제어문(Control Statements)

-제어문은 프로그램 코드의 흐름을 바꿀 때 사용되는 문장이다.

-제어문에는 조건문과 반복문이 있다.

조건문(Branching Statements)

-조건문은 조건에 따라 수행하는 문장을 다르게 할 수 있다.

-if문, if-else문, if-else if문, swithch문 등이 조건문에 해당한다.

\* if문 형식

if(조건식){

//if옆 조건식이 참일 때 수행될 문장들

}

\* if – else문 형식

if(조건식){

//if옆 조건식이 참일 때 수행될 문장들

} else{

// if 조건식이 거짓일 때 수행될 문장들

}

\* if – else if – else 문 형식

if(조건식1){

// 조건식1이 참일 때 수행될 문장들

} else if(조건식2){

// 조건식2가 참일 때 수행될 문장들

} else if(조건식3){

// 조건식3이 참일 때 수행될 문장들

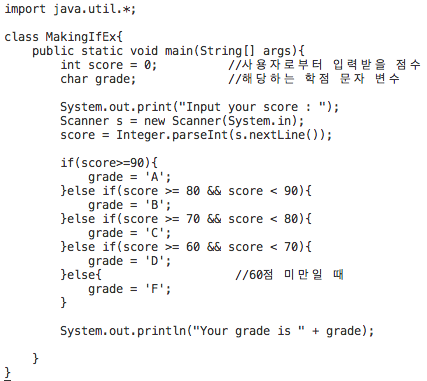
} else {

// 위의 조건식 중 어느 것도 만족하지 않을 때 수행될 문장들

}

조건이 여러가지이면 그에 따라 else-if문으로 조건식을 여러개 지정해줄 수 있다. 조건식이 참인 문장을 수행하면 해당하는 블럭만 수행하고 전체 조건문을 빠져나간다. 또, 모든 조건식을 만족하지 않을 땐 else문을 수행하게 되고, else문을 쓰지 않으면 모든 if-else if문을 빠져나가게 된다.

(예제)

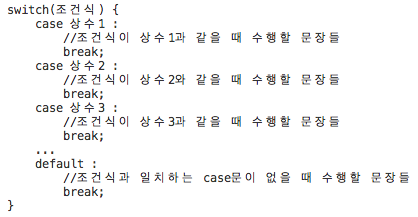


(결과)

../../Desktop/스크린샷%202019-01-18%20오후%204.46.22.pn ../../Desktop/스크린샷%202019-01-18%20오후%204.46.33.pn

+예제에서 볼 수 있듯이 사용자가 입력한 점수를 조건식과 비교하고 해당 조건식이 있는 문장을 수행한다. 84를 입력하면 첫번째 조건문은 건너뛰고 두번째 조건식이 참이되어 grade 변수에 학점을 저장하고 나머지 조건식은 검사할 필요없이 모든 조건문을 빠져나오게 된다. 50을 입력하면 모든 조건식을 비교해봐도 참이 없기 때문에 마지막의 else문의 블럭을 수행하게 된다.

\* switch문 형식

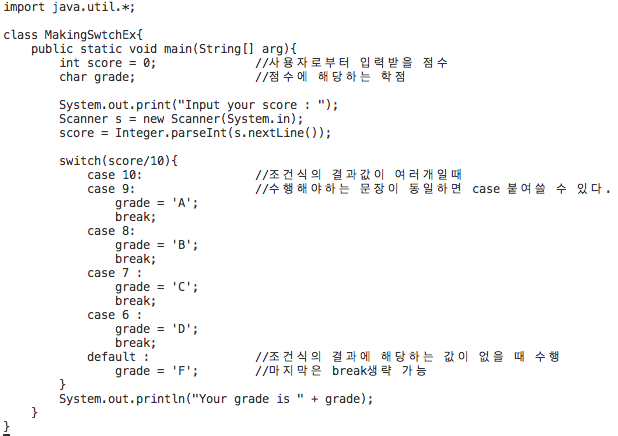


1)조건식을 먼저 계산한 후에, 2)나온 결과값과 일치하는 case문으로 이동하여 수행하고, 3)break문을 만나거나 switch문을 끝까지 수행한 후 전체 switch문을 빠져나온다.

만약 각 case문에 break;를 쓰지 않으면 다른 break를 만날때까지, 혹은 switch문 전체를 수행할 때까지 모든 문장을 수행해야하므로 각 case문의 끝에는 break;를 빠뜨리는 실수를 하지 않도록 해야한다.

if문의 else블럭처럼, 조건식과 만족하는 값이 없을 때 수행할 문장은 default블럭에 작성할 수 있다. default블럭은 마지막에 놓기 때문에 break;를 생략해도 무방하다.

switch문의 조건식 결과는 무조건 정수나 문자열만 가능하고, case문의 상수값은 정수인 상수만 가능하고 중복되면 안된다는 점에 주의해야한다.



../../Desktop/스크린샷%202019-01-18%20오후%206.24.02.pn ../../Desktop/스크린샷%202019-01-18%20오후%206.24.12.pn

+위의 예제 if-else문과 결과가 동일한 switch문이다. switch문의 조건식엔 상수로 만들기 위해 10으로 나눠주고, 90점 이상 100점 이하인 case는 10과 9이므로 두 case문중 하나만 값이 일치해도 해당 문장을 수행할 수 있다.

73을 입력하면 73/10이 되어 결과값이 7이 되고, case 7구문을 수행하게 된다. break를 만났으므로 switch문 전체를 빠져나간다.

반복문(Looping Statements)

-반복문은 특정 문장들을 반복해서 수행할 수 있다.

-for문, while문, do-while문 등이 있다.

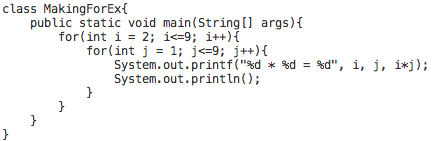
-for문과 while문은 몸체가 실행되기 전에 조건식검사를 하고 수행되는데, do-while문은 몸체를 일단 무조건 한번 실행한 후 조건식을 검사한다.

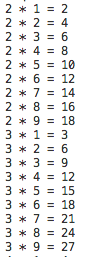
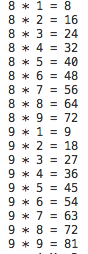
-for문과 while문은 구조와 기능이 매우 유사한데 for문은 주로 반복횟수를 알고 있을 때 사용한다.

\*for문 형식

../../Desktop/스크린샷%202019-01-21%20오전%2011.26.16.pn

1. for문에 사용될 변수 하나를 초기화한다. 처음에 한번만 초기화된다.
2. 조건식이 참인지 거짓인지에 따라 반복문이 수행될지 여부를 결정한다. 참인동안 for문이 반복되게 된다.
3. 조건식이 참이면 for문안의 문장들이 수행되고, 마지막까지 모두 수행되면 중감식에 의해 반복문을 제어하는 변수의 값을 증가 또는 감소시킨다.
4. 증감식에 의해 변한 변수가 조건식에 맞나 다시 검사를 하게 된다.
5. for문 진행 순서는 맨처음 초기화는 한 번만 하고, 그 이후는 계속 조건식, 루프 안의 문장들, 증감식 순으로 진행된다. 그러다 조건식이 거짓이 되면 바로 루프 밖으로 빠져나오게 된다.



 ……….. 

+ for문 안에 for문을 넣는 것도 가능하다. 이중for문이라고도 하는데 변수를 두 개 이상 증감해야 할 때 쓰인다.

위의 예제는 변수 i는 몇 단인지를 출력하기 위한 변수이고, j는 1부터 9까지 곱해주는 변수이다. 9까지 다 곱하면 이중for문에 의해 다음 단으로 넘어가게된다.

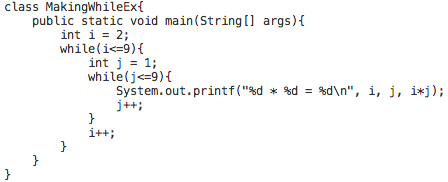
\*while문 형식

/Users/swq/Desktop/스크린샷 2019-01-21 오후 1.03.36.png

-for문과 원리는 굉장히 유사하지만 while문은 조건식만 있고 증감식은 루프 안에서 따로 써주어야한다.

-우선 조건식을 검사하고 참이면 루프안의 문장들을 수행하고 다시 조건식으로 가서 검사를 하게 된다.

-조건식이 거짓이면 루프 밖으로 빠져나가게 된다.



+for문의 예제를 똑같이 while문으로 바꾼 예제이다. 결과는 for문 실행결과와 완전히 동일하다. 초기화는 경우에 따라 while문 안쪽에 써야할 때도 있고, while문 시작 전에 써야할 때도 있다.

i는 몇 단인지를 나타내는 변수이고, j는 1부터 9까지 곱해줘서 출력하는 변수이다. 각각의 조건식에 따라 변수의 증감식도 따로 써주어야한다.

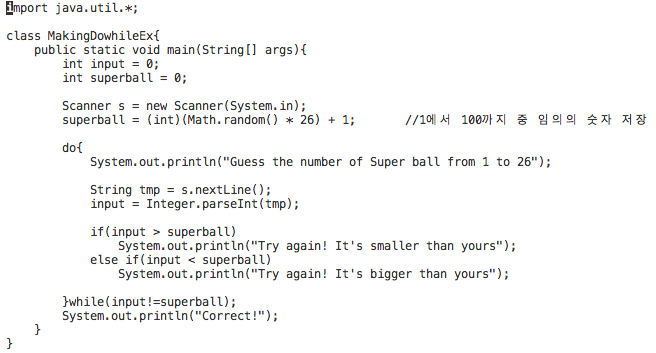
\*do-while문 형식

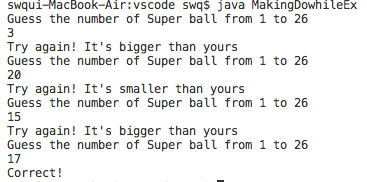
/Users/swq/Desktop/스크린샷 2019-01-21 오후 1.32.34.png

-while문과는 달리 while과 조건식이 루프의 끝에 있다.

-반드시 루프를 한 번 수행한 후 조건식을 검사하고, 참이면 루프를 또 수행하고 거짓이면 루프밖으로 빠져나가게 된다.

-while 옆에 조건식을 쓰고 반드시 세미콜론을 찍어줘야한다.





+사용자가 슈퍼볼 번호를 맞추게 하는 프로그램이다. 랜덤으로 슈퍼볼 번호를 하나 생성을 하고 사용자에게 입력을 받으면 비교해서 맞출때까지 번호를 입력받는다.

사용자로부터 무조건 한번은 입력을 받아야하기 때문에 do-while문을 썼고 조건식에는 사용자로부터 입력받은 숫자와 랜덤으로 생성한 숫자가 일치하지 않을동안 반복문을 수행하도록 했다.