디지털컨버전스 기반 UXUI Front 전문 개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - JungHyun LEE(이정현)
akdl911215@naver.com

1) 10 일차 Matrix 주석으로 해석 및 의문점 작성

```java
package Eleventh;

public class Matrix {
    // row 는 정수형 행, col 은 정수형 열
    private int row, col;
    // 정수형 2차배열 방식의 mat를 만든다.
    private int[][] mat;


    // row(행) 값과 col(열) 값을 입력한다
    public Matrix(int row, int col) {
        // 입력받은 row 값을 현재 메소드의 this.row에 대입한다.
        this.row = row;
        // 입력받은 col 값을 현재 메소드의 this.col에 대입한다.
        this.col = col;

        // 입력받은 this.row와 this.col을
        // new int[row][col]; 대입시키고 mat에 대입한다.
        mat = new int[row][col];
    }
// 2차열 배열과 row(행) 값을 입력받는다.
public Matrix(int[][] arr, int row) {
    // (checkAvailable(arr, row)가
    // 참일 경우에 if문을 작동한다
    if(checkAvailable(arr, row)) {

        // 입력받은 row(행) 값과 col 값을 2차열 배열형태 mat 에 대입한다
        // new int[row][col]; = n by n ( 몇 대 몇 행렬로 할것인가)을 결정하기 위해서
        // ex) 9개 값을 가진 배열을 3 x 3 할지, 9 x 1 할지, 1 x 9 할지 정하기 위해서
        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

        // totalLen = row(행) * col(열) 을 곱한게 전체배열의 값이기 때문이다.
        int totalLen = row * col;
        // 1차원 배열 tmp = new int[totalLen];을 넣으면 전체 배열의 값이 나온다.
        int[] tmp = new int[totalLen];
        // for문을 0부터 arr.legth -1 값까지 1씩 증가한다
        // length 값은 어디에?
        for(int i = 0; i < arr.length; i++) {
            // for문을 0부터 arr.legth -1 값까지 1씩 증가한다
            // length 값은 어디에?
            for(int j = 0; j < arr[0].length; j++) {
                // tmp[i * arr[0].length + j] = arr[i][j] 의
                // 해당 배열의 값을 대입한다.
                // length 값은 어디에?
                //    length??????????????????
                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]);
            }
        }

        // for문을 0부터 row(행) -1 값까지 1씩 증가한다
        for(int i = 0; i < row; i++) {
            // for문을 0부터 col(열) -1 값까지 1씩 증가한다
            for(int j = 0; j < col; j++) {
                //mat[i][j] = tmp[i * col + j];의 배열의 값을 대입한다.
                //   col???????
                mat[i][j] = tmp[i * col + j];
            }
        }
    }
}

// 1차원 배열 arr 값과 row(행) 값을 입력한다.
public Matrix(int[] arr, int row) {
    // checkAvailable(arr, row)이 입력되어서 true일 때 진입한다.
    if(checkAvailable(arr, row)) {
        //mat 에 row 와 col 입력 값을 대입한다.
        mat = new int[row][col];

        // i는 0 ~ row-1값까지 반복하며 1씩 증가한다
        for (int i = 0; i < row; i++) {
            // j는 0 ~ col-1값까지 반복하며 1씩 증가한다
            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] 에 다가 arr[i * col + j]배열안의 값을 대입한다.
                mat[i][j] = arr[i * col + j];
            }
        }
    }
}
```

```java
    // 1차원 배열 arr 과 row(행) 값 col(열) 값을 입력한다.
    public Matrix(int[] arr, int row, int col) {
        // checkAvailable(arr, row, col) 입력 되어서 trur 일 때만 진입한다.
        if(checkAvailable(arr, row, col)) {
            // System.out.println("A 처리 테스트");는 실제로
            // if문이 진행되는가를 확인을 위한 테스트 출력이다.
            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 에다가는 입력받은 new int[row][col]; 을 대입한다.
            mat = new int[row][col];

            // i는 0 ~ row-1값까지 반복하며 1씩 증가한다
            for (int i = 0; i < row; i++) {
                // j는 0 ~ col-1값까지 반복하며 1씩 증가한다
                for (int j = 0; j < col; j++) {
                    // mat[i][j] 값에다가 입력받은 arr[i * col + j] 을 대입한다.
                    // arr[i * col + j] 을 해설하면 i=0,j=0,col=3일 경우에
                    // arr[0 * 3 + 0] 이므로 arr[0]이 된다.
                    // arr[0]은 arr1에서 0번째 배열자리는 값이 1이기 때문에
                    // mat[0][0] 자리에는 1값을 대입하게 된다.
                    mat[i][j] = arr[i * col + j];
                }
            }
        }
    }
```

```java
    // boolean 형식으로 받아야하는 checkDivideElement에다가
    // 1ne 값과 row(행) 값을 입력한다.
    // boolean 형식은 참과 거짓으로 구분이 되야한다.
    // 하지만 참과 거짓을 구별하지 않고 len과 row를 입력받기 때문에
    // 안으로 들어가봐야 한다.
    private boolean checkDivideElement(int len, int row) {
        // len % row == 0 일 경우와 그 외의 경우로 진행된다
        // len % row == 0 일 경우에는
        // this.row = row;
        // this.col = len / row; 를 진행한다.
        // this.col 의 경우 len / row 를 해주는 이유 설명 :
        // 배열 = row(행) * col(열) 이다.
        // 그러므로 col(열)을 모를경우에는 배열의 총길이 / row(행)을
        // 하면 열을 구할 수 있기 때문에 진행된다.

        // return true는 else 아닐시에 if문을 실행하지 않더라도 실행된다.
        // 하지만 if else이기 때문에 if이거나 else가 실행될듯 하다???
        if(len % row == 0) {
            this.row = row;
            this.col = len / row;

            // else는 배열의 총길이와 row(행)값이 나눠지지 않을 경우에 작동한다.
            // 왜냐하면 9자리의 값을 가진 배열일 경우에 1, 3, 9 가 아닌
            // 다른수로 나눠진다면 정렬을 해줄 수 없기 때문이다.
            // ex) {1, 2, 3, 4, 5, 6, 7, 8, 9}; 을 4로 나눈다면
            //     1  2  3  4
            //     5  6  7  8
            //     9
            // 이런식으로 나눠지기 때문이다.
        } 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");
            //else가 실행되면 return은 거짓으로 반환된다.
            return false;
        }
```

```
176
177         // else외의 경우의 return은 참으로 반환 된다.
178         return true;
179     }
180
181     // boolean 형식으로 받아야하는 checkAvailable 에다가
182     // 1차원 배열 arr 과 row(행) 값과 col(열) 을 입력한다.
183 @   private boolean checkAvailable(int[] arr, int row, int col) {
184         // int len 에다가 arr.length; 값을 대입한다.
185         // length 값은 아마 입력받은 1차원 배열 값의 총길이를 넣는듯 하다???????
186         int len = arr.length;
187
188         // boolean 형식의 res 값에다가 len(길이) 와 row(행) * col(열)
189         // 의 길이가 같냐? 그리고 그것이 참 또는 거짓이냐? 판별 가능하냐를 대입한다.
190         boolean res = (len == row * col ? true : false);
191
192         // res가 true일 경우 ( 즉, 길이가 같은 경우 ) 에 실행한다.
193         if(res) {
194             // this.row 값에 row(행) 값을 대입한다
195             this.row = row;
196             // this.col 값에 col(열) 값을 대입한다
197             this.col = col;
198
199         }
200
201         // 반환은 res을 한다.
202         return res;
203     }
204
```

```
205     // boolean 형식으로 받아야하는 checkAvailable 에다가
206     // 2차원 배열 arr 과 num 을 입력한다.
207     // 갑자기 입력값을 num 으로 받는 것일까?????
208 @   private boolean checkAvailable(int[][] arr, int num) {
209         // 입력받은 2차원 배열의 길이를 row로 대입한다.
210         int row = arr.length;
211         // 입력받은 2차원 배열의 0번자리의 길이를 col로 대입한다.
212         // 2차원 배열을 입력받았는데 1차원 배열의 0번자리라는건 뭐지??????????
213         int col = arr[0].length;
214         // len 에다가 row * col 값을 대입하낟.
215         int len = row * col;
216         // System.out.printf("[][] row = %d, col = %d\n", row, col);
217
218         /*
219         if(len % row == 0) {
220             this.row = row;
221             this.col = len / row;
222         } else {
223             System.out.printf("행렬로 변환할 수 없습니다.\n");
224             System.out.printf("올바른 차원을 입력하세요.\n");
225             System.out.printf("혹은 적절한 숫자(행)를 입력하세요\n");
226             return false;
227         }
228          */
229
230         // return 은 checkDivideElement 값으로 반환한다.
231         // checkDivideElement시에는 len, num값 포함.
232         // 아마 메소드 제목에서 입력받은 num값과
233         // 입력받은 2차원 배열의 입력값을 같이 반환하기 위한듯 하다.
234         return checkDivideElement(len, num);
235     }
236
```

```java
    // boolean 형식으로 받아야하는 checkAvailable 에다가
    // 1차원 배열 arr 과 row 을 입력한다.
    private boolean checkAvailable(int[] arr, int row) {
        // int len 에다가 1차원 배열의 값을 arr.length;넣고 len에 대입한다.
        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 은 checkDivideElement 을 반환한다.
        // len, row는 메소드 제목에 있는 1차원 배열의 길이와 row값을
        // 같이 반환한다.
        return checkDivideElement(len, row);
    }

    // boolean 형식으로 받아야하는 checkDimension 에다가
    // Matrix mat을 입력한다.
    // Matrix mat 란???????
    private boolean checkDimension(Matrix mat) {
        // int row 에다가 mat.getRow(); 대입한다.
        // 그것은
        // public int getRow() {
        //       return row;
        //   } 을 호출해서 넣는것과 같다.
        // 밑에 row, col 도 똑같이 get을 호출
        int row = mat.getRow();
        int col = mat.getCol();
```

```java
        // Matrix의 this.row 값과 get으로 받은 row 값이 같은 경우
        // && Matrix의 this.col 값과 get으로 받은 col이 같은 경우
        // return을 checkDimension의
        // int row = mat.getRow();
        // int col = mat.getCol();
        // 을 반환한다.
        return (this.row == row) && (this.col == col);
    }

    // boolean 형식으로 받아야하는 checkDimension 에다가
    // Matrix A 값 과 Mattrix B 값을 입력한다 ???????
    private boolean checkDimension(Matrix A, Matrix B) {
        // int Arow 에다가 getRow값을 대입한다.
        // A. 은 뭐지???????????????
        // 아마도 출력할때 Matrix A와 Matrix B에 따로 출력되기
        // 위해서 구분한것으로 생각된다.
        // 밑에 함수도 동일하게 작동한다.
        int Arow = A.getRow();
        int Brow = B.getRow();
        int Acol = A.getCol();
        int Bcol = B.getCol();

        // Arow 값과 Brow 값이 동일하며 Acol 값과 Bcol 값이
        // 동일 할경우에 반환된다.
        return (Arow == Brow) && (Acol == Bcol);
    }

    // 반환값이 없고 addMatrix에 Matrix mat 값을 입력 한다???
    public void addMatrix(Matrix mat) {
        // if 문이 checkDimension(mat) 값일 때 작동한다.
        if(checkDimension(mat)) {
            // int[][] srcMat 에다가 호출한 mat.getMat(); 값을 대입한다.
            // getMat 는 private int[][] mat; 이며 Matrix의 mat의 2차원
            // 배열 값에 저장된 row(행) col(열) 값을
            // srcMat 에 그대로 대입한다.
            int[][] srcMat = mat.getMat();
```

```java
            for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    // for문이 i 3번 j 3번 도는 동안 밑에 i , j는
                    // 00 ~ 22까지 배열의 값을 저장할 수 있게 된다.

                    // 그러므로 this.mat[i][j] + srcMat[i][j]; 을 풀어보자면
                    // srcMat[i][j] 값에는 mat.getMat()값 +
                    // 호출된 mat 값은 this.mat[i][j] 대입 된후
                    // this.mat[i][j] 모두 더해져서 대입된다.
                    this.mat[i][j] = this.mat[i][j] + srcMat[i][j];
                }
            }
        }
    }

    // 반환값이 없고 addMatrix에 Matrix A, Matrix B 값을 입력 한다???
    // 동일한 크기의 행렬을 더하기 위한 함수
    public void addMatrix(Matrix A, Matrix B) {
        // if문이 checkDimension(A, B) 가 참일 경우에 시작된다.
        if(checkDimension(A, B)) {
            // int[][] matA 2차원 배열을 만들어 = A.getMat를 대입한다.
            int[][] matA = A.getMat();
            // int[][] matB 2차원 배열을 만들어 = B.getMat를 대입한다.
            int[][] matB = B.getMat();

            for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    // for문이 작동하는 동안 matA , matB 의 해당 배열 자리의 값들이 더해져서
                    // mat[i][j] 의 해당 자리에 값들이 대입 된다.
                    mat[i][j] = matA[i][j] + matB[i][j];
                }
            }
        }
    }
```

```java
    // 반환값이 없고 subMatrix Matrix mat 값을 입력 한다???
    // 동일한 크기의 행렬을 빼기 위한 함수
    public void subMatrix(Matrix mat) {
        // if문이 checkDimension(mat) 가 참일 경우에 시작된다.
        if(checkDimension(mat)) {
            // int[][] srcMat 2차원 배열을 만들어 = mat.getMat를 대입한다.
            int[][] srcMat = mat.getMat();

            for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    // for문이 작동하는 동안 matA , matB 의 해당 배열 자리의 값들이 빼져서
                    // mat[i][j] 의 해당 자리에 값들이 대입 된다.
                    this.mat[i][j] = this.mat[i][j] - srcMat[i][j];
                }
            }
        }
    }

    // 반환값이 없고 subMatrix에 Matrix A, Matrix B 값을 입력 한다???
    // 동일한 크기의 행렬을 빼기 위한 함수
    public void subMatrix(Matrix A, Matrix B) {
        if(checkDimension(A, B)) {
            // int[][] matA 2차원 배열을 만들어 = A.getMat를 대입한다.
            int[][] matA = A.getMat();
            // int[][] matB 2차원 배열을 만들어 = B.getMat를 대입한다.
            int[][] matB = B.getMat();

            for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    // for문이 작동하는 동안 matA , matB 의 해당 배열 자리의 값들이 빼져서
                    // mat[i][j] 의 해당 자리에 값들이 대입 된다.
                    mat[i][j] = matA[i][j] - matB[i][j];
                }
            }
        }
    }
```

```java
    // 불린형식의 checkMulDimension Matrix A, Matrix B 값을 입력 한다???
    // 불린형식은 true,false 을 구별하는 건데 메소드 제목에 없으므로
    // 안으로 들어가서 확인해야한다.
    public boolean checkMulDimension(Matrix A, Matrix B) {
        // int Brow 에 B.getRow(); 한 값을 대입한다.
        int Brow = B.getRow();
        // int Acol 에 A.getCol(); 한 값을 대입한다.
        int Acol = A.getCol();

        // Brow와 Acol의 값이 동일할 경우 반환한다.
        // int Brow = B.getRow();
        // int Acol = A.getCol();
        // 인데 아마 2개의 숫자가 동일해야지
        // 3 x 3 , 4 x 4 , 5 x 5 등 만들어 줄 수
        // 있기때문에 만든 메소드인듯 하다.
        return (Brow == Acol);
    }

    // 반환값이 없고 mulMatrix에 Matrix A, Matrix B 값을 입력 한다???
    // 동일한 크기의 행렬을 곱하기 위한 함수
    public void mulMatrix(Matrix A, Matrix B) {
        // if문은 (checkMulDimension(A, B) 가 참일때 작동한다.
        if(checkMulDimension(A, B)) {
            //  int[][] matA 에 A.getMat() 한 값을 대입한다
            int[][] matA = A.getMat();
            //  int[][] matB 에 B.getMat() 한 값을 대입한다
            int[][] matB = B.getMat();
```

```java
    // allocRandomMatrix 이 호출되면 실행한다.
    public void allocRandomMatrix() {
        for(int i = 0; i < row; i++) {
            for(int j = 0; j < col; j++) {
                // for문이 도는동안 mat[i][j] 의 해당 배열의 값에
                // 0에서부터 10미만의 값들이 저장된다.
                mat[i][j] = (int)(Math.random() * 10);
            }
        }
    }

    // getRow()가 호출될 때 Matrix의 row값을 반환한다
    public int getRow() {
        return row;
    }
    // getCol()가 호출될 때 Matrix의 col값을 반환한다
    public int getCol() {
        return col;
    }
    // getMat()가 호출될 때 Matrix의 mat값을 반환한다
    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.
```

```java
    //printMatrix가 호출되면 실행한다.
    public void printMatrix() {
        for(int i = 0; i < row; i++) {
            for(int j = 0; j < col; j++) {
                // printf는 4자리의 정수가 출력되며, mat[i][j]의
                // 배열의 해당 값들이 출력된다.
                System.out.printf("%4d", mat[i][j]);
            }
            System.out.println("");
        }
    }
}
```

2) MySeries에서의 질문

- final int ORDER 값을 준 이유가 뭔가요?

- 코드 위쪽에 public final int AND = 3값을 줬기 때문에 AND가 3값이 들어올때 작동하는게 맞나요???

```
73    // 반환값이 없는 printTwiceOrder문에 int형 oderNum1 입력,
74    // int형 oderNum2 입력, final int 형 ORDER 입력
75
76    // final int ORDER ????????
77    // final 은 고정값을 보수를 편리하게 하기 위해 사용되는데
78    // printTwiceOrder 에 int ORDER 가 아닌 fina int 를 사용한 이유가???
79    public void printTwiceOrder(int orderNum1, int orderNum2,
80                                final int ORDER) {
81        // int형 cnt에 1값을 대입
82        int cnt = 1;
83
84        // for문이 작동한다.
85        // i 는 입력된 start값에서 부터 end를 포함한
86        // 값까지 1씩 증가하며 작동한다.
87        for(int i = start; i <= end; i++) {
88            // OR 인지 AND 인지에 따라 다른 동작을 취해야함
89
90            // 스위치문은 해당하는 동작에만 작동하고 나머지는 무시한다.
91
92            // switch (ORDER) 는 ORDER 값이 들어올때 작동한다.
93            switch (ORDER) {
94                // 케이스 and일때 작동한다.
95                // 위에서 public final int AND = 3; 값을 줬기때문에
96                // 3의 숫자가 들어왔을때 작동할듯 하다 ????????
97                case AND:
98                    // if문은 i 값이 orderNum1 로 나눠서 나머지가 0이면서
99                    // i 값이 orderNum2 로 나눠서 나머지가 0일때 작동한다.
100                   if(i % orderNum1 == 0 && i % orderNum2 == 0) {
101                       // printf 는 i 값을 정수형을 3칸씩 출력한다.
102                       System.out.printf("%3d", i);
```