문제해결 및 실습 : Java

과제 1 보고서

제출일	2024.10.20	전공	소프트웨어학과
과목	문제해결및실습:Java	학번	21011746
담당교수	박상일 교수님	이름	양현석

1. 원에 대한 클래스를 생성하고 2개의 원이 겹치는지 확인

a) 문제 분석

기본적인 클래스를 만들고, 여러 가지 Method들을 구현할 수 있는지 파악하는 문제입니다. 또한 원이 서로 겹치는지 여부를 확인하는 방법에 대해서도 고민해야 합니다.

b) 문제 풀이

일단 원이라는 클래스는 중심 좌표, 반지름만 있으면 간단하게 구현할 수 있습니다.

```
class MyCircle {
    public int x, y;
    public int radius;

    MyCircle(int x, int y, int radius) {
        this.x = x;
        this.y = y;
        this.radius = radius;
    }
}
```

그 다음으로 원이 겹치는지 여부에 대한 판단이 필요합니다. 두 원이 겹치는지의 여부는 두 원의 반지름 합계와 중심 사이의 거리를 비교해보면 됩니다. 만일 반지름의 합계가 중심 사이의 거리와 같다면, 두 원은 접하게 되는 것을 이용하는 것입니다. 그렇다면 문제에 해당되는 경우에 대해서 판단하려면, 반지름의 합계와 중심 사이의 거리를 비교하면 되는 것입니다. 접할 때는 겹치는 것이 아니기 때문에 아래와 같이 판단합니다.

```
if (dist > c1.radius + c2.radius) {
    System.out.println("두 원이 서로 겹치지 않는다.");
} else {
    System.out.println("두 원이 서로 겹친다.");
}
```

두 점 사이의 거리를 구하는 방법에 대해서는 생략하겠습니다.

c) 시행 착오

시행 착오는 없었습니다.

2. 연산하기

a) 문제 분석

이 문제는 2개의 피연산자와 사칙연산 기호를 입력 받아 연산을 시행해야 합니다. 이 때 발생할 수 있는 Exception들에 주의하면서 문제를 풀어야겠다고 생각했습니다.

b) 문제 풀이

피연산자와 연산자가 모두 공백 문자를 기준으로 나누어서 들어오기 때문에, next()와 nextDouble()로 입력을 받아주었습니다. 그 후 Switch문으로 Operator에 대한 구분이 들어갑니다. 이때 주의해야 할 점은 '/'일 때 0으로 나누는 경우가 없도록 해야 합니다.

```
if (n2 == 0.0) {
    System.out.println("0 으로 나눌 수 없습니다.");
    System.exit(-1);    //  프로그램 종료
}
```

➔ 바로 프로그램을 종료하도록 함

또한 잘못된 연산기호가 들어왔을 때도 default : 에서 처리하도록 했습니다. 마지막으로 교재에서 출력 케이스가 불분명하게 나와있어, 정수는 정수로 출력되도록 코드를 짜보았습니다.₩

```
if (n1 == (int) n1) System.out.print(Integer.toString((int) n1) + op);
else System.out.print(Double.toString(n1) + op);

if (n2 == (int) n2) System.out.print(Integer.toString((int) n2) + "의 계산 결과는
");
else System.out.print(Double.toString(n2) + "의 계산 결과는 ");

if (res == (int) res) System.out.print((int) res);
```

c) 시행 착오

이 문제를 풀 때, 어떻게 해야 String을 Switch 문에서 쓸 수 있을지에 대해 고민이 많았습니다. 왜냐하면 String을 비교할 때는 .equals()라는 함수를 이용해야 한다고 생각했기 때문입니다. 결국 검색을 통해 Switch 문이 String에 한해서 비교 연산을 지원한다는 것을 알게 되었고, 사용할 수 있었습니다. 더하여 출력할 때, String이나 char, int 간의 연산에 대해 신경을 써주어야 한다는 것을 깨달았습니다.

3. 들어온 숫자로 정수 배열을 랜덤으로 중복 없게 출력하기

a) 문제 분석

중복된 정수가 없도록 해야 하는 문제이기 때문에, 이 알고리즘을 어떻게 효율적으로 구현할까에 대한 고민이 필요합니다.

b) 문제 풀이

일단 먼저 정수를 입력 받고, 이 수에 맞추어 배열을 할당합니다. 그 후 반복문을 돌며 각 배열의 원소들을 random()함수를 이용해서 초기화 합니다. 이때 주의할 점은 1~100 사이의 숫자를 넣어야 하기에 다음과 같이 코드를 짜줍니다.

```
arr[i] = (int) (Math.random() * 100 + 1);
```

➔ random()은 0이상 1미만의 숫자를 반환

이 후 지금까지의 원소에 같은 값이 존재한다면 다시 처음부터 진행합니다. 이는 flag 변수를 이용해서 구현하였고, 배열의 인덱스 참조를 돕는 변수 i의 값을 1을 감소시켜 다시 반복문이 돌 수 있도록 했습니다.

c) 시행 착오

없습니다.

4. 4x4 배열을 만들고, 1~10 사이의 정수를 10개만 랜덤하게 생성하기

a) 문제 분석

이 문제는 얼핏 보면 꽤 복잡한 것처럼 보이지만, 아주 간단하게 풀 수 있습니다.

b) 문제 풀이

문제를 간단하게 풀기 위해, 먼저 배열을 생성하고 모든 배열의 원소들을 채워줍니다. 그 후 반복문을 돌며 2차원 배열의 row, column에 해당하는 인덱스 x, y를 0~3(인덱스 범위) 사이의 숫자로 랜덤 추출합니다. 배열의 원소에 접근하여 0이 아니면 0으로 바꾸어 주고, 0이면 문제 3번과 같은 형태로 반복문의 종료 조건을 부여하는 i를 1 줄여 다시 진행합니다.

```
for (int i = 0; i < CNT; i++) {
    //  0 ~ SIZE-1 사이의 인덱스를 배출
    int x = (int) (Math.random() * SIZE);
    int y = (int) (Math.random() * SIZE);

    //  0을 가진 Element가 6개가 충족되도록 검사
    if (arr[x][y] != 0) {
        arr[x][y] = 0;
    } else i--; //  만일 0인 원소라면, 다시 진행한다.
}
```

c) 시행 착오

없습니다.

이상입니다.