2021. 12. 28 인지연

1. 문제1 -1

```
import java.util.Scanner;

public class Ans1 {
    public static void main(String[] args) {
        /*  아래와 같은 등비 수열이 있다.
            1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, ...
            사용자 입력을 통해 원하는 위치의 값을 뽑아내도록 프로그래밍 해보자!
            (1 ~ 32번째 혹은 31번째 항까지만 올바른 결과가 나올 것임) */

        // 2^8 = 1byte = 256개
        // -128 ~ -1 / 0 ~ 127
        // 0을 포함하기 때문에 2^32승도 맨 끝이 홀수로 되어 있음
        // 그래서 2^n을 표현한다고 할 때 실질적으로 전체 비트 - 1까지만 표현이 가능함
        // 결국 2^31승을 표현하지 못하고 2^31 - 1이 최대값이 되는데
        // 그래서 이 문제에서는 2^30승을 표현하기 위해 31번째까지가 최대가 됨
        final int MAX = 31;
        final int START_IDX = 0;
        final int BASE = 2;
```

필요한 상수들 입력
Base가 2인 이유는 2를 제곱해야하는 값이 등비수열이기 때문,

```
arr[0] 값은 1
arr[1] 값은 2
arr[2] 값은 4
arr[3] 값은 8
arr[4] 값은 16
arr[5] 값은 32
arr[6] 값은 64
```

```
arr[26] 값은 67108864
arr[27] 값은 134217728
arr[28] 값은 268435456
arr[29] 값은 536870912
arr[30] 값은 1073741824
arr[31] 값은 2147483647
```

이 말의 뜻은 출력문을 보면 확인하기 편하다.
Arr[31], 즉 출력값 32번째 값은 홀수가 된다.
2의 제곱이니 무조건 뒷자리는 짝수가 되어야하는데.

이부분이 잘 이해가 안갑니다.
비트 -1 이 무슨 뜻일까요?

2. 문제1 -2

```
System.out.print("찾고자하는 수열의 항을 입력해주세요: ");

Scanner scan = new Scanner(System.in);
int idx = scan.nextInt();

int[] seq = new int[idx];

if (idx > MAX) {
    System.out.println("낵아 표현이 으앙돼 ㅠ 프로그램을 종료합니다.");
} else {
    for (int i = START_IDX; i < idx; i++) {
        // Math.pow()는 n승을 계산함
        // Math.pow(x, y) = x^y로 x의 y승을 계산함
        // 즉 Math.pow(2, i)는 2의 i승을 의미함
        // 2^0 = 1, 2^1 = 2, 2^2 = 4 ...
        seq[i] = (int) Math.pow(BASE, i);
        System.out.printf("seq[%d] = %d\n", i, seq[i]);
    }
}
```

스캐너 입력,
idx값에 사용자의 입력 값int형을 받을예정

배열 선언 / 배열 크기를 사용자 값으로 받는다.

만약 idx가 MAX(31)을 넘어간다면 출력값 입력
(실행이 안된다는 내용)

for 문 실행 0~사용자 입력값까지

순서마다 배열[i]에 값 입력.

제곱하는 방법
Math. Pow(~,!) → ~을 !번 곱한다.
(base(2)를 i번 곱하니까 등비수열 가능)

3. 문제1 –내가 적은 답

```
Scanner scan = new Scanner(System.in);
System.out.print("값을 입력하시오(1~31사이)");
int num = scan.nextInt();
int [] arr = new int [num];
final int START = 1;

arr[0]=1;
System.out.printf("arr[%d] 값은 %d\n",0,arr[0]);

for(int i = START ; i<num;i++){
    arr[i] = (int)(Math.pow(2,i));
    System.out.printf("arr[%d] 값은 %d\n",i,arr[i]);
}
```

스캐너로 사용자 입력값 받기
그 값 그대로 배열의 크기로 입력

굳이 이렇게 안해도 2의 0승은 1이여서 문제없음

4. 문제2

```java
import java.math.BigInteger;
import java.util.Scanner;

public class Ans2 {
    public static void main(String[] args) {
        /*  1번 문제에서 32번째 항이 21억 정도가 나올 것이다.
            BigInteger를 통해서 50번째 항을 구해보자! */

        final int START_IDX = 0;
        final BigInteger BASE = new BigInteger("2");

        System.out.print("찾고자하는 수열의 항을 입력해주세요: ");

        Scanner scan = new Scanner(System.in);
        int idx = scan.nextInt();

        BigInteger[] seq = new BigInteger[idx];
        seq[START_IDX] = new BigInteger("1");

        for (int i = START_IDX + 1; i < idx; i++) {
            seq[i] = seq[i - 1].multiply(BASE);
            System.out.println("seq[" + i + "] = " + seq[i]);
        }
    }
}
```

BigInteger 이용한 코드

BigInteger로 상수 사용한다.
숫자 입력할때는 "2"로 입력한다.
문자열로 입력해야하는구나

BigInteger형으로 배열을 만들었다.
그리고 start_idx, 0번 배열은 1로 지정,
bigInteger에서는 제곱을 구할 수 있는 매소드가 없나보다. 그래서 미리 빼놓았다.

Seq[1] = seq[1-1] * base(2) = 1 * 2
Seq[2] = seq[2-1] * base(2) = 2 * 2
Seq[3] = seq[3-1] * base(2) = 4 * 2
... 으로도 2의 제곱을 구할 수 있다!

5. 문제2 – 내가 적은답

```java
final int END = 50;
BigInteger[] sequence = new BigInteger[END];


final int START = 1;

sequence[0]= new BigInteger( val: "1");
System.out.printf("arr[%d] 값은 %d\n",0,sequence[0]);

  for(int i = START ; i<END;i++){
      sequence[i] = sequence. // 빅인티져의 제곱하는 법을 모르겠습니다.
        System.out.printf("arr[%d] 값은 %d\n",i,sequence[i]);
```

BigInteger의 제곱 값을 못구해서 결국 답을 찾지 못했다.
꼭 굳이 제곱 매소드가 아니여도 구할 수 있는 답이 있다는 걸 체크하기

6. 문제3 -1

```java
public class Ans3 {
    public static void main(String[] args) {
        /*  배열로 로또 문제를 만들어보기!
            실제 로또 확률은 0.00000023으로 1억명중 23명이 당첨됨
            실제값을 사용하기엔 검토 작업이 너무 고통스러우므로 100명 중 5명을 뽑아보도록 하자!
            배열값에 당첨되는 자리를 배치해놓고 사용자가 돌려서 당첨되는지 안되는지를 판정하도록 한다. */

        final int TOTAL = 100;
        final int SELECT = 5;

        boolean[] lottoBox = new boolean[TOTAL];
        int[] selectIdx = new int[SELECT];

        System.out.println("당첨되는 자리를 배치합니다.");

        // 구현 전략
        // 1. 전체 100개 배열을 만듬
        // 2. 당첨 자리 5개 랜덤하게 할당
        // 3. 할당된 자리 중 중복이 존재할 가능성도 있으므로 검사해야함
        //    선택된 인덱스는 0 ~ 99 사이의 랜덤값임
        //    그렇다면 어떻게 이 랜덤 인덱스의 중복 여부를 판정할 것인가 ?
        //    실제 SELECT는 5개이므로
        //    이 SELECT를 활용한 5개 배열에 할당된 랜덤 인덱스를 배치하면 어떨까 ?
        //    그럼 검사를 최악의 경우라고 가정하더라도 최대 4개만 하면 된다.
```

블리언 형 배열도 있네? 블리언 배열은 100명의 크기를 갖는다.
Int 형 배열은 뽑기 5명 당첨의 배열 크기를 갖는다.
두개를 각각 만들어주는 데는 다 이유가 있겠지? 뒤에서 찾아보자

제일 어려운 부분, 확인하러가자

7. 문제3 -2

```
boolean isRealloc = true;
```
(반복적으로 중복검사 하기위해 블린 설정?)

```
int lottoIdx = 0;
int allocCnt = 0;
```
로또 값 받기위해 변수 설정
allooCnt는? 뭐하기위한 값일까?

```
for (int i = 0; i < SELECT; i++) {      // 총 5개 배치
    while (isRealloc) {
        lottoIdx = (int) (Math.random() * TOTAL);
                        0~99 중 랜덤 값 만들기

        isRealloc = false;

        for (int j = 0; j < allocCnt; j++) {
            if (selectIdx[j] == lottoIdx) {
                System.out.println("중복 발생!");
                isRealloc = true;
                break;
            }
        }
    }

    lottoBox[lottoIdx] = true;
    selectIdx[allocCnt++] = lottoIdx;

    // 이대로 가면 무엇을 놓치게 될까 ? 중복을 놓치게됨
    // 그러므로 중복 발생 여부를 체크하는 루틴이 추가로 필요해짐!

    System.out.println("lottoBox[" + lottoIdx + "] = " + lottoBox[lottoIdx]);

    isRealloc = true;
}
```

순환할 경우

For문, select(5) 만큼 반복.

While문 시작,
시작이 true여서 lottoinx 값 랜덤설정이 되고,
False로 바꿔서 일단 랜덤값 더 생성하지 않게 한다.

중복을 확인하기 위해 j의 For안으로 들어가
Index[0] = 랜덤값 => 중복발생없음.

For I = 0
lottoIdx = 4일때,
false
J = 0 ; allocCnt(0) ➔ 검사 없음
for문 나가기
lottobox[4] = ture;
selectIdx[0] = 4;
true
순환

For I = 2
lottoIdx = 34일때,
false
J = 0 ; allocCnt(1)
selectIdx[0] = 34? ➔ 아님
for문 나가기
lottobox[34] = ture;
selectIdx[1] = 34;
true
순환

8. 문제3 -2

```
boolean isRealloc = true;
```
(반복적으로 중복검사 하기위해 블린 설정?)

```
int lottoIdx = 0;
int allocCnt = 0;
```
로또 값 받기위해 변수 설정
allooCnt는? 뭐하기위한 값일까?

```
for (int i = 0; i < SELECT; i++) {      // 총 5개 배치
    while (isRealloc) {
        lottoIdx = (int) (Math.random() * TOTAL);
                        0~99 중 랜덤 값 만들기
        isRealloc = false;

        for (int j = 0; j < allocCnt; j++) {
            if (selectIdx[j] == lottoIdx) {
                System.out.println("중복 발생!");
                isRealloc = true;
                break;
            }
        }
    }

    lottoBox[lottoIdx] = true;
    selectIdx[allocCnt++] = lottoIdx;

    // 이대로 가면 무엇을 놓치게 될까 ? 중복을 놓치게됨
    // 그러므로 중복 발생 여부를 체크하는 루틴이 추가로 필요해짐!

    System.out.println("lottoBox[" + lottoIdx + "] = " + lottoBox[lottoIdx]);

    isRealloc = true;
}
```

중복일경우

For문, select(5) 만큼 반복.

While문 시작,
시작이 true여서 lottoinx 값 랜덤설정이 되고,
False로 바꿔서 일단 랜덤값 더 생성하지 않게 한다.

중복을 확인하기 위해 j의 For안으로 들어가
Index[0] = 랜덤값 => 중복발생한다면 값을 블린을 true로 만들고, break문 for문 나가기
→ while문으로 돌아가서 다시 실행??

For I = 3
  while
  lottoIdx = 34일때,
  false
    J = 0 ; allocCnt(1)
    selectIdx[0] = 34? → 아님
    J = 1 ; allocCnt(2)
    selectIdx[1] = 34? → 맞음
    true // break →for문 나가기
  while
  다시 랜덤 실행 후 false → 반복

9. 문제3 -3

```
boolean isRealloc = true;

int lottoIdx = 0;
int allocCnt = 0;

for (int i = 0; i < SELECT; i++) {      // 총 5개 배치
    while (isRealloc) {
        lottoIdx = (int) (Math.random() * TOTAL);

        isRealloc = false;

        for (int j = 0; j < allocCnt; j++) {
            if (selectIdx[j] == lottoIdx) {
                System.out.println("중복 발생!");
                isRealloc = true;
                break;
            }
        }
    }

    lottoBox[lottoIdx] = true;
    selectIdx[allocCnt++] = lottoIdx;

    // 이대로 가면 무엇을 놓치게 될까 ? 중복을 놓치게됨
    // 그러므로 중복 발생 여부를 체크하는 루틴이 추가로 필요해짐!

    System.out.println("lottoBox[" + lottoIdx + "] = " + lottoBox[lottoIdx]);

    isRealloc = true;
}
```

만약 for I = 3이였고, if문이 true로 break가 되어서 for문이 종료되면,
For I 값은 증감되지 않고 다시 I = 3으로 시작되는건가요?
→첫번째 for문으로 가는게 아니고 while문으로 돌아가서 다시 랜덤뽑는 루트가 맞나요?

Lottebox[랜덤번호] = true → 로또박스배열에 숫자입력
블린형 배열은 어떤걸 하나요?
배열 안에 값을 입력하는것도 아닌것같은데 왜 쓰는건가요?
→선생님 답변 : 당첨이 됐는지 안됐는지 판별하기 위해 쓰인다.
하지만 쓰지 않아도 if문으로도 확인가능하지 않을까요? 아직도 왜 써야하는지 모르겠습니다.

10. 문제3 -다른풀이

```java
public class Ans3_2 {
    // 3번 꼼수 버전
    // 배열을 이용해서 처리할 경우
    // 추가적인 작업이 존재할 때
    // 예) 번호 범위 어디는 배당률 얼마 같은 설정이 가능해짐
    // 현재 케이스는 배열이 아니기 때문에 이와 같은 복합 설정은 어려움
    // 어쨌든 문제 풀이는 맞다고 볼 수 있음
    public static void main(String[] args) {
        final int MAX_NUM = 5;

        int[] selectedLotto = new int[MAX_NUM];

        final int MAX = 100;
        final int MIN = 1;

        int range = MAX - MIN + 1;

        for (int i = 0; i < MAX_NUM; i++) {
            selectedLotto[i] = (int) (Math.random() * range + MIN);
            System.out.printf("당첨 번호: %d\n", selectedLotto[i]);
        }
    }
}
```

1~100중에 랜덤수를 고르고
5번 반복해서 당첨번호를 입력한다.

11. 문제3 –내가 푼 코드

```java
Scanner sc = new Scanner(System.in);
System.out.print("번호를 입력해주세요(1~100)");
int num = sc.nextInt();

final int MAX = 100;
final int MIN = 1;
int range = MAX - MIN +1;
int rand = 0;


final int END = 100;
int [] arr =  new int[END];

for(int i=0;i<END;i++){ // 1~100 순서대로 일단 넣기
    arr[i] = i+1;
}

for(int i = 0;i<5;i++){ // 랜덤문 5개만 뽑기, 근데 뽑으면.. 값이 겹칠수도있잖아?? 교환해야할것같은데
    rand = (int)(Math.random()*range+MIN);
    arr[i] = rand;
    System.out.printf("당첨번호는 %d입니다.\n",rand);
}
if(rand == num){
    System.out.println("당첨입니다!");
}else {
    System.out.println("땡");
}
```

For문으로 100명의 값을 다 줬다.
근데 이렇게 되면 다음 랜덤문으로 뽑기를 할때 그 배열의 값이 같느냐 이거를 고려해봐야하는 상황.
배열순서 값을 교환해야하는데 그럼 내용이 더 복잡해질듯

For문 사용.. 하지만 뭔가 이상하다
For문으로 랜덤값 5번만 반복하는게 제일 나은것같다.

12. 문제4

```java
public class Ans4 {
    public static void main(String[] args) {
        /*  반 학생이 30명이 있다.
            이들은 모두 시험을 치렀고 모든 학생들은 60점 미만이 없다고 한다.
            이 상태에서 학생들의 점수를 임의로 배치하고
            학급의 평균값을 구해보도록 한다. */

        // 1. 고정값 30
        // 2. 최소값 60
        // 3. 난수 생성
        // 4. 배열 필요

        final int STUDENT_NUM = 30;
        final int MAX = 100;
        final int MIN = 60;

        int[] score = new int[STUDENT_NUM];

        int range = MAX - MIN + 1;
        int sum = 0;

        for (int i = 0; i < STUDENT_NUM; i++) {
            score[i] = (int) (Math.random() * range + MIN);
            sum += score[i];

            System.out.printf("score[%d] = %d\n", i, score[i]);
        }

        System.out.println("반 평균 = " + (float)(sum) / STUDENT_NUM);
    }
}
```

```java
        final int STUDENT = 30;
        int []arr = new int[STUDENT];

        final int MAX = 100;
        final int MIN = 60;
        int range = MAX - MIN +1;
        int rand = 0;
        int sum = 0;

        for(int i=0;i<STUDENT;i++){
            arr[i] = (int)(Math.random()*range+MIN);
            sum += arr[i];
            System.out.printf("%d 학생의 점수는 %d\n",(i+1),arr[i]);
        }
        int avg = sum/STUDENT;

        System.out.printf("학생들의 평균은 ? %d",avg);
    }
}
```

13. 문제5

```java
public class Ans5 {
    public static void main(String[] args) {
        /*  4번 문제에서 평균을 구했으므로 표준편차와 분산을 구하도록 한다.
            힌트: Math.sqrt() - 루트 계산 */

        /* 분산, 표준편차 공식: https://math100.tistory.com/11 참고 */

        final int STUDENT_NUM = 30;
        final int MAX = 100;
        final int MIN = 60;

        int[] score = new int[STUDENT_NUM];

        int range = MAX - MIN + 1;
        float sum = 0;
        float average;

        // 평균을 구하기 위해
        // 랜덤 샘플(각각 1명이 가지는 값) 생성
        for (int i = 0; i < STUDENT_NUM; i++) {
            score[i] = (int) (Math.random() * range + MIN);
            sum += score[i];

            System.out.printf("score[%d] = %d\n", i, score[i]);
        }

        average = sum / STUDENT_NUM;
        System.out.println("평균 = " + average);
```

여기까지는 4번과 똑같다.

Sum초기화
(학생들시험점수)

```java
        sum = 0;

        for (int i = 0; i < STUDENT_NUM; i++) {
            sum += Math.pow(score[i] - average, 2);
        }

        System.out.println("분산 = " + (sum / STUDENT_NUM));
        System.out.println("표준편차 = 루트 분산 = " + Math.sqrt(sum / STUDENT_NUM));
    }
}
```

표준편차
1. 평균 구하기(완)
2. 제곱한 값을 구하기, 각 값에서 평균을 빼고 2제곱
3. 위의 값 모두 더하기
4. 자료개수로 나눈다.(학생수 30명 나누기) ->분산 구한것
5. 제곱근 구하기 (Math.sqrt사용하기)

14. 문제5 - 내가쓴답

```
//표준편차
// 1단계 평균 구하기(완) 2단계 제곱한 값을 구하기, 각 값에서 평균을 빼고 2제곱 /
// 3. 위의 값 모두 더하기/4.자료개수로 나눈다./5. 제곱근 구하기
for(int i=0;i<STUDENT;i++){
    sum2 = (int)(Math.pow((arr[i]-avg),2));
}
int avg2 = sum2 / STUDENT;
int std = (int)(Math.sqrt(avg2));
System.out.printf("분산값은%d\n",avg2);
System.out.printf("표준편차는 %d",std);
```

깜박하고 +=를 안했다.

15. 문제6 -1 흐름파악

```java
import java.math.BigInteger;

import static java.math.BigInteger.*;

/*  1. 먼저 BigInteger 형태의 큰 숫자(A)를 배치한다.
    2. 상용로그(log10)을 활용하여 큰 숫자(A)의 길이를 계산한다.
       이 녀석은 어렵게 생각할 필요없이 큰 숫자(A)가 몇 자리가 되는지 판별하는 것임
       예로 1234는 10^3.xxx 로 10의 3승 정도 되는 것을 판정하는 역할
    3. 알아낸 길이값을 기반으로 배열의 개수를 할당한다.
    4. 루프를 돌면서 10^n승을 나눠 몫을 구하고
       이 몫을 다시 10으로 나눠서 나머지를 배열에 배치한다.
    5. n값을 3번에서 알아낸 길이값부터 0까지 내려가면서 진행한다.
    6. 최종적으로 결과를 보면 각 자리숫자를 모두 추출해낸 결과를 얻는다.
    7. 결국 배열 인덱스 0에 들어간 녀석(x)는 10^0 * x를 의미하며
       인덱스 1에 들어간 녀석(y)는 10^1 * y를 의미하게 된다.
       나머지 인덱스들도 마찬가지 역할이 된다. */

public class Ans6 {
    public static void main(String[] args) {
        /*  45678911234라는 숫자를 BigInteger에 배치한다.
            각 자리수에 맞는 숫자를 배열에 배치하도록 한다.
            ex) 1234 = 1 x 10^3 + 2 x 10^2 + 3 x 10^1 + 4 x 10^0
                arr[0] = 4, arr[1] = 3, arr[2] = 2, arr[3] = 1 */
```

16. 문제6 – 2 흐름

```
BigInteger testNum = new BigInteger("45678911234");
```

빅인티져의 인스턴스를 만들었다. TestNum은 "45678911234"

```
final BigInteger BASE = TEN;
```

10을 나눌거라 그의 기본 값을 만들었다

```
BigInteger mantissa = ZERO;
```

0을 만들었네 왜지?

```
BigInteger n = testNum.divide(TEN);
```

Testnum은 빅인티져의 인스턴스 변수명이였음. 그 안에 우리가 구해야할 값을 넣은 상태,
거기에 10을 나눠서 몫이 있는지 검사한다.

```
while (n.compareTo(ZERO) == 1) {
    n = n.divide(TEN);
    mantissa = mantissa.add(ONE)
}
```

Testnum과 0을 비교할때, 같으면 0 , 크면 1이다.
0보다 크다는것은 아직 계산할게 더 남았다는 뜻
N이 0보다 크면 여전히 10^n으로 나눌 수 있으므로 계속 나눔
두번째 10으로 나누기(즉 100나누기), 다음 1000 나누기..
Log10(100)=2이므로

```
// 각각의 자리수를 배치하라는 뜻은 두 가지로 구현이 가능하다.
// 10^n승으로 나눠서 몫을 취하는 방법과
// 10^n승으로 나눈 나머지를 취하는 방법이 있다.
// 구현의 난이도를 생각하면 10^n승으로 나눠서
// 몫을 취하는 방법이 보다 쉬울 것이므로 이 방식을 선택하도록 한다.

// 1. 몫을 구하는 전략 선택
// 2. 근대 언제까지 몫을 구해야 하는가 ?
//    초기 전체 숫자를 10^n승으로 나눠서 몫이 없는지 파악하면 된다.
// 3. 각 반복마다 찾은 몫은 배열에 배치하는 구조로 구성하면 끝

// 추가적으로 필요한 사항이 발생!
// - BigInteger에서 10^n승을 처리할 방법
// - log_10 (100) = 2
// - BigInteger.Log (~~~~) = 7.6
```

총 번째자리수인지 알아야 for문으로 반복하면서 값 도출!?

Biginteger 연산자

.add() = +
.subtact() = -
.multiply() = *
.divide() = /
.mod() = %

17. 문제6 – 3 흐름

```
int length = mantissa.intValue();
```

Testnum의 길이를 구한다. 그 값을 int로 바꾼다.

```
System.out.println("45678911234의 길이: " + (length + 1));
```

Testnum의 길이를 구한다. 그 값을 int로 바꾼다.
Log(~)의 값은 10, +1을 해줘야 길이가 나온다.

```
int[] numArr = new int[length + 1];
```

numArr에 배열 길이를 준다

```
for (int i = length; i >= 0; i--) {
```

높은수부터 –하면서 반복

```
    numArr[i] = testNum.divide(
            new BigInteger(
                    String.valueOf(
                            //Math.pow(BASE, i)
                            BASE.pow(i)
                    )
            )
    ).mod(
            TEN
    ).intValue();   // BigInteger를 int로 변환함
```

18. 문제6 – 4 흐름

```
numArr[i] = testNum.divide(new BigInteger( String.valueOf(BASE.pow(i))))
```

여기까지는 이해가 갑니다.
Ex) 4321의 10^n(높은순)을 해서 배열 값에 4를 넣는건 알겠는데,

```
numArr[i] = testNum.divide(new BigInteger( String.valueOf(BASE.pow(i)))).mod(TEN).intValue();
```

Mod(ten)을 해서 어떻게 나머지 값을 도출해내는지...
10을 나눠서 남은값?
만약에 Ex) 4321의 10^n(높은순)을 나머지한 값이 321이여서, 여기서 다시 나누고.. 이러면 이해가는데
제가 이해한게 맞나요?

19. 문제6 – 다른분들 질문

```
BigInteger mantissa = ZERO;
// 나머지값을 없애려고? 상자를 만듦 <----- 이게 맞나..?
// mantissa = 소수 이니깐 맞는것 같음
```

소수가 아니라 가수라고 한다.
어찌됐든 값을0으로 초기화시켜주고 +1씩 해준다?
Log() = 0
Log() =1이런식으로 더하는거

20. 문제7 – 1

1. 필요한 상수값 열거

```
final int EMP_NUM = 7;
final float INIT_PAY = 3500;
```

2. 소수점 처리를 하려고 하다보니 꼼수를 활용함
0.1 ~ 0.01 사이의 난수를 출력하기 위해
Random의 nextFloat이나 nextDouble을 사용하는 방법도 있지만
Math.random()을 사용할 경우엔 0.0 ~ 0.9999999 까지라는 부분을
처리하는 방식에 대해 살펴보는 부분이 필요함

3. 실제 Math.random() * 10 + 1을 하면 1 ~ 10.99999 까지고
여기에 우리가 강제로 (int) 캐스팅을 해서 1 ~ 10까지 사용해왔음
만약 int 캐스팅을 적용한다면 값은 0.1, 0.2, 0.3, ... 0.9 로 끊겨서 나올 것

```
final int MAX = 10000;
final int MIN = 1000;
final int END_YEAR = 5;
final float BIAS = 1000;
final float PERCENT = 1 / 100.f;
```

1000~10000 / 1000 을 하면 1~10이 된다.
이렇게 표현한 이유,
랜덤을 int로 변환할때, 1~10을 만들어내려면 사실은, 1~10.9999로 표현된다.
Int로 변환했을때는 10으로 나오지만,
실제로는 10이상이 표현되기 때문에 float으로 계산할 때는 10.00으로 계산 할 수 있도록 체크해야한다.

```
int range = MAX - MIN + 1;
float percent;
```

Float으로 사용하는 이유
인상률 자체가 소수점이기 때문

21. 문제7 – 2

```
float[] emp = new float[EMP_NUM];
```

Float형의 직원 배열 만들어준다. 7명의 연봉을 구해야하기 때문

```
for (int i = 0; i < EMP_NUM; i++) {
    emp[i] = INIT_PAY;
}
```

For문으로 우선 직원들의 연봉을 넣어준다.

```
for (int i = 1; i < END_YEAR; i++) {
    for (int j = 0; j < EMP_NUM; j++) {
```

5년동안 반복, 7명의 직원의 연봉을 구한다.

```
    percent = (float) ((int) (Math.random() * range + MIN) / BIAS) * PERCENT;
```

퍼센트 = 1~10/0.01 ➔ 0.01~0.1 의 값
For문 안에 들어있는 이유는 7명의 직원 연봉이 항상 퍼센트마다 다르기 때문

```
    emp[j] += (emp[j] * percent);
```

직원 [0] = 직원[0] + 인상률 ➔ 이런식으로 7명이 5년 반복

```
System.out.printf("연봉[%d] = %f, 증가율 = %f\n", j, emp[j], percent);
```

23. 문제8-1

문제, 순서정하기

2명이 주사위 게임을 한다. (배열 활용)
주사위는 각자 2개씩 굴릴 수 있다.
처음 주사위를 굴렸을때 결과가 짝수라면 한 번 더 돌릴 수 있다.
(2, 4, 6, 8, 10, 12)

한 번 더 돌리는 주사위는 특수 스킬을 가지고 있다.
(특수 스킬 주사위는 1번만 굴린다)
이 특수 스킬들은 1, 3, 4, 6에서 동작한다.

1번의 경우 상대방의 주사위 눈금을 2 떨군다.
3번의 경우 다 같이 -6을 적용한다. (결과는 0 이하로 떨어지지 않는다 - 무승부 노리기)
4번의 경우 그냥 패배 처리한다.
6번의 경우 모든 상대방에게 3을 뺏어서 내거에 3을 더한다.
2번, 5번은 그냥 특수 스킬이 동작하지 않고 단순히 더해진다. */

1. 사용자 수: 2
2. 주사위 수: 2
3. 특수 주사위 수: 1 ???
4. 특수 스킬: 1, 3, 4, 6
5. 각 스킬 이펙트 값들 ???
6. 주사위 생성

1)

```java
public class Ans8 {
    public static void main(String[] args) {
      final int PLAYER_NUM = 2;
      final int DICE_NUM = 2;

      final int SKILL_NUM1 = 1;
      final int SKILL_NUM2 = 3;
      final int SKILL_NUM3 = 4;
      final int SKILL_NUM4 = 6;

      final int DEATH = 4444;

      final int MAX = 6;
      final int MIN = 1;
      int range = MAX - MIN + 1;
```

상수값 입력.

주사위 랜덤 값

24. 문제8-2

2)

```
int dice;
int[] diceSum = new int[PLAYER_NUM];
```

사용자 배열 지정하기

```
for (int i = 0; i < PLAYER_NUM; i++) {
    for (int j = 0; j < DICE_NUM; j++) {
        dice = (int) (Math.random() * range + MIN);
        diceSum[i] += dice;
    }
```

2명의 플레이어가~2번 주사위를 던진다.

주사위 값 랜덤으로 도출 후 기존 값에 더하기!

상대방의 값을 내리기 위해 남을 파악해야한다.

구분하기 위한 방법
i=0 / J=0일때 i=j면 동일인물이라는뜻.
Continue로 값 도출하지 않고
J++해서 처음부터 다시시작

i=0 / J=1일때 i가 같지 않으면 동일인물 아니라서
J의 값을 -2한다!

3)특수스킬 -1

```
for (int i = 0; i < PLAYER_NUM; i++) {

    if (diceSum[i] % 2 == 0) {
        dice = (int) (Math.random() * range + MIN);

        if (dice == SKILL_NUM1) {
            System.out.println("상대방 주사위 눈금을 2 떨군다.");

            for (int j = 0; j < PLAYER_NUM; j++) {
                if (j == i) {
                    continue;   // skip의 의미임
                }

                diceSum[j] -= 2;
            }
        } else if (dice == SKILL_NUM2) {
            System.out.println("모두 함께 자폭 ^^ -6");

            for (int j = 0; j < PLAYER_NUM; j++) {
                diceSum[j] -= 6;
            }
        } else if (dice == SKILL_NUM3) {
            System.out.println("그냥 가세요 ㅠㅜ");

            diceSum[i] = DEATH;
```

25. 문제8-3

3)특수스킬-끝/for문도 우선끝

```
} else if (dice == SKILL_NUM4) {
    System.out.println("모두에게서 3씩 뺏어서 내거에 추가한다.");

    for (int j = 0; j < PLAYER_NUM; j++) {
        if (i == j) {
            continue;
        }

        diceSum[j] -= 3;
        diceSum[i] += 3;
    }
} else {
    diceSum[i] += dice;
}
```

4)음수처리 – 음수가되면 0으로 처리

```
for (int i = 0; i < PLAYER_NUM; i++) {
    if (diceSum[i] < 0) {
        diceSum[i] = 0;
    }

    System.out.printf("dice[%d] = %d\n", i, diceSum[i]);
}
```

5)승패판정

```
boolean checkWinner = true;

for (int i = 0; i < PLAYER_NUM; i++) {
    if (diceSum[i] == DEATH) {
        System.out.printf("플레이어%d가 패배하였습니다!\n", i);
        checkWinner = false;
    }
}

// 승부 판정
if (checkWinner) {
    if (diceSum[0] > diceSum[1]) {
        System.out.println("플레이어 1 승리!");
    } else if (diceSum[0] < diceSum[1]) {
        System.out.println("플레이어 2 승리!");
    } else {
        System.out.println("무승부!");
    }
}
```

Ture로 기본설정,
for 실행될때, 플레이어가 패배하면 다른 결과 볼 필요도없음.
False로 변경한다.

26. 문제10

상수값 입력.

```java
public class Ans10 {
    public static void main(String[] args) {
        final int APPLE_NUM = 5;
        final int MANDARIN_NUM = 3;
        final int ORANGE_NUM = 5;
        final int WATERMELON_NUM = 2;
        final int MELON_NUM = 3;
        final int GRAPE_NUM = 4;

        final int APPLE_IDX = 0;
        final int MANDARIN_IDX = 1;
        final int ORANGE_IDX = 2;
        final int WATERMELON_IDX = 3;
        final int MELON_IDX = 4;
        final int GRAPE_IDX = 5;

        final int TOTAL = 6;

        int[] kindsOfFruit = new int[TOTAL];
        int[] numOfFruit = new int[TOTAL];
```

각각의 값을 배열로 둔다.
과일의 종류 / 숫자

```java
        kindsOfFruit[APPLE_IDX] = 1000;
        kindsOfFruit[MANDARIN_IDX] = 500;
        kindsOfFruit[ORANGE_IDX] = 2000;
        kindsOfFruit[WATERMELON_IDX] = 10000;
        kindsOfFruit[MELON_IDX] = 5000;
        kindsOfFruit[GRAPE_IDX] = 3000;

        // 실제 위에서 사용자가 고른 정보가 배치되고
        // 이 물품을 몇 개 고른지 기록하는 상황임
        numOfFruit[APPLE_IDX] = APPLE_NUM;
        numOfFruit[MANDARIN_IDX] = MANDARIN_NUM;
        numOfFruit[ORANGE_IDX] = ORANGE_NUM;
        numOfFruit[WATERMELON_IDX] = WATERMELON_NUM;
        numOfFruit[MELON_IDX] = MELON_NUM;
        numOfFruit[GRAPE_IDX] = GRAPE_NUM;

        int sum = 0;

        for (int i = 0; i < TOTAL; i++) {
            sum += kindsOfFruit[i] * numOfFruit[i];
        }

        System.out.println("전체 합산가: " + sum);
```

과일의 종류[과일이름] = 가격입력

과일의 숫자[과일이름] = 가격입력
0~5 = 과일_인덱스 = 과일의 숫자

과일종류 * 과일 숫자