2021. 12. 23 인지연

1. Final 사용법 + 사용이유

```java
public class QnA1 {
    public static void main(String[] args) {

        final int MAX1 = 90;
        final int MIN1 = 65;

        final int MAX2 = 97;
        final int MIN2 = 122;

        // 범위: 최대값 - 최소값 + 1 (난수가 나타나는 범위)
        int range = MAX1 - MIN1 + 1;
        int rand = (int) (Math.random() * range + MIN1);
        System.out.println("65 ~ 90 사이의 무작위 난수: " + rand);

        range = MAX2 - MIN2 + 1;
        rand = (int) (Math.random() * range + MIN2);
        System.out.println("97 ~ 122 사이의 무작위 난수: " + rand);
    }
}
```

Random 및 작업 시, 변수 사용하여 값 만들기
그래야 값이 변경될 때 수정이 편하다.
ex) MAX, MIN, range 변수 이름 정하기

사용법

✓ Final 작성하면서 상수값을 지정한다.
✓ 상수 사용할 때의 변수명은 대문자 사용하기.

상수란? 값을 지정하는것 / 변하지 않는 값

사용 이유

✓ 상수로 지정함으로써, 바꾸지 말아야하는 것을 확정한다.
✓ 나 뿐만 아닌, 다른 사람도 함께 쓰는 값일 때 변경불가 표현이 쉬움

Final int 와 int의 차이는?

✓ Final은 상수를 만든다. 변경이 불가능
✓ Int는 변경이 가능

참고해야할 관습

✓ Final → 변수명 전부 대문자
✓ Class → 시작은 대문자
만약 다중 단어라면 단어 이니셜마다 대문자 표기
✓ 변수, 매서드 → 시작은 소문자
만약 다중 단어라면 단어 이니셜마다 대문자 표기

2. 22일 문제은행 QnA -1

Q. 결과 값이 0 만 나오는 이유는?

그 전에, 두 박스의 차이점은?
Random에 곱하기를 하느냐 마느냐의 차이
원래 랜덤 값은 0~0.9999...의 값을 출력한다.
만약 곱셈없이 int형으로 변환한다면 0으로만 출력될 것.

```java
public class QnA2 {
    public static void main(String[] args) {

        int a1 = (int)Math.random();
        int a2 = (int)Math.random();

        System.out.println("a1 = " + a1);
        System.out.println("a2 = " + a2);

        a1 = (int)(Math.random() * 6 + 1);
        a2 = (int)(Math.random() * 12 + 1);

        System.out.println("a1 = " + a1);
        System.out.println("a2 = " + a2);
    }
}
```

A1 = 0~0.9999...의 int형 변환
A2 = 0~0.9999...의 int형 변환

A1 = 1.0~6.9999...의 int형 변환
A2 = 1.0~12.9999..의 int형 변환

결과값

```
a1 = 0
a2 = 0
a1 = 4
a2 = 5
```

랜덤 값이니,
A1은 1~6 / A2는 1~12를 출력한다.

3. 22일 문제은행 QnA -2

```java
public class QnA5 {
    public static void main(String[] args) {
        // 65 ~ 122 까지의 난수를 무작위로 생성하고
        // 65 ~ 90 혹은 97 ~ 122 에 해당하는 숫자만 출력

        final int MAX = 122;
        final int MIN = 65;

        int range = MAX - MIN + 1;
        int rand = (int) (Math.random() * range + MIN);

        boolean condition1 = rand >= 65 && rand <= 90;
        boolean condition2 = rand >= 97 && rand <= 122;

        if (condition1 || condition2) {
            System.out.printf("rand는 영문자 대소문자중 하나임: %c\n", rand);
        }
    }
}
```

final지정

Random수 지정

Boolean → 65<=rand<=90 ➔ 대문자

Boolean → 97<=rand<=122 ➔ 소문자

Or연산자

%c : 아스키코드 출력

논리연산자

- ✓ ||(or연산자) → 두개 조건 중 하나만이라도 일치하면 진행
- ✓ &&(and연산자) → 두 개의 조건중 하나만 일치하면 진행
  OR은 합집합, AND 는 교집합
- ✓ ! (NOT연산) → 참을 거짓으로, 거짓을 참으로 만든다.

출력값

```
rand는 영문자 대소문자중 하나임: P
```

대문자 or 소문자 중 하나의 값만 출력

4. 22일 문제은행
선생님이 의도한 답

```java
public class QnA6 {
    public static void main(String[] args) {
        // 65 ~ 122 까지의 난수를 무작위로 생성하고
        // 65 ~ 90 혹은 97 ~ 122 에 해당하는 숫자만 출력

        final int MAX = 122;
        final int MIN = 65;

        int range = MAX - MIN + 1;
        int rand = (int) (Math.random() * range + MIN);

        // if (rand >= 65 && rand <= 90)
        // 위의 코드와 동의어
        if (rand >= 65) {
            if (rand <= 90) {
                System.out.printf("대문자 범주: %c(%d)\n", rand, rand);
            }
        }

        // if (rand >= 97 && rand <= 122)
        // 위의 코드와 동의어
        if (rand >= 97) {
            if (rand <= 122) {
                System.out.printf("소문자 범주: %c(%d)\n", rand, rand);
            }
        }
    }
}
```

If 문 안에 추가 if 문 입력하면서,
만약 65보다 크고 (if문 안으로)
→ 만약 90보다 작다면 (if문 안으로)
➔ 값 출력

```java
public class WhileTest {
    public static void main(String[] args) {
        boolean isEven = false;
        boolean isOdd = true;
        int num = 1;

        //while (!isEven) {
        while (isOdd) {

            num += (int) (Math.random() * 2 + 1);

            if (num % 2 == 0) {
                //isEven = true;
                isOdd = false;
            }

            System.out.println("num = " + num);
        }
    }
}
```

1. isEven → 기본값 false
While의 소괄호는 true일 때 실행
(!isEven) → false의 반대.. True의미
If문 isEven이 true이다. 그래서 실행문 종료??

2. isOdd → 기본값 true
While의 소괄호는 true일 때 실행
If문 조건이 맞다면 false로 변경,
그래서 실행문 종료

출력값은 1,2번 모두 동일하다.
홀수일때는 반복하면서 num에 값을
더하다가 짝수가 나오면 실행을 종료한다.

```
num = 3
num = 4
```

5. While -1

while을 작성하는 방법
1. while () { } 을 적는다.
2. 소괄호 내부에 조건을 적는다.
3. 중괄호 내부에는 소괄호 조건이 만족되었을 경우 작업할 내용을 작성한다.

num += x
num = num + x와 동의어
현재 아래 케이스에서는 아래와 같은 사항
num = num + (int) (Math.random() * 2 + 1);

단항연산자
+=, -=, *=, /=, %=
- ✓ x += y ===> x = x + y
- ✓ x -= y ===> x = x - y
- ✓ x *= y ===> x = x * y
- ✓ x /= y ===> x = x / y
- ✓ x %= y ===> x = x % y

질문 :
isEven 블리언 기본 값이 false라서
if문의 값이 true로 나왔을때,
실행문을 종료하게 되는건가요?

6. While -2

While문의 기본 형태1

```java
public class WhileTest2 {
    public static void main(String[] args) {
        int i = 0;
        while (i++ < 10) {
            System.out.println("i = " + i);
        }
    }
}
```

i=0 -> i++ -> i =1
i=1 -> i++ -> i =2
i=2 -> i++ -> i =3
.
.
i=9 -> i++ -> i =10

질문:
(i++<10) 구문을 나왔기 때문에
i+1을 한걸로 보면 될까요?
While()에서는 I = 0
While() 이후에는 I = 1

While문의 기본 형태2

```java
public class WhileTest3 {
    public static void main(String[] args) {
        int i = 0;

        while (i < 10) {
            i++;
            System.out.println("i = " + i);
        }
    }
}
```

위에나 이거나, 어찌됐든
출력 전 ++을 하여 값이 +1을 하게된다.

출력값

```
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
```

7. While -3

```java
public class WhileTest4 {
    public static void main(String[] args) {
        int i = 0;

        // 1 ~ 30까지의 숫자중 5의 배수를 모두 출력하세요
        // 와 같은 역할을 수행하게 됨
        while (i < 30) {
            i++;

            if (i % 5 == 0) {
                System.out.printf("i는 5의 배수 = %d\n", i);
            }
        }
    }
}
```

While로 0~30까지의 값을 반복하여
If문에서 원하는 값이 나올 때,
그 값을 출력하게 해주는 배수 구하는 문제,
참고 하기

출력값

```
i는 5의 배수 = 5
i는 5의 배수 = 10
i는 5의 배수 = 15
i는 5의 배수 = 20
i는 5의 배수 = 25
i는 5의 배수 = 30
```

8. 전위 연산자 vs 후위 연산자

```java
public class PrePostOperationTest {
    public static void main(String[] args) {

        int i = 0;


        System.out.println("전위 연산");
        System.out.println("i = " + (++i));
        System.out.println("i = " + i);

        i = 0;

        System.out.println("초기화 확인 i = " + i);

        System.out.println("후위 연산");
        System.out.println("i = " + (i++));
        System.out.println("i = " + i);
    }
}
```

전위 연산자의 경우
해당 줄이 실행되기 이전에 덧셈이 완료됨

후위 연산자의 경우
해당 줄이 실행된 이후에 덧셈이 완료됨

출력값

```
전위 연산
i = 1
i = 1
초기화 확인 i = 0
후위 연산
i = 0
i = 1
```

++I → 0+1
i++ → 0 (이후 값에 +1)

9. For문 -1

```java
public class ForTest {
    public static void main(String[] args) {

        for (int i = 1; i <= 10; i++) {
            System.out.println("i = " + i);
        }


    }
}
```

출력값

```
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
```

I는 1이고, 1부터 증가해서 10까지
해당 구문을 반복해서 실행해라
(총 10번 실행)

for문을 작성하는 방법
for (초기화; 조건; 증감) 형식으로 구성됨

1. for () { }를 적는다.
2. 소괄호 내부에는 작성하는 기준이 조금 많다.
   2-1. 초기화 루틴
   2-2. 조건
   2-3. 증감
3. 소괄호 내의 조건이 만족된다면
동작시킬 코드를 중괄호 내부에 작성한다.

[1] 초기화 부분은 for문 진입시 최초 한 번만 실행된다.
[2] 조건은 매 반복마다 검사한다.
[3] 증감은 중괄호 내부의 내용이 끝나면 진행한다.

동작 시나리오 일부 예
1. for (int i = 1; i <= 10; i++)를 만났으므로 i는 1이 됨
2. i <= 10 조건을 비교하고 결과는 참이므로 중괄호 진행
3. i값인 1을 출력
4. 중괄호 파트가 끝났으므로 증감이 진행되어 i는 2가 됨
5. 조건은 매 반복마다 검사하라고 하였으므로 i <= 10을 비교(역시 참)
6. 중괄호 진행하여 i값인 2 출력
7. 4번 ~ 6번을 끝날때까지 반복

* 초기화, 증감, 조건 모두 없어도 된다.

10. For문 -2
For문 이용한 합산

```java
public class ForSumTest {
    public static void main(String[] args) {

        int sum = 0;

        for (int i = 1; i < 101; i++) {
            sum += i;
        }
        System.out.println("1 ~ 100까지의 합산 = 5050 맞니 ? " + sum);
    }
}
```

sum이 for 외부에 필요한
이유 1.
sum은 지역 변수로 main 내부에서는 어디서든 살아있음
만약 sum을 for문 내부에 적으면
for문이 끝나는 순간 for문과 함께 소멸함 (정보 손실)

이유 2.
sum = 0 실제로 메모리에는 중구난방으로 데이터들이 들어있음
그렇기 때문에 int sum 이라고 해놓으면
실제로 sum에는 239042739 무작위로 이런 값이 들어있다.

For문을 통해 1~100의 수가 반복되면서 그 값을 sum해주게 된다.
ex)
0 + 1 = 1 → sum값 1로 변환 (i는 1부터 100까지 순차적으로 증감한다.)
1 + 2 = 3 → sum값 3로 변환 (sum은 반복되면서 계속 값이 변한다.)
3 + 3 = 6 …

질문 : 지역변수의 의미를 알고 싶습니다.

```
1 ~ 100까지의 합산 = 5050 맞니 ? 5050
```

11. For문 -3
For문과 if 문 함께 쓰기

```java
public class ForIfTest {
    public static void main(String[] args) {
        // 1 ~ 100까지 숫자중 짝수만 출력하세요.
        for (int i = 1; i < 101; i++) {
            if (i % 2 == 0) {
                System.out.println("짝수 = " + i);
            }
        }
    }
}
```

For문 1~100까지
If문의 값 (2 나눈 값이 0이냐)이 true면
출력하면서 순환한다.

출력값 : 1~100까지의 2의 배수가 출력됨

```
짝수 = 2
짝수 = 4
짝수 = 6
짝수 = 8
짝수 = 10
짝수 = 12
짝수 = 14
짝수 = 16
짝수 = 18
짝수 = 20
짝수 = 22
짝수 = 24
```

12. 무한루프

For문 이용한 무한루프

```java
public class ForInfiniteLoopTest {
    public static void main(String[] args) {

        // 초기화, 증감, 조건 모두 없어도 된다.
        // 조건이 없으면 ? 무조건 <<< 결국 무한 루프(반복)

        // 문법 구조가
        // (초기화; 조건; 증감)
        for (;;) {
            System.out.println("안녕!");
        }
    }
}
```

While문 이용한 무한루프

```java
public class WhileInfiniteLoopTest {
    public static void main(String[] args) {
        while (true) {
            System.out.println("Hello!");
        }
    }
}
```

For문이 무한루프 쓸 때는 좀 더 간단함!

for와 while의 차이?
-> while 은 조건 내에서 반복(원하면 너가 증감을 쓰던지)
for는 조건 내에 초기화, 조건, 증감이 필수
만약 증감할 일이 없다고 한다면 while 쓰는 것이 편리하다.

무한루프 하는 예 :
줌(무한루프를 돌면서 화면에 정보를 네트워크에 전송
-> 나의 화면에 전달받는 작업을 반복하고있다.)