1. 변수

변수를 만드는 방법

1. 데이터 타입을 적는다. (int, float, double, long)

int : 정수형 (숫자 - 음수, 양수)

long: 비슷함

float: 소수점

double: 더 정밀한 소수점 (실제 작업하면서 double 사용할 일은 드물다.)

2. 변수 이름을 작성한다. (이름은 만들고 싶은대로 만들 수 있다.)

3. 필요하다면 초기화를 진행한다.

변수를 만들 때 가장 중요한 것은 변수의 이름이다.

변수의 이름은 팀원들이 이 변수가 무엇을 의미하는 지 명확하게 알수 있도록 명시성있게 만들어야한다.

final 변수

final 을 사용하는 이유

: 상수로 고정시킬 수 있다는 이점이 있다.
그렇다면 상수로 고정키는 것의 이점은 무엇인가?

1. 유지보수
아래 있는 [비교 대상]에서 3.3f 란 값에 변동이 있다면 모든 코드에서 3.3f 를 찾아서 수정해야한다.

반면 TAX 상수에 숫자를 기입하고 이 상수를 사용한다면 TAX 값만 변경하면 되므로 유지보수가 편리하다.

```
final float FULL_PERCENT = 100;
final float TAX = 3.3f;
int income = 1000000;

System.out.println("프리랜서 세전 수입: " + income +
        ", 세후:" + income * (FULL_PERCENT - TAX) / FULL_PERCENT);
/*
[ 비교 대상 ]
System.out.println("프리랜서 세전 수입: " + income +
        ", 세후:" + income * (FULL_PERCENT - 3.3f) / FULL_PERCENT);
```

2. 사고방지
불변 객체 (Immutable Object) (인스턴스 = 객체)
클래스를 인스턴스화 하여 객체를 만들었고 이것이 불변이라면 무엇이 좋을까?

위 코드는 TAX 가 final 이기 때문에

TAX = TAX + 4;

위 같은 코드를 통한 새로운 값의 대입, 덧셈, 뺄셈 등이 불가능하다.
입력되는 값을 변경하지 못하게 막음으로서 원래 동작해야 하는 동작의 무결성을 보호한다.

2. For 문

for문을 만드는 방법

1. for ( ) { }

2. 소괄호 내부의 구성은 아래와 같다.
(초기화; 조건; 증감)

여기서 초기화란 for 문을 최초로 만나는 순간에만 동작한다. 따라서 생략해도 된다.
- 조건을 만족하는 동안 for 문이 반복된다. (while, if, switch 의 조건식과 동일하다.)

- 조건이 없으면 무조건이므로 무한 반복이다. // for (; ;) { }
- 증감도 생략 가능하다.

3. 중괄호 내부는 for 문을 반복하며 작업할 내용이다.

4. for 문은 조건 파트가 참인 동안은 계속 반복된다. (중요)

```
for (int idx = START; idx < END; idx++) {
    System.out.println("idx = " + idx);
}
```

fot문 루프 만드는 방법

1. 외부에 0 으로 초기화 변수를 선언한다.

2. for (; ; 변수++) { }

```
int sum = 0;
int count = 0;

for (int idx = 3; idx <= 10; idx++) {
    sum = sum + idx;
    System.out.println("count =" + (++count) + ", sum = " + sum);
}

System.out.println("3 ~ 10 까지의 합: " + sum);
```

count 는 1, sum 은 3 부터 count 는 8, sum 은 52 까지 출력후에

마지막 줄인 " 3 ~ 10 까지의...." 문장을 출력한다.

for문의 continue

```
for (int i = START; i <= END; i++) {
        if (1 % 2 == 0) { continue; }
    System.out.println("i = " + i);
   }
```

continue 는 skip 과 동일하다.
continue 에 걸리면 뒷문장을 출력하지 않고 다시 첫줄의 i++로 넘어간다.

(+ whlie 은 단순반복에, for 는 조건반복에 주로 쓰인다.)

3. If 문

if 문을 만드는 방법

1. if ( ) { }
2. 소괄호 내부는 조건식을 작성한다.
3. 중괄호 내부는 조건이 참일 경우 동작할 코드를 작성한다.

4.if ( ) { } else if { }

  else if 를 많이 넣기 보단 그냥 if 를 여러번 쓰는 것이 가독성 좋다.

```
final int PERMIT_AGE = 18;
final int inputAge = 19;

if (PERMIT_AGE < inputAge) {
    System.out.println("입장 가능하십니다!");
} else  {
    System.out.println("입장 불가능하십니다!");
}
```

4. Scanner 사용자 입력

Scanner 사용자 입력

사용자 입력이란 구체적으로 키보드 입력을 의미한다.

Scanner 는 키보드 입력 처리를 위해 사용하는 객체이다.

아래 코드를 통해 현재 콘솔 상황에서 사용자 입력을 받을 수 있다.

```
Scanner scan = new Scanner(System.in);
System.out.println("숫자를 입력하세요: ");
int inputNumber = scan.nextInt();
```

입력 받은 값을 사용 시에는

키보드 입력으로 int 타입을 수신한다면 next**Int**()를 사용한다.
만약 double 타입을 원한다면 next**Double**()
float 타입을 원하면 next**Float**() 형태로 사용한다.

5. Switch 문

Switch문

switch 문을 작성하는 방법

1. switch ( ) {

    case *조건1*:

        조건만족 시 실행할 것

        break;

```
    case 조건2:
        조건만족 시 실행할 것
        break;

    default:
        조건만족 시 실행할 것
        break;
}
```

2. 소괄호 내부에 switch case 에서 사용할 조건을 적는다.

->현재 케이스에서

case 0:의 의미는 '입력된 숫자가 0 이면'이란 뜻이 rh,

case 1: 의 의미는 '입력된 숫자가 1 이면'이란 뜻이다.

3.중괄호 내부에는 case 조건들을 적고, 각 조건에 대응하는 코드를 작성한다.

```
switch (number1) {
    case 0:
        System.out.println("종료!");
        isLoop = false;
        break;

    case 1:
        System.out.println("입금!");
        break;

    case 2:
        System.out.println("출금!");
        break;
```

```
    case 3:
        System.out.println("조회!");
        break;

    default:
        System.out.println("그런 명령은 존재하지 않습니다!");
        break;
```

6. While 문

While문

While 문 작성 방법

1. While ( ) { }

2. 소괄호 내부에 조건식을 작성한다.
조건이 만족되면(True) 루프를 돌고 만족되지 않으면(False) 루프를 빠져나온다.

3. 중괄호 내부에 조건이 만족되는 동안 반복시킬 코드를 작성한다.

```
int idx = 0;
final char ch = 'A';

while (idx < 10) {
    System.out.println("idx: " + idx + "안녕: " + (char)(ch + idx));
    idx++;
}
```

' *idx: 0 안녕: A* '부터

' *idx: 9 안녕: J* '까지

출력 후 루프를 빠져나온다.

8.주사위 문제

주사위 문제

```
public class DiceTwo {

    public  static void main(String[] args) {
        final int MAX = 6; //주사위 최고 숫자
        final int MIN = 1; //주사위 최소 숫자

        final int RandomDice1 = (int)(Math.random() * 6) + 1; //주사의 1
값은 랜덤 1~6
        final int RandomDice2 = (int)(Math.random() * 6) + 1; //주사위 2
값은 랜덤 1~6

        System.out.println("주사위 1의 값은 " + RandomDice1 + "입니다.");
        System.out.println("주사위 2의 값은 " + RandomDice2 + "입니다.");

        int DiceSum = RandomDice1 + RandomDice2; //두 주사위의 합
        int number1 = DiceSum % 4; //4로 나누고난 나머지 값

        if(number1 == 0) { //나머지가 0이면 출력
            System.out.println("주사위의 합은 " + DiceSum + "이며
승리하셨습니다.");
        } else { //나머지가 0이 아니면 출력
            System.out.println("주사위의 합은 " + DiceSum + "이며
패배하셨습니다.");
        }
    }
}
```

+

동작은 잘 되지만

첫줄인 public class DiceTwo { 부분에서 오류가 있는 것 같은데 이유를 모르겠다.

->그냥 클릭 잘못해서 빨간 동그라미 생긴것ㅋㅋ

while 이나 for, switch 로도 시도해볼까?

10. format 출력 printf

format을 출력하는 printf

printf 사용방법

정수형(int) → %d

문자형(String) → %s (그러나 자바에서 문자형의 경우 그냥 +를 사용하면 됨)
실수형(float, double) → %f

```
int a = 100;
String b = "백";

System.out.printf ("%d 은 %s 이다.", a, b);
System.out.println();

System.out.println( a + "은 " + b + "이다." );
```

둘은 같은 문장을 출력한다.

11. 생성자와 Method 메서드

생성자

생성자 만드는 방법

Public 클래스이름 (자료형 입력받을 값) {

구동시키고자 하는 작업

}

1. 생성자는 class 의 이름과 같다. 리턴 타입이 없다.
2. 만약에 외부에서 값을 입력 받을 것이라면 소괄호( ) 에 입력받을 형태를 작성한다.
3. 실제 클래스가 new 를 통해 객체화 될 때 구동시키고 할 작업을 중괄호{ } 내부에 배치한다.

Method 메서드

클래스 내부에서 기능을 수행하는 집합

메서드 만드는 방법

public 리턴타입 매서드이름() {

구동시키고자 하는 작업

return 리턴할값 ;

}

1. public 옆에 리턴타입과 매서드 이름을 작성한다.

2. 중괄호{ } 내에서는 실제 매서드 구동할 작업을 작성한다.

+리턴

: 함수의 결과 값을 반환한다는 뜻의 예약어

- 리턴타입

반환되는 값 기준

↓리턴타입
3 -> [ ] -> 9 int
버튼을 누름 -> [ ] -> true boolean
1 -> [ ] -> "예금" String
회원 정보 -> [ ] void (리턴하지 않음) (생성자의 경우 리턴타입 자체가 없는 것.)
> [ ] -> 20 int

- 입력타입

입력되는 값 기준

↓입력타입
3 -> [ ] -> 9 int
버튼을 누름 -> [ ] -> true Button class
1 -> [ ] -> "예금" int
-> [ ] -> 20 void (소괄호 비워두면됨)

참/거짓 -> [ ] boolean

12. static

static

static 은 언제나 메모리에 상주한다. Stack 도 Heap 도 아니다.
그러므로 별도로 new 를 할 필요 없이 사용할 수 있다.
대표적으로 main, Math.ranmdom 이 있다.

```java
public class CustomRandom {
    final private static int MIN = 0;
    public static int generateNumber (int min, int max) {
        return (int) (Math.random() * (max - min + 1)) + min;
    }
    public static int generateNumber (int max) {
        return generateNumber(MIN, max);
    }
}
```