<문제은행 [4] >

1. 2 by 2 이중 배열을 초기화해서 아무 값이나 넣어보자.
2. 1번 문제에서 초기화한 배열을 출력해봅시다.
4. 1번 문제를 클래스화 해봅시다.

[main - MyArray - MatrixD2 or MatrixD3]

- main

1. MyArray 형식의 객체인 myArray 할당
2. dim값을 결정한 뒤 MyArray 클래스의 initMyArray 메소드 호출해 배열 생성
3. MyArray 클래스의 printArray 메소드 호출해 배열 출력

- MyArray 클래스

dimension 값과 비교할 변수 D2, D3 선언
클래스 변수 matD2, matD3 선언

```
int dimension;

final int D2 = 2;
final int D3 = 3;

MatrixD2 matD2;
MatrixD3 matD3;
```

1. initMyArray 메소드

: main에서 전달된 dimension 값에 따라 MatrixD2의 메소드나 MatrixD3의 메소드를 실행
dim에 해당하는 객체를 생성 (matD2 or matD3)
dim에 해당하는 Matrix 클래스(MatrixD2 or MatrixD3)의 initMatrix() 메소드 실행
-> 배열 생성

```
public void initMyArray (int dim) {
    dimension = dim;

    if (dim == D2) {
        matD2 = new MatrixD2();
        matD2.initMatrix();
    } else if (dim == D3) {
        matD3 = new MatrixD3();
        matD3.initMatrix();
    }
}
```

2. printArray 메소드

dim에 해당하는 Matrix 클래스(MatrixD2 or MatrixD3)의 printMatrix() 메소드 실행
-> 배열 출력

```
public void printArray () {
    if (dimension == D2) {
        matD2.printMatrix();
    } else if (dimension == D3) {
        matD3.printMatrix();
    }
}
```

- MatrixD2 클래스

난수의 범위를 결정하는 RANGE 변수 선언
2차원 배열을 생성하기 위한 DIM 변수 선언
난수값들이 저장될 배열 mat 선언

```
final int RANGE = 10;
final int DIM = 2;

private int[][] mat;
```

1. initMatrix 메소드

2행 2열의 배열 mat 생성
이중 for문을 통해 0~9까지의 값 저장

```
public void initMatrix () {
    mat = new int[DIM][DIM];

    System.out.println("행렬 생성");

    for (int i = 0; i < DIM; i++) {
        for (int j = 0; j < DIM; j++) {
            mat[i][j] = (int) (Math.random() * RANGE);
        }
    }
}
```

2. printMatrix 메소드

[0][0], [0][1], [1][0], [1][1] 순으로 배열 출력
%3d는 세자리 수보다 적을 경우 오른쪽으로 정렬된다.

```
public void printMatrix () {
    // 처음 루프 i = 0 -> 1
    for (int i = 0; i < DIM; i++) {
        // 루프 j = 0
        for (int j = 0; j < DIM; j++) {
            System.out.printf("%3d", mat[i][j]);    //  [0][0], [0][1]
            // x   y                                    [1][0], [1][1]
        }
        System.out.println();
    }
    System.out.println();
}
```

```
행렬 생성
  8  9
  6  8
```

- **MatrixD3 클래스**

난수의 범위를 결정하는 RANGE 변수 선언
3차원 배열을 생성하기 위한 DIM 변수 선언
난수값들이 저장될 배열 mat 선언

```
final int RANGE = 10;
final int DIM = 3;

private int[][][] mat;
```

1. initMatrix 메소드

3행 3열의 배열 mat 생성
삼중 for문을 통해 0~9까지의 값 저장

```
public void initMatrix () {
    mat = new int[DIM][DIM][DIM];

    System.out.println("텐서 생성");

    for (int i = 0; i < DIM; i++) {
        for (int j = 0; j < DIM; j++) {
            for (int k = 0; k < DIM; k++) {
                mat[i][j][k] = (int) (Math.random() * RANGE);
            }
        }
    }
}
```

2. printMatrix 메소드

[0][0][0], [0][0][1], [0][0][2], [1][0][0]... 순으로 배열 출력
%3d는 세자리 수보다 적을 경우 오른쪽으로 정렬된다.

```
public void printMatrix () {
    for (int i = 0; i < DIM; i++) {
        for (int j = 0; j < DIM; j++) {
            for (int k = 0; k < DIM; k++) {
                System.out.printf("%3d", mat[i][j][k]);
            }
            System.out.println();
        }
        System.out.println();
    }
    System.out.println();
}
```

```
텐서 생성
  4  4  4
  3  7  0
  1  1  3

  8  3  1
  4  8  8
  7  7  3

  9  8  3
  3  3  3
  2  4  8
```

3. 고양이 클래스를 만들어주세요.

[main - MyCat]

- main

1. MyCat 형식의 객체인 myCat 생성
2. age와 weight 값을 넘겨주며 MyCat 클래스의 initMyCat 메소드 호출
3. MyCat 클래스의 simulation 메소드 호출 or simulation2 메소드 호출

- MyCat 클래스

1. 변수 선언

```
int age;
float weight;
final String defaultSound = "Meow";

final int FEED = 1;
final int TRAINING = 2;
final int RANGE = TRAINING - FEED + 1;

final int SCHEDULE_NUM = 3;

int scheduleNum;

int simulationCnt;
```

고양이의 나이 : age
고양이의 몸무게 : weight
waringMessage 수행할 때 출력할 메시지 : Meow
하루 일과 종류 : FEED, TRAINING
난수 발생 범위 : RANGE
하루에 3번의 행동을 수행하게 하기 위한 변수 SCHEDULE_NUM = 3
발생된 난수값을 저장할 변수 scheduleNum
simulation2의 종료조건을 판별하기 위한 변수 simulationcnt

2. initMyCat

age와 weight를 받아서 MyCat클래스의 age 변수와 weight에 대입한다.
simulationCnt값을 0으로 초기화한다.

```
public void initMyCat (int age, float weight) {
    this.age = age;
    this.weight = weight;

    simulationCnt = 0;
}
```

3. feed

고양이의 랜덤행동 중 feed가 선택되었을 때 weight 값을 0.1 증가시킨다.

```
private void feed () {
    System.out.println("고오급 사료를 먹습니다!");

    weight += 0.1f;

    System.out.println("운동 하세요!");
}
```

4. training

고양이의 랜덤행동 중 training이 선택되었을 때 weight 값을 0.1 감소시킨다.

```
private void training () {
    weight -= 0.1f;

    System.out.println("몸무게를 조금 뺀것 같습니다!");
}
```

5. warningMessage

현재 고양이의 몸무게에 따른 경고메세지 출력

```
private void warningMessage () {
    System.out.println(defaultSound);

    if (weight > 8) {
        System.out.printf("운동 안하면 고양이 비만 됩니다: %.1f\n", weight);
    } else if (weight < 6) {
        System.out.printf("너무 많이 빼도 위험합니다: %.1f\n", weight);
    } else {
        System.out.printf("현재 고양이는 정상입니다: %.1f\n", weight);
    }
}
```

6. getRandomPattern
고양이의 하루일과를 결정하기 위한 난수 발생 구문
난수발생 범위는 1~2

```
private int getRandomPattern () { return (int) (Math.random() * RANGE + FEED); }
```

7. simulation

- endFlag와 cnt값 초기화

```
boolean endFlag = false;
int cnt = 0;
```

- while 무한반복문: endFlag가 true되었을 때 반복 종료

```
while (true) {
    System.out.println("오늘은 무엇을 할까요 ? (하루 일과는 3가지를 선택합니다)");
```

- endFlag가 true가 되는 조건
1. cnt가 365를 초과 (365일)
2. weight가 12.0 초과
3. weight가 5.1 미만

```
if (cnt > 365) {
    System.out.println("냥이는 가족들과 함께 오래오래 행복하게 살았답니다 ^^");
    endFlag = true;
    break;
} else if (weight > 12.0f) {
    System.out.println("병원 ㅠ");
    endFlag = true;
    break;
} else if (weight < 5.1f) {
    System.out.println("병원 가요 ㅠ");
    endFlag = true;
    break;
}
```

- 하루 일과 3가지를 정하기 위해 for문을 통해 난수를 3번 발생시킨다.
각 루프에서 발생된 난수값 (1~2)에 따라
feed메소드를 실행할지 training메소드를 실행할지 결정 //조건문 switch 사용

```
for (int i = 0; i < SCHEDULE_NUM; i++) {
    scheduleNum = getRandomPattern();

    switch (scheduleNum) {
        case FEED:
            feed();
            break;
        case TRAINING:
            training();
            break;
    }
```

- 하루 일과를 모두 수행한 뒤 경고메세지를 출력하고 cnt의 값을 1 증가시킨다.

```
warningMessage();
cnt++;
```

- 루프 종료 or 다음 루프 실행 결정

```
if (cnt > 365) {
    System.out.println("냥이는 가족들과 함께 오래오래 행복하게 살았답니다 ^^");
    endFlag = true;
    break;
} else if (weight > 12.0f) {
    System.out.println("병원 ㅠ");
    endFlag = true;
    break;
} else if (weight < 5.1f) {
    System.out.println("병원 가요 ㅠ");
    endFlag = true;
    break;
}
```

-

```
if (endFlag) {
    break;
}
```

8. simulation2 : simulation 개선
무한 반복문 안에 반복과 조건문을 모두 넣은 simulation과 달리
반복문의 조건을 checkSimulation()이라는 메소드로 지정해, checkSimulation 메소드에서 false 값을 반환할 때까지 반복이 계속된다.

```
public void simulation2 () {
    boolean endFlag = false;
    int cnt = 0;

    while (checkSimulation()) {
        System.out.println("오늘은 무엇을 할까요 ? (하루 일과는 3가지를 선택합니다)");
```

9. checkSimulation( )
while문의 각 루프가 수행될 때마다 실행되는 메소드
while문 종료판별값인 checkSim 선언 후 false로 초기화
SimulationCnt는 일자를 의미하며 메소드가 실행될 때마다 1증가한다.

```
private boolean checkSimulation () {
    boolean checkSim = false;

    simulationCnt++;
    System.out.printf("%d 일차!", simulationCnt);

    if (simulationCnt > 365) {
        System.out.println("냥이는 가족들과 함께 오래오래 행복하게 살았답니다 ^^");
    } else if (weight > 12.0f) {
        System.out.println("병원 ㄱ");
    } else if (weight < 5.1f) {
        System.out.println("병원 가요 ㄱ");
    } else {
        checkSim = true;
        return checkSim;
    }

    return checkSim;
}
```

일자 365 초과 or 고양이의 몸무게 12.0 초과 or 고양이의 몸무게 5.1 미만일 때
checkSim값이 false로 변하고 이를 반환한다.

그 외의 경우 true를 반환해 while 반복을 계속 수행한다.

5. 학생 클래스를 만들어봅시다.
6. 학생 클래스에 수학, 영어, 국어 점수를 입력 받도록 개조합니다.
7. 학생 클래스의 평균을 계산해봅시다.
9. 7번 문제에서 분산을 구해봅시다.
10. 7번 문제에서 표준 편차를 구해봅시다.

[main - Student - Score]

-main

1. Student 형식의 객체인 student 생성
2. name, age, major 3개의 값을 전달해주며 Student 클래스의 initStudents 메소드 호출
3. toString을 통해 맵핑된 student 객체의 정보 출력

-Student 클래스

1. 변수와 객체 선언
name, age, major
Score 형식의 객체 score 선언

```
String name;
int age;
String major;

Score score;
```

2. initStudents
문자열 값인 name, int형 값인 age, 문자열 값인 major를 받아와
Student 클래스의 name, age, major에 저장한다
Score 형식의 객체인 score를 생성한 뒤, Score 클래스의 initScore 메소드를 호출한다.

```
public void initStudents (String name, int age, String major) {
    this.name = name;
    this.age = age;
    this.major = major;

    score = new Score();
    score.initScore();
}
```

3. toStirng
toString()으로 Student 클래스와 score 클래스의 정보를 맵핑시킨다.

```
@Override
public String toString() {
    return "Student{" +
            "name='" + name + '\'' +
            ", age=" + age +
            ", major='" + major + '\'' +
            ", score=" + score +
            '}';
}
```

- Score 클래스

1. 변수 선언
배열의 최대크기를 10으로 설정, 성적은 난수를 통해 결정되기 때문에
난수의 최댓값 SCORE_MAX와 난수의 최솟값 SCORE_MIN을 선언해준다.
랜덤함수에 사용할 범위값인 range를 선언
성적을 저장할 배열 score 선언
과목명을 저장할 배열 scoreName 선언
사용자의 입력을 받기 위한 스캐너 선언
float형 변수 mean(평균), var(분산), stdDev(표준편차) 선언
몇 과목의 성적을 처리할지 결정하는 변수 cnt 선언

```
final int ARR_MAX = 10;
final int SCORE_MAX = 100;
final int SCORE_MIN = 30;

int range;

int[] score;
String[] scoreName;
Scanner scan;

float mean;
float var;
float stdDev;

int cnt;
```

2. initScore

스캐너를 선언해주고 cnt값을 0으로 초기화해준다.

```
public void initScore () {
    scan = new Scanner(System.in);
    cnt = 0;

    addSubject();
    // 편의상 랜덤값을 활용하도록 함
    inputScore();
    // 평균 계산
    calcMean();
    // 분산 계산
    calcVariance();
    // 표준편차 계산
    calcStdDev();
}
```

- 과목을 추가하는 메소드인 addSubject

백업 배열을 하나 만들어 Java의 Collection 역할을 하게 만든다.

(실제로는 Java의 Collection에 있는 Map이나 Vector를 사용해야 함)

do_while문을 통해 과목을 반복해서 입력받는다.

!) 입력받은 값을 backup 배열에 저장시킬 때 cnt의 값을 1증가시켜 다음 인덱스로 넘어가게 하고, inputScore에서 cnt 변수를 재활용해 난수발생횟수를 증가된 cnt만큼으로 지정할 수 있다.

while문의 조건을 checkInput 메소드로 지정해, 입력을 계속해서 받을지 결정한다.

```
public void addSubject () {
    // 실제로는 Java의 Collection에 있는 Map이나 Vector를 사용해야 하는데
    // 여기서 직접 자료구조를 구현하는것도 에바고
    // 그렇다고 문제를 안 풀수도 없으니 일종의 꼼수로서
    // 백업 배열을 하나 만들어서 Java의 Collection 역할을 하게 만든다.
    String[] backup = new String[ARR_MAX];

    System.out.print("다루고 싶은 과목을 입력하세요: ");

    // 보편적으로 지저분한 코드는 어느 시점에 루프를 멈출지를 결정하는 부분에서 만들어진다.
    // 그러므로 루프를 언제 멈출지 판정하는 부분을 매서드화하면
    // 상대적으로 깔끔한 코드를 얻을 수 있다.
    do {
        backup[cnt++] = scan.nextLine();

        // 현재 다시 입력하세요를 구현하는 부분에서 boolean 변수가 추가됨에 따라
        // 코드의 가독성을 저해하는 부분이 있어
        // 기본적인 로직 작성후 매서드화 시키는 부분이 좋다 판단하여 삭제 처리
    } while (checkInput());
```

과목 입력이 끝나면 for문을 통해 백업 배열에 저장된 값을 scoreName 배열에 대입한다.

```
scoreName = new String[cnt];

for (int i = 0; i < cnt; i++) {
    scoreName[i] = backup[i];
}
```

- checkInput 메소드

문자열 변수인 yOrN에 사용자가 입력한 값을 저장하고, 이를 n과 y와 비교한다.
n을 입력했을 경우 false값을 리턴해 while 조건이 false가 되어 반복이 종료된다.
y를 입력했을 경우 true값을 리턴해 while 조건이 true가 되어 반복이 계속된다.
n과 y가 아닌 다른 값을 입력했을 경우 루프 재동작

```
public boolean checkInput () {
    System.out.print("계속 입력하시겠습니까 ? (y/n): ");

    while (true) {
        String yOrN = scan.nextLine();

        if (yOrN.equals("n")) {
            return false;
        } else if (yOrN.equals("y")) {
            System.out.print("다음 과목을 입력하세요: ");
            return true;
        } else {
            System.out.print("다시 입력하세요! (y/n): ");
        }
    }
}
```

- 점수를 입력하는 메소드인 inputScore

int형 배열인 score 선언 (배열의 크기는 addSubject에서 증가된 cnt 값)

```
public void inputScore () {
    score = new int[cnt];

    setRange();
```

+ setRange 메소드 : SCORE_MIN ~ SCORE_MAX에 해당하는 범위 설정

```
public void setRange () { range = SCORE_MAX - SCORE_MIN + 1; }
```

for문을 통해 cnt 횟수만큼 난수를 발생시켜 score 배열에 저장한다.

```
for (int i = 0; i < cnt; i++) {
    score[i] = (int) (Math.random() * range + SCORE_MIN);
}
```

- 평균을 계산하는 메소드인 calcMean
평균을 의미하는 변수 mean의 값 0으로 초기화
for문을 통해 score 배열의 성적을 모두 더한 뒤, cnt로 나눠 전체 평균을 계산한다.

```
public void calcMean () {
    mean = 0;

    for (int i = 0; i < cnt; i++) {
        mean += score[i];
    }

    mean /= cnt;
}
```

- 분산을 계산하는 메소드인 calcVariance
분산을 의미하는 변수 var의 값 0으로 초기화
for문을 통해 각각의 성적과 평균의 편차를 구한 뒤 제곱하여 이를 var에 더한다.
반복이 종료되면 var를 cnt로 나눠 분산을 계산한다.

```
public void calcVariance () {
    var = 0;

    for (int i = 0; i < cnt; i++) {
        var += Math.pow(score[i] - mean, 2);
    }

    var /= cnt;
}
```

- 표준편차를 계산하는 메소드인 calcStdDev
var의 제곱근을 구해 float으로 형변환시켜 stdDev에 저장한다.

```
public void calcStdDev () { stdDev = (float) Math.sqrt(var); }
```

- toString()으로 Student 클래스와 score 클래스의 정보를 맵핑시킨다.

```
@Override
public String toString() {
    return "Score{" +
            ", range=" + range +
            ", score=" + Arrays.toString(score) + "\n" +
            ", scoreName=" + Arrays.toString(scoreName) +
            ", mean=" + mean +
            ", var=" + var +
            ", stdDev=" + stdDev +
            '}';
}
```

- 출력 결과

```
다루고 싶은 과목을 입력하세요: 국어
계속 입력하시겠습니까 ? (y/n): y
다음 과목을 입력하세요: 영어
계속 입력하시겠습니까 ? (y/n): y
다음 과목을 입력하세요: 수학
계속 입력하시겠습니까 ? (y/n): n
Student{name='나배달', age=20, major='배달', score=Score{, range=71, score=[65, 92, 69]
, scoreName=[국어, 영어, 수학], mean=75.333336, var=141.55556, stdDev=11.897713}}
```