04장. 기본구문 - 조건문, 선택문

1. 조건문 if

자바의 기본 구문은 어떤 문제를 해결하기 위해 한 일련의 과정인 알고리즘을 기반으로 한다. 프로그램은 기본적으로 위에서 순서대로 실행 해 나가고 있다. 그러나 어떤 상황에서는 실행할 수 있는 상황과 실행 할수 없는 상황을 조건에 따라 실행 순서를 제어 할 수 잇다.

상황에 따라 적합한 **제어문** (Control flow statements)을 사용해서 문제를 해결하도록 하는 것이 **제어문** (Control flow statements)을 사용하는 목적이며 자바의 기본 구문들로 이루어 진다.
자바의 기본 구문은 조건문, 선택문, 반복문, 흐름제어구문으로 이루어진다.
그 중 조건문은 if문을 말하며 if문, if~else, 다중if~else, 중첩 if~else구문이 있다.

>> if문

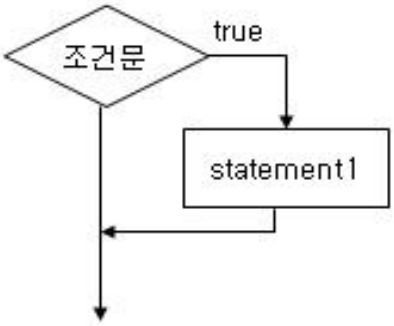
if 문은 조건 비교 분기문의 하나로 주어진 조건을 비교해서 조건의 리턴 값이 boolean 의 true 와 false 에 따라 명령을 제어하는 구문이다. if 문은 조건식의 결과가 true 인 경우에는 「{」에서 「}」의 블록 내에 기술 된 문장을 실행한다. 조건식이 false 인 경우에는 블록의 처리는 아무것도 실시하지 않고 if 문 다음 문장으로 처리가 이동한다.

또한 조건식이 true 일 때 수행 할 문장이 하나 즉 한 줄인 경우에는 "{"와 "}"를 생략할 수 있다.

[형식]

```
if ( 조건문) {
    // statement1 ->조건문의 결과가 true 일 때 명령이 실행
}
```

[흐름도]



[실행 구문 예]

변수 "num"의 80보다 큰 경우에는 블록의 처리를 실행하여 "합격입니다"라고 화면에 출력하는 코드

```
int num = 90;

if (num> 80) {  ---------→ 조건의 결과가 true 이기 때문에 실행결과는 "합격입니다"가 출력
    System.out.println ( "합격입니다");
}
```

변수 "mark" 가 50보다 크거나 같을 경우 "축하합니다, 합격입니다"를 출력 하는 코드

```
int mark =40;
if (mark >= 50) { ----------→ 조건의 결과가 false 이기 때문에 명령이 실행되지 않는다
    System.out.println("축하합니다 !");
    System.out.println("합격입니다.");
}
```

>> if문 ~ else

조건을 비교해서 조건을 만족하는 경우에만 어떠한 문장 statement1을 수행하고 조건을 만족하지 못한 경우에는 statement2를 수행한다.

[형식]

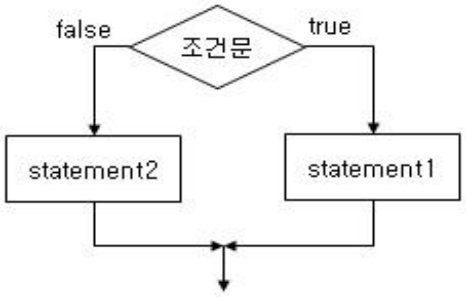
```
if (조건식) {
  true 의 경우 수행 할 문장 1;
  true 의 경우 수행 할 문장 2;
  ...
```

```
} else {
  false의 경우 수행 할 문장 1;
  false의 경우 수행 할 문장 2;
  ...
}
```

조건식을 평가하고 true 인 경우에는 조건 식 후에 블록 내의 문장을 실행하고 false 인 경우에는 else 후에 블록 내의 문장 실행한다 실행 문이 하나 인 경우에는 "{"와 "}"를 생략하여 다음과 같이 선언할 수 있다.

```
if (조건식)
  true의 경우 수행 할 문장;
else
  false의 경우 수행 할 문장;
```

[흐름도]



[프로그램 예]

변수 "num"의 값이 80보다 큰 경우에는 "합격입니다"라고 출력 하고 변수 "num"의 값이 80 이하인 경우에는 "불합격입니다"라고 화면에 출력하는 프로그램

```
int num = 90;

if (num> 80) { -----→ 리턴 결과가 true 이기 때문에 합격입니다가 출력
  System.out.println ( "합격입니다");
} else {
```

```
  System.out.println ( "불합격입니다");
}
```

다음과 같이 연산식의 결과값을 res라는 변수에 대입해서 조건문에 지정 할 수 있다. if(조건식)의 결과가 Boolean으로만 올 수 있기 때문에 여러 개의 조건식을 중첩해서 사용할 수 있다.

```
boolean res = (score> = 52);
if (res) {
                System.out.println ( "합격");
              } else {
                    System.out.println ( "불합격");
              }
```

예에서는 변수 i 에 할당 된 값이 j 의 값 이하이면 블록 A 가 실행되고 그렇지 않다면 블록 B 가 실행됩니다. 이 경우 블록 B 가 실행되게 합니다

```
package com.ch04;

public class ch04_1 {
   public static void main(String[] args) {
      int i = 10;
      int j = 5;
      System.out.println("Before");
      if (i <= j) {
         // 블록 A
         System.out.println("Win!");
      } else {
         // 블록 B
         System.out.println("Loose!");
      }
      System.out.println("After");

   }
}
```

-----→ 실행결과

```
Before
Loose!
After
```

>>다중 if~ else

다중 if~else 는 여러 조건식이 사용되는 경우에 사용됩니다

[형식]

```
if (조건식 1) {
  조건식 1 이 true 의 경우 수행 할 문장 1;
  조건식 1 이 true 의 경우 수행 할 문장 2;
  ...
} else if (조건식 2) {
  조건식 1 이 false 에서 조건식 2 가 true 의 경우 수행 할 문장 1;
  조건식 1 이 false 에서 조건식 2 가 true 의 경우 수행 할 문장 2;
  ...
}
else {
  모든 조건식이 모두 false 의 경우 수행 할 문장 1;
  모든 조건식이 모두 false 의 경우 수행 할 문장 2;
  ...
}
```

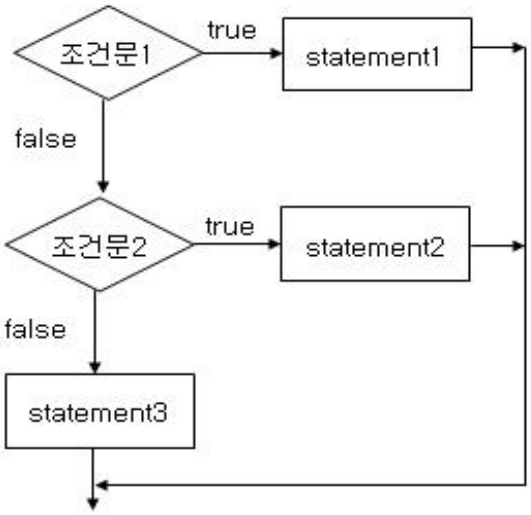
우선 조건식 1 을 평가 true 인 경우 직후 블록의 처리를 실행하는 if 문을 종료한다. . 조건식 1 의 평가가 false 인 경우에만 다음의 조건식 2 를 평가한다.. 조건식 2 가 true 인 경우에는 직후 블록의 처리를 실행 if 문을 종료한다. . 이 같이 순차적으로 조건식을 평가하고 모든 조건식이 false 인 경우에 else 후에 블록 내의 문장을 실행한다

또한 모든 조건식이 false 일 때 실행되는 else 절은 필요하지 않을 경우 생략 할 수 있다

```
if (조건식 1) {
  조건식 1 이 true 의 경우 수행 할 문장 1;
  조건식 1 이 true 의 경우 수행 할 문장 2;
  ...
} else if (조건식 2) {
  조건식 1 이 false 에서 조건식 2 가 true 의 경우 수행 할 문장 1;
  조건식 1 이 false 에서 조건식 2 가 true 의 경우 수행 할 문장 2;
```

```
   ...
}
```

[흐름도]



[프로그램 예]

i 는 68 로 설정되어 있으며, i 는 80 보다 작기 때문에 첫 번째 조건문 (80 <i)는 false 입니다. 따라서 "A"를 출력하는 줄은 실행되지 않는다. 다음 조건문 (60 <i)는 true 로 평가되므로 문자열 B 를 출력한다. <<4-2>>

```
package com.ch04;

public class ch04_2 {
   public static void main(String[] args) {
      int i = 68;
      if (80 < i) {
         System.out.println("A");
      } else if (60 < i) {
         System.out.println("B"); // B가 출력
      } else {
         System.out.println("C");
      }

   }
}
```

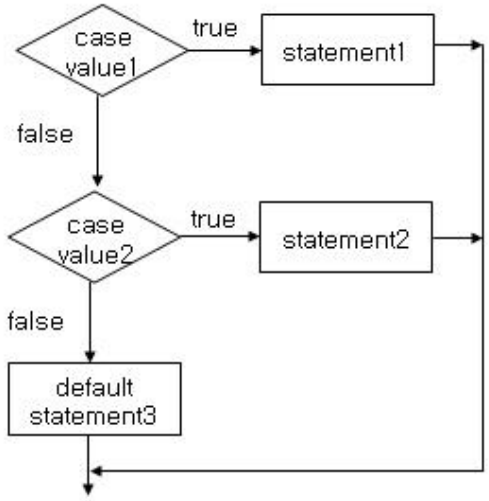
2. 선택문 _switch~ case문

switch 문은 특정 변수가 특정 값을 취하는 경우에 실행할 코드를 지정할 수 있다. 변수는 switch로 지정하여 조건이되는 값은 case로 지정한다. switch문은 다중조건 분기일 때, 블록if문를 대체하는 효과를 가지며 switch문안에 표현식을 기술하고 그 표현식의 결과값에 따라 그 값을 만족하는 case(경우)로 분기하는 형태를 사용한다.
표현식은 반드시 정수형 또는 문자열(String) 이어야 한다.
case 문에는 수행 해야하는 문장들이 나열되고 반드시 맨 마지막문장에는 break 문이 나와야 하며 break 문이 나오지 않으면 아래의 문장들을 계속 수행하기 때문에 원하는 형태의 결과가 나오지 않을 수 있다.
모든 case 문에 해당되지 않는 경우를 위해서 default 문을 사용한다. default 문도 break 문을 가진다.

[형식]

```
switch (식) {
  case 상수 1 :
    실행 문 1;
    실행 문 2;
    ...
    break;
  case 상수 2 :
    실행 문 1;
    실행 문 2;
    ...
    break;
}
```

[흐름도]



[프로그램 예]

```
int num = 5;
switch (num) {// 변수 값이 식을 계산 한 값
  ...
}
switch (num % 2) {// 연산 결과가 식을 계산 한 값
  ...
}
```

다음은 switch 문장의 "{"부터 "}"블록 내에 수행 할 작업을 나열하고 있다. 실행한 결과는 case 문이 없어 숫자 5 에 해당하는 문장으로 판단하여 모든 값을 출력한다.

```
int num = 5;

switch (num) {
  System.out.println ( "매우 불만족");
  System.out.println ( "약간 불만");
  System.out.println ( "어느 쪽이라고도 말할 수 없다");
  System.out.println ( "약간 만족");
  System.out.println ( "매우 만족");
}
```

switch 문은 식을 평가 한 값과 일치하는 라벨로 처리를 이동시키기 위해 사용된다. 그래서 어떤 값이라면 어떻게 처리 할 것인지를 지정하기 위해 블록에 레이블을 설명하고 있다.. 레이블은 다음과 같은 형식으로 사용된다.

```
case 상수 :
```

블록에 레이블을 작성하면 식을 계산 한 값과 일치하는 레이블 곳에 처리를 옮길 수 있다. 예를 들어 5 개의 레이블을 지정하면 다음과 같이 사용된다.

```
int num = 3;

switch (num) {
  case 1 :
    System.out.println ( "매우 불만족");
  case 2 :
    System.out.println ( "약간 불만");
  case 3 :
    System.out.println ( "어느 쪽이라고도 말할 수 없다");
  case 4 :
    System.out.println ( "약간 만족");
  case 5 :
    System.out.println ( "매우 만족");
}
```

라벨에 쓴 값이 "1" "2" "3" "4" "5"이므로, 식의 값이 1 에서 5 의 경우에 대응하는 라벨의 위치로 처리가 이동한다. 이번 경우 변수 "num"의 값이 3 이므로 "case 3 :"레이블의 위치로 이동되어 "System.out.println ("어느 쪽이라고도 말할 수 없다 ");"가 실행된다.

주의해야 할 것은 어디 까지나 라벨이 적힌 위치에 처리가 이동 뿐이라는 것이다. 라벨의 위치로 이동 한 후 switch 문장의 끝까지 그 이후의 문장을 순서대로 수행한다. 따라서 "System.out.println ("어느 쪽이라고도 말할 수 없다 ");"이 실행 된 후 "System.out.println ("약간 만족 "); '과'System.out.println ("매우 만족 ") ; "가 실행된다.

만약 레이블 위치로 이동 한 후 다음 레이블 앞에까지 오면 switch 문을 종료 할 경우에는 break 문을 사용한다. switch 문에서 break 문이 실행되면 switch 문을 종료한다. 구체적으로는 다음과 같이 사용된다.

```
int num = 3;

switch (num) {
  case 1 :
    System.out.println ( "매우 불만족");
    break;
  case 2 :
    System.out.println ( "약간 불만");
    break;
  case 3 :
    System.out.println ( "어느 쪽이라고도 말할 수 없다");
    break;
  case 4 :
    System.out.println ( "약간 만족");
    break;
  case 5 :
    System.out.println ( "매우 만족");
    break;
}
```

위와 같이 break 문을 기술하면 break 문을 실행 한 시점에서 switch 문은 종료된다. 따라서 위의 경우는 "System.out.println ("어느 쪽이라고도 말할 수 없다 ");"이 실행 된 후 "break;"가 실행되고 switch 문은 종료된다.

식의 값과 일치하는 레이블이 없었던 경우는 아무것도 실행하지 않고 switch 문을 종료하지만 일치하는 레이블이 없을 경우에는 뭔가 실행하려는 프로세스가 있는 경우에는 default 절을 사용한다. 형식은 다음과 같다.

```
switch (식) {
  case 상수 1 :
    실행 문;
    ...
    break;
  case 상수 2 :
    실행 문;
```

```
    …
    break;
  default :
    실행 문 1;
    실행 문 2;
    …
}
```

위의 경우 식을 평가 한 결과 값이 상수 1 에도 정수 2 와 일치하지 않은 경우에는 "default :"위치로 처리가 이동한다. 그리고 이후에 쓰여진 문장이 순서대로 수행된다.

예를 들어 다음과 같이 **사용된다.**

```
int num = 3;
switch (num) {
  case 4 :
    System.out.println ( "약간 만족");
    break;
  case 5 :
    System.out.println ( "매우 만족");
    break;
  default :
    System.out.println ( "만족한다");
}
```

위의 경우 변수 "num"의 값을 4 또는 5 이외의 경우에는 모두 "System.out.println ("만족한다");"문장을 실행한다.