Matrix 행렬

date. 2021_01_13

3 by 4 ===> 4 by 3

1	2	3	4
2	4	6	8
3	6	9	12

1	2	3
4	2	4
6	8	3
6	9	12

```
//연산을 위해 행렬을 맞추는 과정
public Matrix(int[][] arr, int row) {
  if(change_for_element(arr, row)) {
    matrix = new int[row][col];
    int totalLen = row * col;
    int[] tmp = new int[totalLen];

    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];
      }
    }

    for(int i = 0; i < row; i++) {
      for(int j = 0; j < col; j++) {
        matrix[i][j] = tmp[i * col + j];
```

true!!

```
private boolean change_for_element
                        (int[][] arr, int um) {
  int row = arr.length;    //행은 배열의 길이
  int col = arr[0].length; //열은 배열의 0번째 의 길이
  int length = row * col;
  return is_avilable_matrix(length, num);
}
```

```
private boolean is_avilable_matrix
                        (int length, int row) {
  if(length % row == 0) {
    this.row = row;
    this.col = length / 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;
}
```

```
0 * 3 + 0 = 0
0 * 3 + 1 = 1
0 * 3 + 2 = 2
   ...
2 * 3 + 0 = 6
2 * 3 + 1 = 7
2 * 3 + 2 = 8
```

```
i : 0 0 0 1 1 1 2 2 2
j : 0 1 2 0 1 2 0 1 2
arr : 0 1 2 3 4 5 6 7 8
---------------------------------
i * col + j : 0 1 2 3 4 5 6 7 8
```

False가 나올경우 Error 메세지

행렬로 변환할 수 없습니다.
올바른 차원을 입력하세요.
혹은 적절한 숫자(행)를 입력하세요

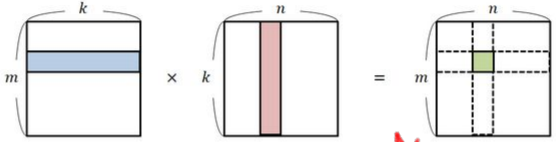
Matrix 행렬의 곱셈

```
public void mul_matrix(Matrix A, Matrix B) {

  if(check_mul_dimension(A, B)) {
    int[][] matA = A.get_matrix();
    int[][] matB = B.get_matrix();
```

```
public boolean check_mul_dimension
                        (Matrix A, Matrix B) {
  int B_row = B.get_row();
  int A_col = A.get_col();

  return (B_row == A_col);
}
```



M by K * K by N = M by N

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 \cdot a + 2 \cdot c & 1 \cdot b + 2 \cdot d \\ 3 \cdot a + 4 \cdot c & 3 \cdot b + 4 \cdot d \end{bmatrix}$$

3중 for문을 이용하여 행렬의 곱셈을 하는것도
물론 가능하지만 빠른 데이터 처리를 위해
하드코드로 작성하였다! **오른쪽 메모지 참조**
하지만, 3 by 3을 넘어가는 행렬은 ... 코드를 이용하자!

```
matrix[0][0] =
        matA[0][0] * matB[0][0]
      + matA[0][1] * matB[1][0]
      + matA[0][2] * matB[2][0];
matrix[0][1] =
        matA[0][0] * matB[0][1]
      + matA[0][1] * matB[1][1]
      + matA[0][2] * matB[2][1];
matrix[0][2] =
        matA[0][0] * matB[0][2]
      + matA[0][1] * matB[1][1]
      + matA[0][2] * matB[2][2];
matrix[1][0] =
        matA[1][0] * matB[0][0]
      + matA[1][1] * matB[1][0]
      + matA[1][2] * matB[2][0];
matrix[1][1] =
        matA[1][0] * matB[0][1]
      + matA[1][1] * matB[1][1]
      + matA[1][2] * matB[2][1];
matrix[1][2] =
        matA[1][0] * matB[0][2]
      + matA[1][1] * matB[1][2]
      + matA[1][2] * matB[2][2];
matrix[2][0] =
        matA[2][0] * matB[0][0]
      + matA[2][1] * matB[1][0]
      + matA[2][2] * matB[2][0];
matrix[2][1] =
        matA[2][0] * matB[0][1]
      + matA[2][1] * matB[1][1]
      + matA[2][2] * matB[2][1];
matrix[2][2] =
        matA[2][0] * matB[0][2]
      + matA[2][1] * matB[1][2]
      + matA[2][2] * matB[2][2];
```