```java
package Tenth;

public class Matrix {
    private int row, col;
    private int[][] mat;

    public Matrix(int row, int col) {
        this.row = row;
        this.col = col;

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

    /*
        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) {
        if(checkAvailable(arr, row)) {
            mat = new int[row][col];

            // 4 by 3        ===>      3 by 4
            // 1 2 3 4                 1 2 3
            // 2 4 6 8                 4 2 4
            // 3 6 9 12                6 8 3
            //                         6 9 12
            int totalLen = row * col;
            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];
                }
            }
        }
    }
```

```java
    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;
    }
```

```java
    private boolean checkAvailable(int[] arr, int row, int col) {
        int len = arr.length;

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

        if(res) {
            this.row = row;
            this.col = col;
        }

        return res;
    }

    private boolean checkAvailable(int[][] arr, int num) {
        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);
    }

    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);
    }
```

```java
    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];
                }
            }
        }
    }
```

```
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     }
221 
222     // n by m * n by m - 성립 불가
223     // n by m * m by n: n by n
224     public boolean checkMulDimension(Matrix A, Matrix B) {
225         int Brow = B.getRow();
226         int Acol = A.getCol();
227 
228         return (Brow == Acol);
229     }
```

3by3 x 3x2 = ok. → 3by2

```
    public void mulMatrix(Matrix A, Matrix B) {
        if(checkMulDimension(A, B)) {
            int[][] matA = A.getMat();
            int[][] matB = B.getMat();

            // 00   01   02      00   01   02
            // 10   11   12      10   11   12
            // 20   21   22      20   21   22
            // 00 * 00 + 01 * 10 + 02 * 20: [0][0]
            // 00 * 01 + 01 * 11 + 02 * 21: [0][1]
            // 00 * 02 + 01 * 12 + 02 * 22: [0][2]

            // 10 * 00 + 11 * 10 + 12 * 20: [1][0]
            // 10 * 01 + 11 * 11 + 12 * 21: [1][1]
            // 10 * 02 + 11 * 12 + 12 * 22: [1][2]

            // 20 * 00 + 21 * 10 + 22 * 20: [2][0]
            // 20 * 01 + 21 * 11 + 22 * 21: [2][1]
            // 20 * 02 + 21 * 12 + 22 * 22: [2][2]
            mat[0][0] = matA[0][0] * matB[0][0] +
                    matA[0][1] * matB[1][0] +
                    matA[0][2] * matB[2][0];
            mat[0][1] = matA[0][0] * matB[0][1] +
                    matA[0][1] * matB[1][1] +
                    matA[0][2] * matB[2][1];
            mat[0][2] = matA[0][0] * matB[0][2] +
                    matA[0][1] * matB[1][1] +
                    matA[0][2] * matB[2][2];
            // 10 * 00 + 11 * 10 + 12 * 20: [1][0]
            // 10 * 01 + 11 * 11 + 12 * 21: [1][1]
            // 10 * 02 + 11 * 12 + 12 * 22: [1][2]
            mat[1][0] = matA[1][0] * matB[0][0] +
                    matA[1][1] * matB[1][0] +
                    matA[1][2] * matB[2][0];
            mat[1][1] = matA[1][0] * matB[0][1] +
                    matA[1][1] * matB[1][1] +
                    matA[1][2] * matB[2][1];
            mat[1][2] = matA[1][0] * matB[0][2] +
                    matA[1][1] * matB[1][2] +
                    matA[1][2] * matB[2][2];
            // 20 * 00 + 21 * 10 + 22 * 20: [2][0]
            // 20 * 01 + 21 * 11 + 22 * 21: [2][1]
            // 20 * 02 + 21 * 12 + 22 * 22: [2][2]
            mat[2][0] = matA[2][0] * matB[0][0] +
                    matA[2][1] * matB[1][0] +
                    matA[2][2] * matB[2][0];
            mat[2][1] = matA[2][0] * matB[0][1] +
                    matA[2][1] * matB[1][1] +
                    matA[2][2] * matB[2][1];
            mat[2][2] = matA[2][0] * matB[0][2] +
                    matA[2][1] * matB[1][2] +
                    matA[2][2] * matB[2][2];
        }
    }
```

```java
    public void allocRandomMatrix() {
        for(int i = 0; i < row; i++) {
            for(int j = 0; j < col; j++) {
                mat[i][j] = (int)(Math.random() * 10);
            }
        }
    }

    public int getRow() {
        return row;
    }

    public int getCol() {
        return col;
    }

    public int[][] getMat() {
        return mat;
    }

    // n by n 행렬의 판별식
    // ex) 3 by 3
    // 1    2    3
    // 4    5    6  =====>
    // 7    8    9
    //
    // 1 * {(5 * 9) - (6 * 8)} +
    // 2 * {(6 * 7) - (4 * 9)} +
    // 3 * {(4 * 8) - (5 * 7)}
    // 이 결과가 0 이 아니면 역행렬이 존재한다.

    public void printMatrix() {
        for(int i = 0; i < row; i++) {
            for(int j = 0; j < col; j++) {
                System.out.printf("%4d", mat[i][j]);
            }
            System.out.println("");
        }
    }
}
```

int[row][col] size 에 0~10까지 랜덤값 넣어.

덧셈/뺄셈/곱셈 위한 getter!

모든 Class/method 출력을 위한 메소드!