

아는 내용만 정리하였습니다. (김진선)

1. 변수, 상수
2. 조건문 if, if else, switch
3. 반복문 while, for, foreach
4. 대입연산자, 나머지 연산자 , continue
5. Random
6. 클래스, Private, setter, getter

1. 변수, 상수

■ 변수란

- 변하는 수로, 어떤 데이터를 저장하는 공간
- 변수 이름을 선언할 때 그 이름이 한눈에 무엇을 의미하는지 쉽게 해석 될수 있어야 한다.

데이터 타입 종류

Int : 정수형

Long: int 보다 더 큰 내용의 데이터

Float: 소수점

Double: float 보다 더 정밀한 소수점 (실제 사용할 일은 드물다.)

오른쪽 그림에서
노란색 표시된 부분을 보면 (변수)
대입연산자 뒤에 오는 값이 어떤 의미인지
쉽게 알수 있다.

```
Int appleCount = 3;
```

```
Int grapeCount = 5;
```

```
Int totalCount = appleCount + grapeCount;
```

1. 변수, 상수

■ 상수란

- 변하지 않는 수로, 해당 데이터가 변하지 않도록 final 을 붙인다.

상수가 왜 필요할까 ?

• 일괄처리 가능

특정 변수값을 여러번 변경해야 할때,
상수화 시키게 될 경우
제일 윗단에서 **한번만 변경**하면 된다
(오른쪽 그림의 노란색 빗금 참고)

• 불변 객체


바뀌면 큰일나는 변수들의 값을 고정시켜
원래 동작해야 하는 동작의 **무결성을**
보장하게 된다.

```
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. 조건문 if, if else, switch

▪ If 문

- 조건식이 참일 경우, 중괄호 내용을 수행하고 거짓일 경우, else 문을 수행 하거나 if문을 빠져나간다.

If 문 작성방법

- if를 작성하고 소괄호() (조건식) 를 작성하고 중괄호{} (동작할 코드)를 작성한다.

if else 문 보다는 if if 문을 사용하는게 좋은 이유

- If else로 작성하게 되면 처음 조건, 그다음 조건, 그다음 조건을 다 생각해야 하기 때문에 depth(깊이)가 깊어질수록 코드를 파악하기 위한 혼동이 가중된다는 문제가 있다.

그러므로 if if 문을 쓰는것이 더 좋다.

```
public static void main(String[] args) {  
    final int PERMIT_AGE = 18;  
    final int inputAge = 15;  
    final int PERMIT_KIDS = 13;  
  
    //if문 으로 표현  
    if (PERMIT_AGE < inputAge) {  
        System.out.println("성인용입니다!");  
    }  
  
    if (PERMIT_KIDS < inputAge) {  
        System.out.println("아동용입니다!");  
    }  
  
    //ifelse문 으로 표현  
    if (PERMIT_AGE < inputAge) {  
        System.out.println("성인용입니다!");  
    } else if (PERMIT_KIDS < inputAge) {  
        System.out.println("아동용입니다!");  
    }  
}
```

2. 조건문 if, if else, switch

▪ Switch 문

- if문과 비슷하지만

조건식의 결과가 정수 또는 문자열 값이고

그 값에 따라

수행문이 각각 다른 경우에 사용한다.

```
int inputNumber = scan.nextInt();

switch (inputNumber) {

    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;

}
```

3. 반복문 while, for, foreach

- **while 문: 단순반복문**

소괄호 안에 있는 조건식이 참일 동안 중괄호 안에 있는 코드가 **반복수행** 되고

그렇지 않으면 수행을 중단한 후 while문을 빠져나온다.

무한반복 수행을 하고 싶을때는 조건식을 true로 설정해주면 된다.

```
public static void main(String[] args) {  
    int idx = 0;  
    final char ch = 'A';  
  
    while (idx < 10) {  
        System.out.println("idx: " + idx + ", 안녕: " + (char)(ch + idx));  
        idx++;  
    }  
}
```

3. 반복문 while, for, foreach

- **for 문: 조건반복문**

소괄호 안에 있는 조건식 (초기화; 조건; 증감) 에서

초기화 값이 조건에 맞으면 중괄호 안의 코드를 수행한다.

후에 증감을 하고 조건이 만족할 때까지

반복한 후에 조건을 벗어나면

for 문을 빠져나온다.

오른쪽 예시를 보면

위의 for문과 아래의 for문이
같은 뜻을 의미하고 있다.

보통은 위의 문장으로 사용한다.

```
public static void main(String[] args) {  
    final int START = 3;  
    final int END = 10;  
    int index = START;  
  
    for (int index = START; index < END; index++) {  
        System.out.println("index = " + index);  
    }  
  
    //위와 같은 뜻.  
    for (; index < END;) {  
        System.out.println("index = " + index++);  
    }  
}
```

3. 반복문 while, for, foreach

▪ foreach 문: for의 변형 버전

foreach 사용법

1. 배열의 데이터 타입을 작성한다.
2. 배열의 원소를 표현할 이름을 지정한다.
3. 콜론 하나 찍는다. (:)
4. 정보를 하나씩 꺼내 올 배열을 적어준다.

오른쪽 예시를 보면

위의 for문과

아래의 foreach문 출력 결과가 같다.

```
final int[] numberArray = {1,2,3,4,5};  
  
for (int i = START; i < numberArray.length; i++) {  
    System.out.println("배열 출력: " + numberArray[i]);  
}
```

```
final int[] numberArray = {1,2,3,4,5};  
  
for (int num: numberArray) {  
    System.out.println("배열 원소 출력: " + num);  
}
```


4. 대입연산자, 나머지 연산자, continue

- **대입연산자: =**

~와 같다라는 뜻이 아니고
오른쪽 정보를 왼쪽에 대입한다라는 뜻이다.
오른쪽의 sum에
sum+idx 값을 증감하면서 누산한다.

```
System.out.println("3부터 10까지 더하기");  
int sum = 0;  
int count = 0;  
  
for (int idx = START; idx <= END; idx++) {  
    sum += idx;  
    System.out.println("count = " + (++count) + ", sum = " + sum);  
}  
System.out.println("3 ~ 10까지의 합: " + sum);
```

- **나머지 연산자: %**

오른쪽과 같이
i 값을 2로 나눈 나머지가 0이냐를 묻는 것이고
그 뜻은 짝수인지를 묻는 뜻이다.

- **Continue;**

수행문 중간에 continue; 가 나왔다면
이후 수행을 생략하고 다시 위로 올라가
증감후 조건이 맞으면 처음부터 다시 수행한다.

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

5. Random

랜덤 숫자를 구하는 식은 아래와 같다.

```
(int)(Math.random() * (최대값 - 최소값 + 1)) + 최소값;
```

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

    final int START = 0;
    final int END = 100;

    // 1~100 랜덤 구하기
    final int randomNumber = (int)(Math.random() * 100) + 1;
    System.out.println("randomNumber = " + randomNumber);

    // 5~15 랜덤 구하기
    final int MAX = 15;
    final int MIN = 5;

    int randomValue = 0;
    for (int i = START; i < END; i++) {
        randomValue = (int)(Math.random() * (MAX - MIN + 1)) + MIN;
        System.out.println("i = " + i + ", randomValue = " + randomValue);
    }
}
```

6. 클래스, Private, setter, getter

클래스를 만드는 이유는

메인 메서드 안에서 모든 기능들을 작성하게 되면 가독성이 떨어지게 되므로

각각의 기능에 따라 클래스를 분리하여 메인 메서드와의 역할 분담을 시켜 코드 가독성을 높이고 각각의 기능별로 관리할 수 있도록 제어하기 위함이다.

Private

- Public 과 반대 성격으로 객체 정보를 보호한다. (정보은닉)
- 디폴트 생성자를 무조건 만든다.
- Private 멤버변수를 접근하기 위해 setter, getter 메소드를 사용한다.
(보통 getter 만 사용한다. Setter를 사용하다가 실수할 가능성이 높기 때문이다.)
- 객체가 가지고 있는 값을 확인하기 위해서 toString 메소드를 정의한다.