07.27 노트정리

1. 제어문

실행문의 **수행 순서를 변경**할 때 사용하는 것이다.
(**일반적으로 프로그램**에 포함된 실행문은 **순차적으로 실행**이 되어 프로그램이 매우 길어져 표현하기 어려운 점이 있다. 이 어려운 점을 **보완하기 위해 생긴 것이 제어문**이다.)
ex) 선택적 행동, 무한대로 더하기 등등

종류 : 조건문(=선택문), 반복문, 분기문

2. 조건문 (if, switch)

조건식의 결과에 따라 여러 실행 경로 중 하나를 선택하는 제어문이다.

단순 if문

조건식이 true일 때만 실행문 수행할 때 사용한다.

```
if(조건식) {
수행문;    // 조건식이 '참'인 경우 수행
}    // 조건식이 '거짓'인 경우 수행 안함
```

if~else문

조건식의 true나 false에 따라 다른 실행문을 수행할 때 사용하는 조건문이다.

```
if (조건식) {
수행문1;    // 조건식이 '참'인 경우 수행
}
else {
수행문2;    // 조건식이 '거짓'인 경우 수행
}
```

다중 if문

if문 다음에 else if문을 연속 추가해 각 조건을 차례대로 점검한 후 만족하는 실행문을 수행하는 조건문이다. (**조건이 다양할 때 사용**)

```
if(조건식1) {
실행문1;    // 조건식1이 '참'인 경우 수행 + 전체 조건문 빠져 나감
}
else if (조건식2) {    // else if문은 개수 제한 없음
실행문2;    // 조건식1이 '거짓' + 조건식2가 '참'인 경우 수행 + 전체 조건문 빠져나감
}
else {
실행문3;    // 위 조건들을 모두 만족하지 않는 경우 수행(default)
}
```

07.27 노트정리

중첩 if문

if문 내에 다른 if 문이 포함시켜 참의 결과값에 대한 세부적인 조건 작성하는 데 사용하는 조건문이다.

```
if (조건식1) {
  if(조건식2) {    // if문 내에서 세부적인 조건 작성
    실행문;
  }
  else {
    실행문;
  }
}
```

switch문(=다중 if문)

switch 문은 여러 경로 중 하나를 선택할 때 사용하고 다중 if 문으로 구현 가능한 조건문이다.
(조건이 상당히 많아 if 문으로 표현하기 힘들 때 간단하게 표현할 수 있는 조건문임)

```
switch (변수명) {
case n :    // {}대신 콜론(:) 사용
  실행문;
    break;    // break문이 없으면 다른 case까지 미끄러지듯 차례대로 실행
default :
  실행문;
}
```

ex) break 사용 무

```
int floors = 2;
switch (floors) {
case 1 :
  System.out.println("1층입니다.");
case 2 :
  System.out.println("2층입니다.");
case 3 :
  System.out.println("3층입니다.");
default :
  System.out.println("고층입니다.");
}
```

=> 1층입니다.
2층입니다.
3층입니다.
고층입니다.
1층입니다.
...

ex) break 사용 유

```
int floors = 2;
switch (floors) {
case 1 :
  System.out.println("1층입니다.");
    break;
case 2 :
  System.out.println("2층입니다.");
    break;
case 3 :
  System.out.println("3층입니다.");
    break;
default :
  System.out.println("고층입니다.");
}
```

=> 2층입니다.

07.27 노트정리

3. 반복문 (for, while)

조건에 따라 같은 처리를 반복하는 제어문이다.

for문

반복할 횟수를 알 수 있을 때 주로 사용하는 반복문이다.

(선언된 변수를 조건만큼 증가하거나 감소하는 방향으로 for문 내의 실행문을 계속 반복시킬 수 있음)

```
for (초기식(변수선언); 조건식; 증감식) {    // 세미콜론으로 구분
  반복 실행문;
}
```

초기식 : for문이 실행될 때 **단 1회 실행**

조건식 : **n회 반복되기 전에** 조건을 확인하여 **참이면 코드 블록 실행, 거짓이면 for문을 빠져나옴**

증감식 : **n회 반복된 후 실행되어 변수에 변화를 주는 식**

ex)
```
for(int i = 1; i <= 5; i++) {
    System.out.print(i);
}
```

=> 12345

* 1 ->변수 적용식 출력

2~5 -> 증감식 적용 + 조건식 적용 출력 (조건에 맞을 때까지 적용하며 출력)

while문

반복할 **횟수는 미리 알 수 없지만 조건은 알 수 있을 때** 주로 사용하는 반복문이다.

(초기 변수를 사용하기 위해서는 **변수를 while문 밖**에 선언해야 하고 **증감식은 코드 블록 안**에 넣어야 함)

```
while (조건식) {    // 조건식이 true일 때 멈추는 방법 : break 사용하여 멈춤
  반복 실행문;}
```

ex)
```
int a = 1;
while (a<=5) {
  System.out.println(a);
  a++;
}
```

=> 12345

ex)
```
int b=10;
while(true) {    //무한반복
  System.out.print(b);
  break;
}
```

=> 10101010101010...

ex)
```
int b = 10;
while(true) {    //무한반복
  System.out.print(b);
  b++
  if(b%2 == 0) {
    break;
  }
}
```

=> 1011

*10은 **먼저 출력**이 되어 **10이 출력**이 되었고

11은 **2로 나누었을 때 1이** 나와서 **11도 출력**된다.

그리고 **12는 2로 나누었을 때 0이** 나오기 때문에 **12는 출력이 안된다.**

07.27 노트정리

for문과 while문의 차이점

1. 무한루프

while문에는 정확히 참인 문장을 써줘야 하고 for문에는 아무것도 써주지 않아도 됩니다.
(for도 while처럼 1 또는 참인문장을 써줘도 됩니다.)

2. 초기식의 유무

```
for (int i = 0; i < 3; i++){     //for문은 유
  sum += i;
}

int i = 0;                       //while문은 무
while (i < 3){
  sum += i;
  i++;
}
```

3. 변수 i의 수명의 차이점

for문에서의 변수의 수명은 **for문 안에서만 생존 가능**하다.
while문에서의 변수의 수명은 **while문 밖에서도 생존 가능**하다.

4. 정리

(A: 초기식(변수선언), B: 조건식, C: 증감식, D: 실행문)

for문	**while문**
for (A; B; C){ D;}	A; while (B){ D; C;}

do~while문

while문과 형태는 동일하지만 while문과는 다르게 조건식에 상관없이 **do라는 실행문을 무조건 한번은 실행**하고 반복 시작하는 반복문이다.

```
do {   // 조건에 상관없이 무조건 한번은 실행
  반복 실행문;
} while (조건식);
```

ex)
```
int a = 7;
do {
    System.out.print(i);
    a++;
} while(a<5);
```

=> 7

07.27 노트정리

4. 분기문 (break, continue)

실행 흐름을 무조건 변경하는 제어문이다.

break문

switch문에서는 본체를 벗어나려고 break 문을 사용해 왔는데, 반복문에서 반복을 종료할 때도 사용하는 분기문이다.

ex) while문 조건식이 true일 때 멈추는 방법 : break 사용하여 멈춤
swich문 중 case문을 사용할 때 **멈추는 방법 : break를 해줘야함,**
사용 안하면 모두 실행해야함 (case문은 **{}이 아닌 :을 사용함)**

continue문(=띄어쓰기문)

반복문(for문, while문, do-while문)에서만 사용하고 **현재 반복은 건너 뛴 채 나머지 반복만 계속 실행**하는 분기문이다.

* for 문 - 증감식으로 바로 이동 /while, do~while - 조건식으로 바로 이동

ex)
```
for(int a=0; a<10; a++) {
        if(a%2 == 0)  {
            continue;
        }
        System.out.print(i);
    }
```

=>13579

07.27 노트정리

4주차 과제

첫 번째 과제

숫자를 입력받고 숫자가 0보다 크면 “양수” 작으면 “음수” 둘 다 아니라면 “0”을 출력해보기

KSI_0727_1.java

```java
package week4;
// 숫자를 입력받고 숫자가 0보다 크면 “양수” 작으면 “음수” 둘 다 아니라면 “0”을 출력해보기
import java.util.Scanner;

public class KSI_0727_1 {

    public static void main(String[] args) {
        int a;

        Scanner sc = new Scanner(System.in);
        System.out.println("정수를 입력하세요");
        a= sc.nextInt();
            if(a>0) {
                System.out.println("양수");
            }
            else if(a<0) {
                System.out.println("음수");
            }
            else {
                System.out.println("0");
            }
        }

}
```

Console

```
<terminated> KSI_0727_1 [Java Application] C:₩Users₩suim7₩OneDrive₩바탕 화면₩잡동산이₩eclipse-java-2021-06
정수를 입력하세요
-1
음수
```

두 번째 과제

1부터 10까지 모두 더한 값 출력해보기

KSI_0727_2.java

```java
package week4;
// 1부터 10까지 모두 더한 값 출력해보기
public class KSI_0727_2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        for(int a=1; a<11; a++) {
            System.out.println(a);
        }
    }

}
```

Console

```
<terminated> KSI_0727_2 [Java Application] C:₩Users₩suim7₩OneDrive₩바탕 화면₩잡동산이₩eclipse-java-2021-06-
1
2
3
4
5
6
7
8
9
10
```