

# Object – Oriented Programming

---

## LAB #2-2. FLOW OF CONTROL

# if - else

- 조건문의 값에 따라 프로그램의 실행순서를 결정하는 제어문
- if문의 조건문의 값이 true일 시, 종속문장을 실행
- else if문은 여러 개를 작성 가능

```
if (netIncome <= 15000)
    tax = 0;
else if ((netIncome > 15000) && (netIncome <= 30000))
    tax = (0.05*(netIncome - 15000));
else {
    fivePercentTax = 0.05*15000;
    tenPercentTax = 0.10*(netIncome - 30000);
    tax = (fivePercentTax + tenPercentTax);
}
```

조건문

종속문장

# switch - case

---

- 다중 if문의 표현식과 비슷한 방법의 제어문
- case문은 순서대로 실행되며 1개의 case문을 실행한 후 switch문을 빠져나오려면 break를 실행 문장 뒤에 작성
- case문에 해당하는 값이 없을 시, default문의 문장이 실행
- Java 7 이후부터는 switch 문장에 String 타입 지원(단, null에 대한 선행 처리 필수)

# switch – case example

```
private static int convertLastNameToCode(String lastName) {  
    int code = 0;  
    lastName = lastName.toLowerCase();  
  
    switch(lastName) {  
        case "shin" :  
            code = 1;  
            break;  
        case "kim" :  
            code = 2;  
            break;  
        case "lee" :  
            code = 3;  
            break;  
        case "park" :  
            code = 4;  
            break;  
        default :  
            code = 0;  
    }  
  
    return code;  
}
```

```
private static String makeOrdinalNumber(int num) {  
    switch(num % 10) {  
        case 1 :  
            return num + "st";  
  
        case 2 :  
            return num + "nd";  
  
        case 3 :  
            return num + "rd";  
    }  
  
    return num + "th";  
}
```

# for

---

- 초기문, 조건문, 증감문으로 이루어진 기본적인 반복문
- for문의 초기문, 조건문, 증감문은 일부 혹은 전부 생략 가능

```
public static void main (String[] args)
{
    String lastNameList[] = {"Shin", "Kim", "kim", "Lee", "PARK", "yoon", "JEONG"};
    int lastNameCode[] = new int[lastNameList.length];
    for(int i=0;i<lastNameList.length;++i) {
        lastNameCode[i] = convertLastNameToCode(lastNameList[i]);
        System.out.println(makeOrdinalNumber(i + 1) + " code : " + lastNameCode[i]);
    }
}
```

```
int[] nums = {1, 2, 3, 4, 5};
for(int num : nums) {
    System.out.println(num);
}
```

# while

---

- 조건문의 값이 true일 동안 실행문을 반복
- do-while문 : 조건문의 값에 상관 없이 처음에는 반드시 실행문을 수행  
그 이후에는 while문과 동일하게 동작

```
Scanner scan = new Scanner(System.in);  
ArrayList<String> lastNameList = new ArrayList<>();
```

```
while(scan.hasNext()) {  
    ...  
    lastNameList.add(scan.next());  
}
```

```
int lastNameCode[] = new int[lastNameList.size()];  
int i = 0;  
do {  
    lastNameCode[i] = convertLastNameToCode(lastNameList.get(i));  
    System.out.println(makeOrdinalNumber(i + 1) + " code : " + lastNameCode[i]);  
    i++;  
} while(i < lastNameCode.length);
```

# break, continue

---

- break : 반복문의 실행 중에 가장 근접한 반복문을 빠져 나옴

여러 반복문의 중첩 시, 현재에 위치한 반복문만 탈출

- continue : 반복 도중 처리를 중단하고 반복문의 시작 위치로 이동

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

int i = 1;
loop1 : while(i < 100) {
    i++;

    if((i % 2 == 0) && (i != pickedNum)) {
        continue loop1;
    }

    if(i == pickedNum) {
        System.out.println("Exit");
        break loop1;
    }

    System.out.println("number " + i);
}
```

# 실습

---



1. 각 학생들의 알파벳 점수(A, B, C, D, F)를 받는다.
  - 학생들의 수는 코드 내에서 정하지 않고 각 알파벳 점수를 띄어쓰기로 구분하여 입력하는 만큼 존재한다.
  - A, B, C, D, F 이외의 알파벳을 작성하는 경우는 없다고 가정한다.
  - 예시 : 입력 값이 A B c이면 3명이 존재하고, A B d C f이면 5명이 존재한다.
2. 알파벳 점수를 숫자 점수로 환산하여 다음 형식과 같이 출력한다.
  - 알파벳의 대소문자는 환산 전 toLowerCase()나 toUpperCase()로 통일해준다.
  - A는 100, B는 90, C는 80, D는 70, F는 0점으로 환산한다.
  - ppt를 참조하여 기수를 서수로 만드는 메소드를 활용한다.
3. 전체 학생들의 평균 값을 구하여 다음 형식에 맞게 소수점 두번째 자리까지 출력한다.
  - 소수점 두번째 자리까지 출력하는 방법 : `String fixedNum = String.format("0.2f", originalNum);`

while



# 실습

## 입력화면

 input  Output

A B d

 input  Output

A B d C f a

## 출력화면

```
1st student's score is 100
2nd student's score is 90
3rd student's score is 80
This class's average = 90.00
```

```
1st student's score is 100
2nd student's score is 90
3rd student's score is 70
4th student's score is 80
5th student's score is 0
6th student's score is 100
This class's average = 73.33
```