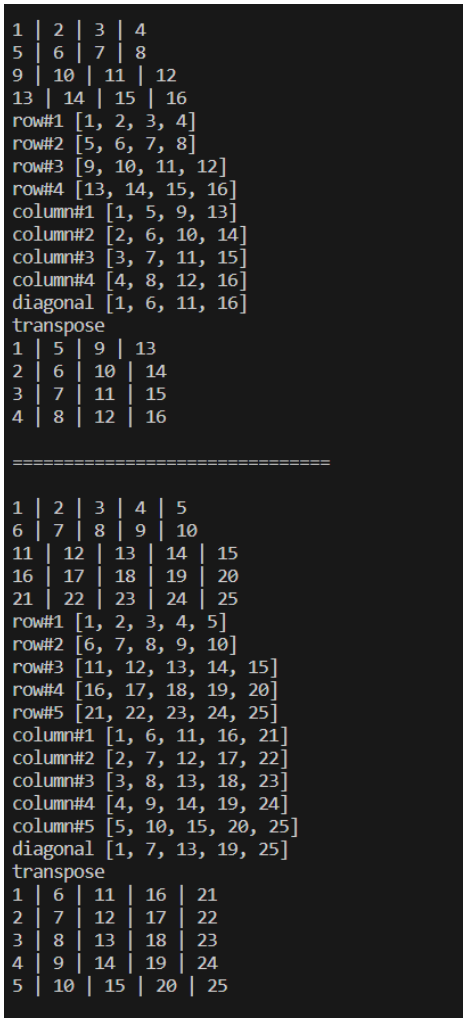


```
PS C:\Users\박정운\OneDrive\바탕 호
tto\jdk11.0.6_10\bin\java.exe' '-c
1 | 2
3 | 4
row#1 [1, 2]
row#2 [3, 4]
column#1 [1, 3]
column#2 [2, 4]
diagonal [1, 4]
transpose
1 | 3
2 | 4

==============================

1 | 2 | 3
4 | 5 | 6
7 | 8 | 9
row#1 [1, 2, 3]
row#2 [4, 5, 6]
row#3 [7, 8, 9]
column#1 [1, 4, 7]
column#2 [2, 5, 8]
column#3 [3, 6, 9]
diagonal [1, 5, 9]
transpose
1 | 4 | 7
2 | 5 | 8
3 | 6 | 9

==============================
```



```
1 | 2 | 3 | 4
5 | 6 | 7 | 8
9 | 10 | 11 | 12
13 | 14 | 15 | 16
row#1 [1, 2, 3, 4]
row#2 [5, 6, 7, 8]
row#3 [9, 10, 11, 12]
row#4 [13, 14, 15, 16]
column#1 [1, 5, 9, 13]
column#2 [2, 6, 10, 14]
column#3 [3, 7, 11, 15]
column#4 [4, 8, 12, 16]
diagonal [1, 6, 11, 16]
transpose
1 | 5 | 9 | 13
2 | 6 | 10 | 14
3 | 7 | 11 | 15
4 | 8 | 12 | 16

============================

1 | 2 | 3 | 4 | 5
6 | 7 | 8 | 9 | 10
11 | 12 | 13 | 14 | 15
16 | 17 | 18 | 19 | 20
21 | 22 | 23 | 24 | 25
row#1 [1, 2, 3, 4, 5]
row#2 [6, 7, 8, 9, 10]
row#3 [11, 12, 13, 14, 15]
row#4 [16, 17, 18, 19, 20]
row#5 [21, 22, 23, 24, 25]
column#1 [1, 6, 11, 16, 21]
column#2 [2, 7, 12, 17, 22]
column#3 [3, 8, 13, 18, 23]
column#4 [4, 9, 14, 19, 24]
column#5 [5, 10, 15, 20, 25]
diagonal [1, 7, 13, 19, 25]
transpose
1 | 6 | 11 | 16 | 21
2 | 7 | 12 | 17 | 22
3 | 8 | 13 | 18 | 23
4 | 9 | 14 | 19 | 24
5 | 10 | 15 | 20 | 25
```