

package Tenth;

public class Matrix {

private int row, col;

private int[][] mat;

정답자선과 2중배열

public Matrix(int row, int col) { 생성자

this.row = row;

this.col = col;

mat = new int[row][col];

} 2중배열 mat → row × col 의 행렬!

/*

Matrix wrong = new Matrix(arr1, 4);

Matrix A = new Matrix(arr1, 3, 3);

Matrix B = new Matrix(arr2, 3);

*/

public Matrix(int[][] arr, int row) { 2중배열과 row 값을 알기 위한

if(checkAvailable(arr, row)) { 알기 위한 method

mat = new int[row][col];

생성자를 위한 메소드?
3중배열에 존재!

// 1 2 3 4 ==> 3 by 4

// 2 4 6 8 1 2 3

// 3 6 9 12 4 2 4

// 6 8 3

// 6 9 12

int totalLen = row * col; 행렬의 총길이 = 행 × 열

int[] tmp = new int[totalLen];

//System.out.println("arr.length = " + arr.length);

//System.out.println("arr[0].length = " + arr[0].length);

for(int i = 0; i < arr.length; i++) {

for(int j = 0; j < arr[0].length; j++) {

tmp[i * arr[0].length + j] = arr[i][j];

//System.out.printf("tmp[%d] = %d\n",

// i * arr[0].length + j,

// tmp[i * arr[0].length + j]);

}

}

for(int i = 0; i < row; i++) {

for(int j = 0; j < col; j++) {

mat[i][j] = tmp[i * col + j];

}

}

}

}

* Question Mark.
문장어 해석.
명제부분
부연설명

→ 상수처리
→ 2번 사진

checkDivideElement 함수.

0번 제 행의 길이만큼 설정

왜? Col의 값을 알기 위해서 때문.

temp 배열만

상수값들은 i, j로 이중배열 시키기 위해서.

모든 이항에 나오며 다음장에 설명되어 있음.

우리가 원하는 mat 이중배열 도출 위한 식.

```
public Matrix(int[] arr, int row) {
```

```
    if (checkAvailable(arr, row)) {
```

```
        mat = new int[row][col];
```

```
        for (int i = 0; i < row; i++) {
```

```
            for (int j = 0; j < col; j++) {
```

```
                // 0 ~ 8:
```

```
                // i = 0 ~ 2 x
```

```
                // j = 0 ~ 2 x
```

```
                // i + j = 0 ~ 4 x
```

```
                // (i + 1) = 1 ~ 3 x
```

```
                // (i + 1) * j = 0 ~ 6 x
```

```
                // (i + 1) * (j + 1) = 1 ~ 9 x
```

```
                // (i + 1) * (j + 1) - 1 = 0 ~ 8 x
```

```
                // (i + 1) * 3 + j = 3 ~ 9 + j x
```

```
                // i * 3 + j = 0 ~ 8 o
```

```
                mat[i][j] = arr[i * col + j];
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
public Matrix(int[] arr, int row, int col) {
```

```
    if (checkAvailable(arr, row, col)) {
```

```
        System.out.println("A 처리 테스트");
```

```
        // 실제로 이 매서드 내의 모든 코드는
```

```
        // 단일 배열에서 넘어온 값들을 행렬로 변환한다는 취지를 가짐
```

```
        // 그러므로 동일하게 중복되는 코드들이 발생할 것이고
```

```
        // 별도의 매서드로 분리하여 관리할 수 있음
```

```
        mat = new int[row][col];
```

```
        for (int i = 0; i < row; i++) {
```

```
            for (int j = 0; j < col; j++) {
```

```
                mat[i][j] = arr[i * col + j];
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
private boolean checkDivideElement(int len, int row) {
```

```
    if (len % row == 0) {
```

```
        this.row = row;
```

```
        this.col = len / row;
```

```
    } else {
```

```
        System.out.printf("행렬로 변환할 수 없습니다.\n");
```

```
        System.out.printf("올바른 차원을 입력하세요.\n");
```

```
        System.out.printf("혹은 적절한 숫자(행)를 입력하세요\n");
```

```
        return false;
```

```
    }
```

```
    return true;
```

```
}
```

Matrix A.

성능과 클래스.

⇒ 배열 → 2중배열에 넣기 위한

→ 2중배열로 동적한다

int [] arr = {0, 1, 2, 3, 4, 5, 6, 7, 8}

= 1 ~ 9까지

→ 0 ~ 8까지 배열 필요.

Matrix A 처리한
class.

→ 분리/분리가 쉬워짐.

→ 예외처리를 위한
Method.

* return false
- false면 처리할 수 없는 상황
return true
if 가 있다면 true로
다루는 방법이다.

private boolean checkAvailable(int[] arr, int row, int col) { 1중배열 회상원한 class 3번째.

int len = arr.length;

boolean res = (len == row * col ? true : false);

if(res) {

this.row = row;

this.col = col;

}

return res;

}

checkDivideElement class과
같은 것.

private boolean checkAvailable(int[][] arr, int num) { 이중배열 제공한 조건.

int row = arr.length;

int col = arr[0].length;

int len = row * col;

// System.out.printf("[] row = %d, col = %d\n", row, col);

/*

if(len % row == 0) {

this.row = row;

this.col = len / row;

} else {

System.out.printf("행렬로 변환할 수 없습니다.\n");

System.out.printf("올바른 차원을 입력하세요.\n");

System.out.printf("혹은 적절한 숫자(행)를 입력하세요\n");

return false;

}

*/

return checkDivideElement(len, num);

}

→ 2H num은 쓰지?
row와 같은가?

같은 결과로
없는 식이지만
class를 나눴?

private boolean checkAvailable(int[] arr, int row) {

int len = arr.length;

/*

if(len % row == 0) {

this.row = row;

this.col = len / row;

} else {

System.out.printf("행렬로 변환할 수 없습니다.\n");

System.out.printf("올바른 차원을 입력하세요.\n");

System.out.printf("혹은 적절한 숫자(행)를 입력하세요\n");

return false;

}

*/

return checkDivideElement(len, row);

}

private boolean checkDimension(Matrix mat) {

int row = mat.getRow();

int col = mat.getCol();

return (this.row == row) && (this.col == col);

}

private boolean checkDimension(Matrix A, Matrix B) {

int Arow = A.getRow();

int Brow = B.getRow();

int Acol = A.getCol();

int Bcol = B.getCol();

return (Arow == Brow) && (Acol == Bcol);

}

public void addMatrix(Matrix mat) {

if(checkDimension(mat)) {

int[][] srcMat = mat.getMat();

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

this.mat[i][j] = this.mat[i][j] + srcMat[i][j];

}

}

}

}

public void addMatrix(Matrix A, Matrix B) {

if(checkDimension(A, B)) {

int[][] matA = A.getMat();

int[][] matB = B.getMat();

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

mat[i][j] = matA[i][j] + matB[i][j];

}

}

}

}

public void subMatrix(Matrix mat) {

if(checkDimension(mat)) {

int[][] srcMat = mat.getMat();

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

this.mat[i][j] = this.mat[i][j] - srcMat[i][j];

}

}

}

}

Matrix mat?

Matrix class를 가져왔다.

EX Matrix A B/C

이전 클래스의
값을 가져가기 위해
getter 사용.

2개의 클래스는
다른 클래스의 멤버와
공유할 수 있다.

위:

아래:

더하는 메소드. (A+B)

B

private mat?

private mat?

사실상
이런
방법.

더하는 메소드 (R=A+B)

빼는 method
나오는 위와 같다.

```
208
209 public void subMatrix(Matrix A, Matrix B) {
210     if(checkDimension(A, B)) {
211         int[][] matA = A.getMat();
212         int[][] matB = B.getMat();
213
214         for (int i = 0; i < row; i++) {
215             for (int j = 0; j < col; j++) {
216                 mat[i][j] = matA[i][j] - matB[i][j];
217             }
218         }
219     }
220 }
```

~~// n by m * n by m - 성립 불가~~
~~// n by m * m by n: n by n~~

$3 \times 3 \times 3 \times 2 = 0K. \rightarrow 3 \times 2$

```
224 public boolean checkMulDimension(Matrix A, Matrix B) {
225     int Brow = B.getRow();
226     int Acol = A.getCol();
227
228     return (Brow == Acol);
229 }
```

```

231 public void mulMatrix(Matrix A, Matrix B) {
232     if(checkMulDimension(A, B)) {
233         int[][] matA = A.getMat();
234         int[][] matB = B.getMat();
235
236         // 00  01  02      00  01  02
237         // 10  11  12      10  11  12
238         // 20  21  22      20  21  22
239         // 00 * 00 + 01 * 10 + 02 * 20: [0][0]
240         // 00 * 01 + 01 * 11 + 02 * 21: [0][1]
241         // 00 * 02 + 01 * 12 + 02 * 22: [0][2]
242
243         // 10 * 00 + 11 * 10 + 12 * 20: [1][0]
244         // 10 * 01 + 11 * 11 + 12 * 21: [1][1]
245         // 10 * 02 + 11 * 12 + 12 * 22: [1][2]
246
247         // 20 * 00 + 21 * 10 + 22 * 20: [2][0]
248         // 20 * 01 + 21 * 11 + 22 * 21: [2][1]
249         // 20 * 02 + 21 * 12 + 22 * 22: [2][2]
250         mat[0][0] = matA[0][0] * matB[0][0] +
251             matA[0][1] * matB[1][0] +
252             matA[0][2] * matB[2][0];
253         mat[0][1] = matA[0][0] * matB[0][1] +
254             matA[0][1] * matB[1][1] +
255             matA[0][2] * matB[2][1];
256         mat[0][2] = matA[0][0] * matB[0][2] +
257             matA[0][1] * matB[1][2] +
258             matA[0][2] * matB[2][2];
259         // 10 * 00 + 11 * 10 + 12 * 20: [1][0]
260         // 10 * 01 + 11 * 11 + 12 * 21: [1][1]
261         // 10 * 02 + 11 * 12 + 12 * 22: [1][2]
262         mat[1][0] = matA[1][0] * matB[0][0] +
263             matA[1][1] * matB[1][0] +
264             matA[1][2] * matB[2][0];
265         mat[1][1] = matA[1][0] * matB[0][1] +
266             matA[1][1] * matB[1][1] +
267             matA[1][2] * matB[2][1];
268         mat[1][2] = matA[1][0] * matB[0][2] +
269             matA[1][1] * matB[1][2] +
270             matA[1][2] * matB[2][2];
271         // 20 * 00 + 21 * 10 + 22 * 20: [2][0]
272         // 20 * 01 + 21 * 11 + 22 * 21: [2][1]
273         // 20 * 02 + 21 * 12 + 22 * 22: [2][2]
274         mat[2][0] = matA[2][0] * matB[0][0] +
275             matA[2][1] * matB[1][0] +
276             matA[2][2] * matB[2][0];
277         mat[2][1] = matA[2][0] * matB[0][1] +
278             matA[2][1] * matB[1][1] +
279             matA[2][2] * matB[2][1];
280         mat[2][2] = matA[2][0] * matB[0][2] +
281             matA[2][1] * matB[1][2] +
282             matA[2][2] * matB[2][2];
283     }
284 }
285

```

행도 곱함!
 위를 곱한 식이

```

286     public void allocRandomMatrix() {
287         for(int i = 0; i < row; i++) {
288             for(int j = 0; j < col; j++) {
289                 mat[i][j] = (int)(Math.random() * 10);
290             }
291         }
292     }
293
294     public int getRow() {
295         return row;
296     }
297
298     public int getCol() {
299         return col;
300     }
301
302     public int[][] getMat() {
303         return mat;
304     }
305
306     // n by n 행렬의 판별식
307     // ex) 3 by 3
308     // 1   2   3
309     // 4   5   6 =====>
310     // 7   8   9
311     //
312     // 1 * {(5 * 9) - (6 * 8)} +
313     // 2 * {(6 * 7) - (4 * 9)} +
314     // 3 * {(4 * 8) - (5 * 7)}
315     // 이 결과가 0 이 아니면 역행렬이 존재한다.
316
317     public void printMatrix() {
318         for(int i = 0; i < row; i++) {
319             for(int j = 0; j < col; j++) {
320                 System.out.printf("%4d", mat[i][j]);
321             }
322             System.out.println("");
323         }
324     }
325 }

```

int[row][col] array
에 0~10까지 랜덤할당.

다행/비행/행렬
getter!

이 Class/method
주석을 위한 예시!