

자바 기초

April. 9, 2007

제어문 (조건문, 반복문 그리고 분기문)

학습 목표

■ 학습 목표

- 프로그램을 제어하는 방법을 배운다.
- 조건문, 반복문, 분기문에 의해 같은 일을 좀 더 효율적으로 처리할 수 있다.
- if문, switch문, for문, while문, do – while문, continue문, break문 등에 대하여 알고, 적시 적절하게 사용할 수 있다.

제어문이란?

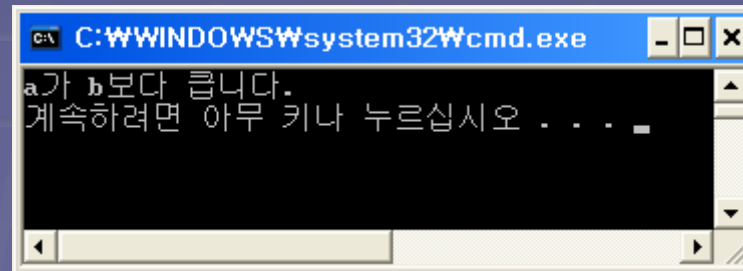
- Control statement
- 프로그램의 실행 순서를 바꾸는 명령문
- 종류
 - 조건문 : if문, if else문, switch문
 - 반복문 : for문, while문, do~while문
 - 분기문 : break문, continue문

조건문

- if문
 - 조건에 따라 판단하여 분기하는 명령문
 - 종류
 - if문
 - if else문
 - 중첩된 if else문
 - else if문
 - ()안에 있는 수식이 참(0이 아닌 모든 값)이면 if 문의 {} 영역 안의 문장을 실행하고 거짓이면 실행하지 않는다. 거짓이면 else 문의 {} 영역 안의 문장을 실행한다.

조건문

```
■ public class IfExam {  
    ■ public static void main(String args[]){  
        ■ int a = 5;  
        ■ int b = 4;  
        ■ if( a > b){  
            ■ System.out.println("a가 b보다 큼니다.");  
        ■ }  
        ■ if( a == b){  
            ■ System.out.println("a와 b가 같습니다.");  
        ■ }  
        ■ if( a < b){  
            ■ System.out.println("a가 b보다 작습니다.");  
        ■ }  
    ■ }  
■ }
```

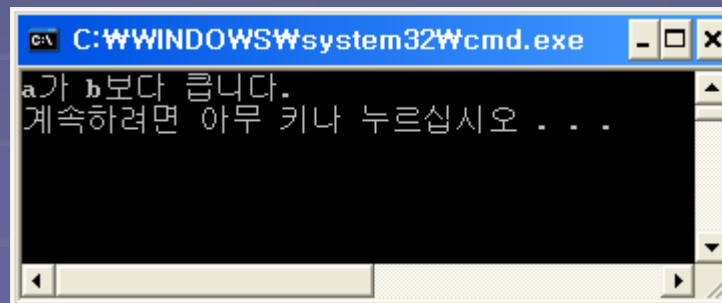


조건문

- else if문
 - if문을 여러 개 사용해야 할 때, 코드를 훨씬 더 효율적으로 만들 수 있음
 - if(조건) {
 - 문장1;
 - }else if(조건){
 - 문장2;
 - }

조건문

- public class IfExam {
 - public static void main(String args[]){
 - int a = 5;
 - int b = 4;
 - if(a > b){
 - System.out.println("a가 b보다 큼니다.");
 - }else if(a == b){
 - System.out.println("a와 b가 같습니다.");
 - }else if(a < b){
 - System.out.println("a가 b보다 작습니다.");
 - }
 - }
- }



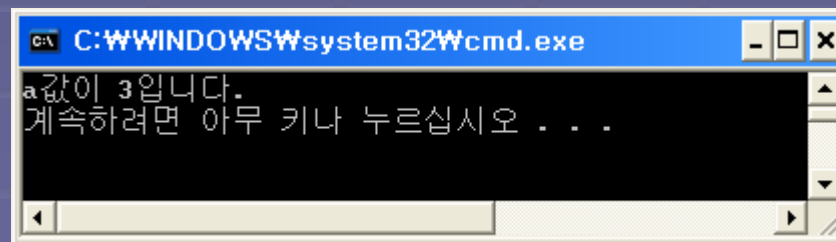
조건문

- switch문

- 비교해야 할 내용이 여러 개일 때 효과적으로 사용
- 비교 값이 정수이거나 문자인 경우 사용
- switch(비교할 변수){
 - case 값1:
 - 실행문1;
 - break;
 -(여러 개의 case가 올 수 있다.)
 - default:
 - 실행문2;
 - break;
- }
- Switch문의 조건식의 데이터 타입은 int 타입과 char 타입만 사용할 수 있습니다.(데이터 타입의 제한)

조건문

```
■ public class SwitchExam {  
    ■ public static void main(String args[]){  
        ■ int a = 3;  
        ■ switch(a){  
            ■ case 1: System.out.println("a값이 1입니다.");  
            ■     break;  
            ■ case 2: System.out.println("a값이 2입니다.");  
            ■     break;  
            ■ case 3: System.out.println("a값이 3입니다.");  
            ■     break;  
            ■ default: System.out.println("a값을 모르겠습니다.");  
            ■     break;  
        ■ }  
    ■ }  
■ }
```



반복문

- for문

- 실행문을 반복하는 횟수를 따로 지정하는 경우에 사용
- 가장 많이 사용되는 기본적인 반복문
- `for( 변수 시작(from); 변수 끝(to); 변수 값 변화){}`

반복문

- for문

- public class ForExam{

- // 1부터 10까지 더한 결과 보기

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

- int sum=0, inc=1;

- for(inc=1;inc<=10;inc++){

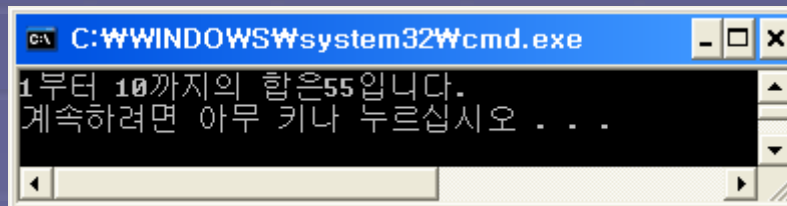
- sum=sum+inc;

- }

- System.out.println("1부터 10까지의 합은"+sum+"입니다.");

- }

- }



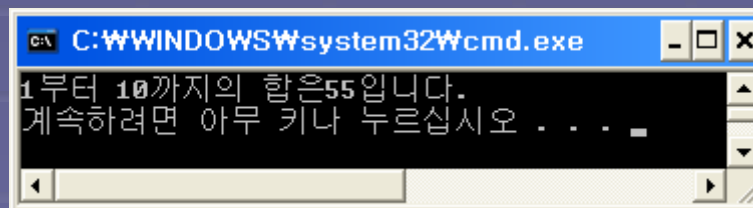
```
C:\WINDOWS\system32\cmd.exe
1부터 10까지의 합은55입니다.
계속하려면 아무 키나 누르십시오 . . .
```

반복문

- while문
 - 실행문을 계속 되풀이해서 수행할 수 있음
 - 반복문을 끝내는 방법은 조건에 의함
 - 조건이 참(0이 아닌 모든 값)이면 {}안이 실행되고, 거짓(0의 값)이면 {}을 실행하지 않고 반복문을 끝낸다.
 - while(조건){
 - 실행문;
 - }

반복문

- public class WhileExam {
 - public static void main(String args[]){
 - int sum = 0;
 - int inc=1;
 - while(inc <=10){
 - sum = sum + inc;
 - inc++;
 - }
 - System.out.println("1부터 10까지의 합은" + sum + "입니다.");
 - }
 - }



반복문

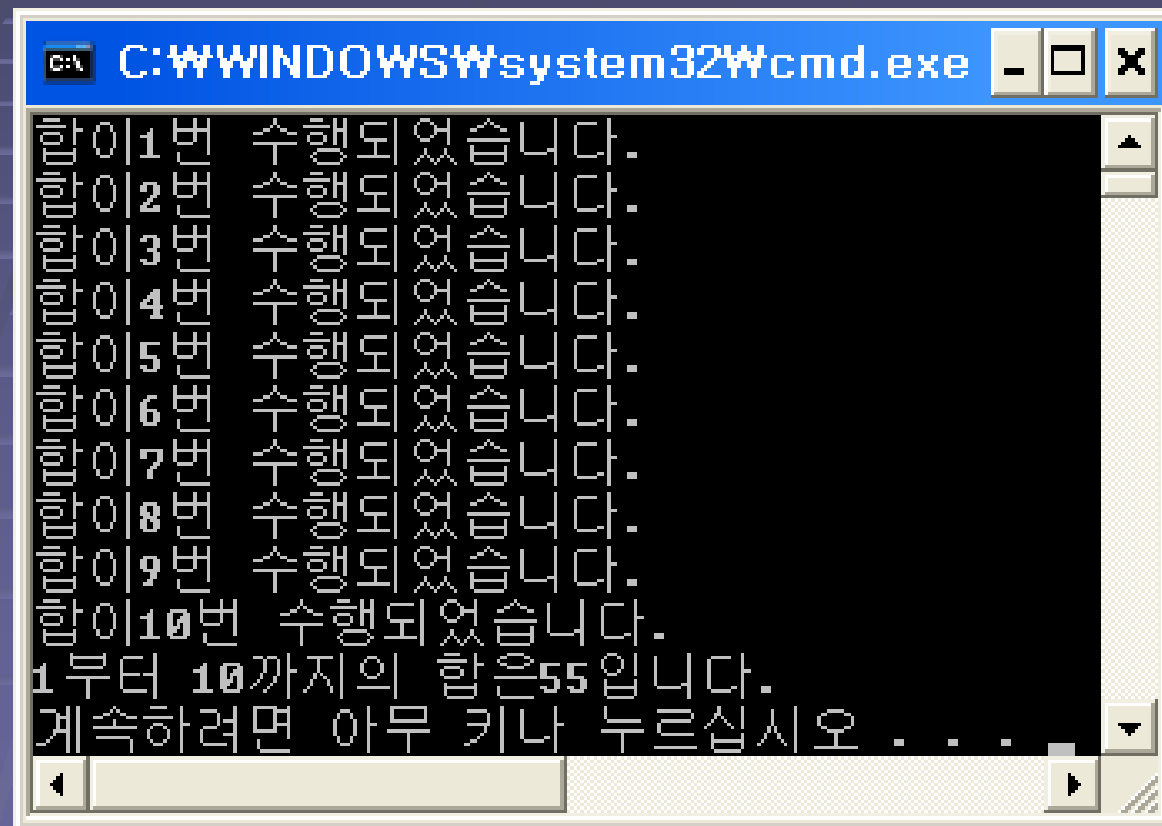
- do~while문
 - while문과의 차이점은 최소한 한번은 실행됨
 - 실행문을 계속 되풀이해서 수행할 수 있음
 - 반복문을 끝내는 방법은 조건에 의함
 - 조건이 참(0이 아닌 모든 값)이면 {}안이 실행되고, 거짓(0의 값)이면 {}을 실행하지 않고 반복문을 끝낸다.
 - do{
 - 실행문;
 - } while(조건)

반복문

- public class DoWhileExam {
 - public static void main(String args[]){
 - int sum = 0;
 - int inc=1;
 - do{
 - sum = sum + inc;
 - System.out.println("합이" + inc + "번 수행되었습니다.");
 - inc++;
 - }while(inc <=10);
 - System.out.println("1부터 10까지의 합은" + sum + "입니다.");
 - }
- }

반복문

- 결과창



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window displays the output of a program that calculates the sum of numbers from 1 to 10 using a loop. The output is as follows:

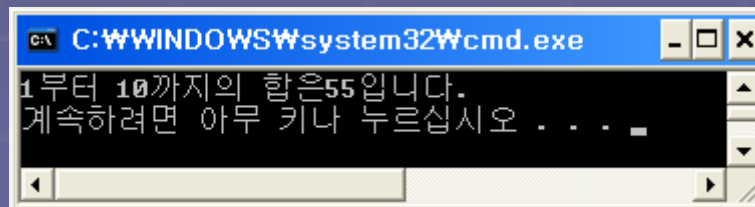
```
합이 1번 수행되었습니다.  
합이 2번 수행되었습니다.  
합이 3번 수행되었습니다.  
합이 4번 수행되었습니다.  
합이 5번 수행되었습니다.  
합이 6번 수행되었습니다.  
합이 7번 수행되었습니다.  
합이 8번 수행되었습니다.  
합이 9번 수행되었습니다.  
합이 10번 수행되었습니다.  
1부터 10까지의 합은 55입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

분기문

- break문
 - switch case문, for문, while문, do~while문의 영역을 벗어나기 위하여 사용하는 명령문
 - break문 하나 당 가장 가까운 곳에 있는 한 개의 루프를 벗어나므로 중첩된 루프에서는 중첩된 수만큼의 break문이 필요하다.

분기문

```
public class BreakExam {  
    public static void main(String args[]){  
        int sum = 0;  
        int inc=1;  
        while(true){  
            sum = sum + inc;  
            inc++;  
            if(inc > 10){  
                break;  
            }  
        }  
        System.out.println("1부터 10까지의 합은" + sum + "입니다.");  
    }  
}
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The output of the Java program is displayed as follows:

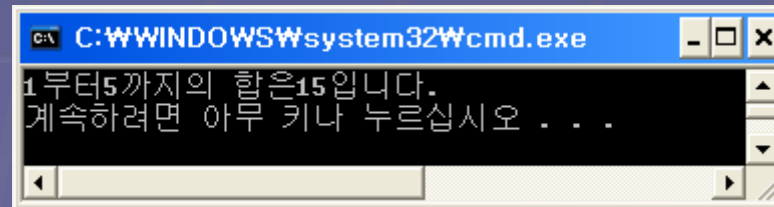
```
1부터 10까지의 합은55입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

분기문

- continue문
 - Break문과 반대 개념
 - 가장 가까이에 놓인 반복문의 조건부로 이동함

분기문

- public class ContinueExam {
 - public static void main(String args[]){
 - int sum = 0, tonum=5;
 - int inc=1;
 - for(inc=1;inc < 100; inc++){
 - if(inc > tonum){
 - continue;
 - }
 - sum = sum + inc;
 - }
 - System.out.println("1 부터"+ tonum+"까지의 합은" + sum + "입니다.");



수행 과제

- 자바 애플릿으로 (200,200) 포인트를 왼쪽 위 기준으로 지름 20인 원을 그리고, 지름이 1씩 점점 커져서 150까지 커졌다가, 다시 20으로 작아진 후 사라지도록 프로그램을 작성한 후, 실행화면 및 소스코드를 Lecture service에 제출하시오.
 - 단, Graphics g로 선언하고,
 - g.setColor 메서드와 g.drawOval 메서드를 활용하시오.
 - Ex)
 - g.setColor(Color.blue);
 - g.drawOval(100,100,zirum,zirum);