03. 자바 제어문

제어문 - if, switch

- 조건문은 조건식과 실행될 하나의 문장 또는 블럭{}으로 구성
- if문이 주로 사용되며, 경우의 수가 많은 경우 switch문을 사용할 것을 고려
- 모든 switch문은 if문으로 변경이 가능하지만, if문은 switch문으로 변경할 수 없는 경우가 많다.

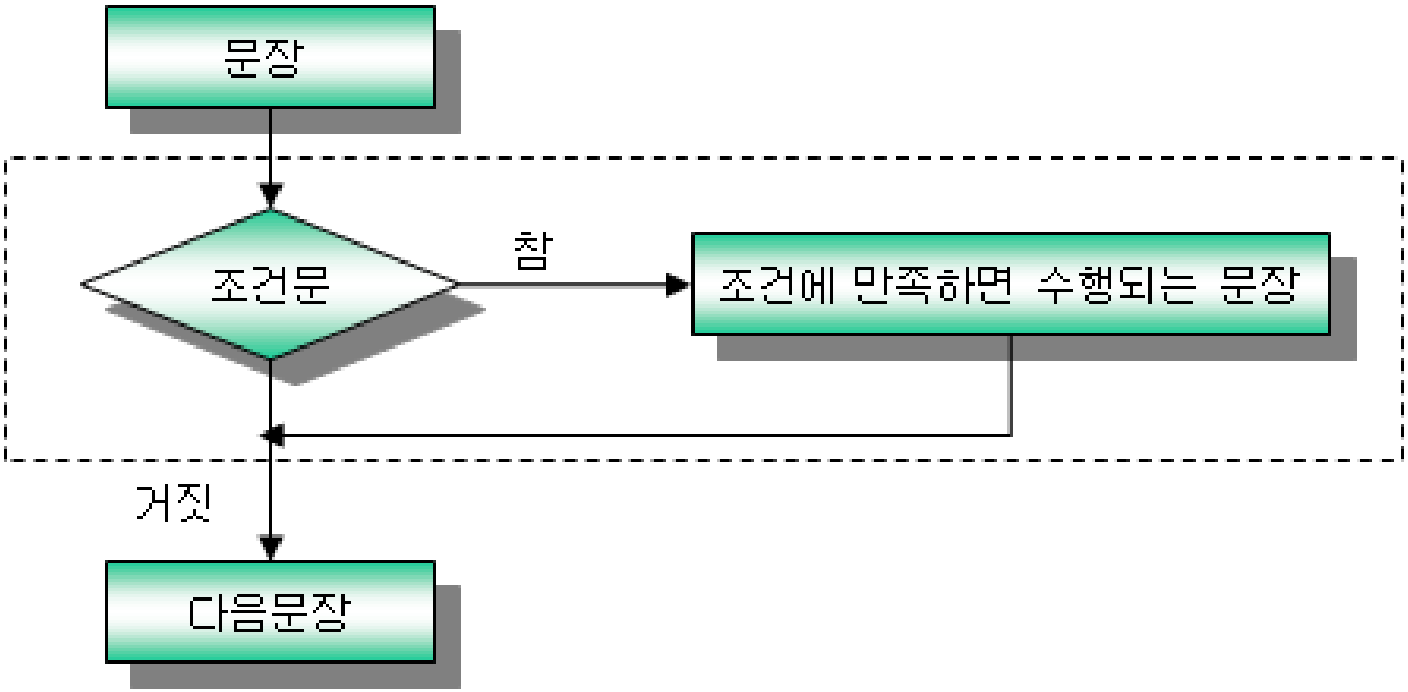
```
if(num==1) {
    System.out.println("SK");
} else if(num==6) {
    System.out.println("KTF");
} else if(num==9) {
    System.out.println("LG");
} else {
    System.out.println("UNKNOWN");
}
```

```
switch(num) {
    case 1:
        System.out.println("SK");
        break;
    case 6:
        System.out.println("KTF");
        break;
    case 9:
        System.out.println("LG");
        break;
    default:
        System.out.println("UNKNOWN");
}
```

1] if 문

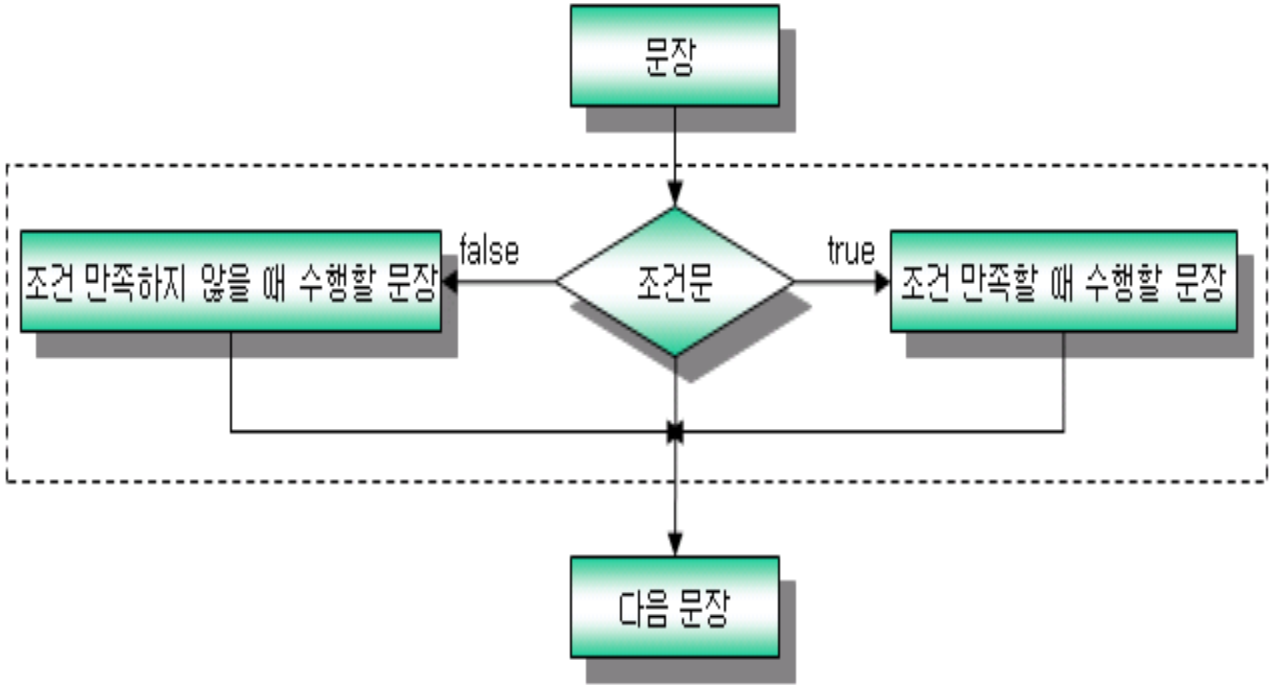
if(조건문){ 조건에 만족할 경우 실행되는 문장; }

if(조건){ 조건에 만족할 경우 실행되는 문장; } else{ 조건에 만족하지 않을 경우 실행되는 문장; }



2] if~else 문

if~else 문은 참일 때와 거짓일 때 각각 다른 문장을 수행하도록 지정



2] 분기문 - 단일if문 예제

```
class IfTest1 {
    public static void main(String[]  arg){
    int a=5;
    if(a<5)   System.out.println("a는  5보다 작다");
    else      System.out.println("a는  5보다 작지 않다.");
     }
}
```

2] if~else~if 문

둘이 아닌 셋 이상에서 하나를 선택해야 할 경우에는 if~else 문을 중첩해서 사용해야 한다. if~else~if 문

```
if(조건1){
조건1에 만족할 때 처리할 문장;
}
else if(조건2){
조건1에 만족하지 않지만 조건2에 만족할 때 처리할 문장;
}
. . .
else if(조건n){
조건1부터 조건n-1에 만족하지 않지만 조건n에 만족할 때 처리할 문장;
}
else{
위에서 언급한 모든 조건에 대해서 만족하지 않을 때 처리할 문장;
}
다음 문장;
```

2]분기문 - 다중if문

```
class IfTest2 {
    public static void main(String[] arg){
        int a=5, b=6;
        int c;
        if(a<5){
            c=a+b;
            System.out.println(c);
        }else{
            c=a-b;
            System.out.println(c);
        }
    }
}
```

2]분기문 - 다중if문1

```
class IfTest2 {
    public static void main(String[] arg){
            int a=5, b=6;
            int c;
            if(a<5){
                    c=a+b;
                    System.out.println(c);
            }else{
                    c=a-b;
                    System.out.println(c);
            }
    }
}
```

2]분기문 - 다중if문2

```
//다중 if else 문
class IfTest3{
	public static void main(String[]  arg) {
		int a;
		a=Integer.parseInt(arg[0]);
		if(a>=95) System.out.println("당신의 점수는 "+a+" 이고 학점은 A++입니다.");
		else if(a>=90) System.out.println("당신의 점수는 "+a+" 이고 학점은 A입니다.");
		else if(a>=85) System.out.println("당신의 점수는 "+a+" 이고 학점은 B++입니다.");
		else if(a>=80) System.out.println("당신의 점수는 "+a+" 이고 학점은 B입니다.");
		else if(a>=75) System.out.println("당신의 점수는 "+a+" 이고 학점은 C++입니다.");
		else if(a>=70) System.out.println("당신의 점수는 "+a+" 이고 학점은 C입니다.");
		else if(a>=65) System.out.println("당신의 점수는 "+a+" 이고 학점은 D++입니다.");
		else if(a>=60) System.out.println("당신의 점수는 "+a+" 이고 학점은 D입니다.");
		Else		System.out.println("당신의 점수는 "+a+" 이고 학점은 F입니다.");
	}
}
```

3] 중첩 if문

if문 안에 또 다른 if문을 중첩해서 넣을 수 있다

- if문의 중첩횟수에는 거의 제한이 없다.

```
if (조건식1) {
      // 조건식1의 연산결과가 true일 때 수행될 문장들을 적는다.
      if (조건식2) {
            // 조건식1과 조건식2가 모두 true일 때 수행될 문장들
      } else {
            // 조건식1이 true이고, 조건식2가 false일 때 수행되는 문장들
      }
} else {
      // 조건식1이 false일 때 수행되는 문장들
}
```

3] 중첩 if문

```
if (score >= 90)    {        // score가 90점 보다 같거나 크면 A학점(grade)
    grade = "A";

    if ( score >= 98) {      // 90점 이상 중에서도 98점 이상은 A+
            grade += "+";    // grade = grade + "+";
    } else if ( score < 94) {
            grade += "-";
    }
} else if  (score >= 80){    // score가 80점 보다 같거나 크면 B학점(grade)
    grade = "B";

    if ( score >= 88) {
        grade += "+";
    } else if ( score < 84) {
        grade += "-";
    }
} else {                     // 나머지는 C학점(grade)
    grade = "C";
}
```

4] switch문 I

- ▶ switch-case문

☞ 형식

```
switch(변수) {
   case 값 1: 실행문 1;
                break;
   case 값 2: 실행문 2;
               break;
   case 값 3: 실행문 3;
                break;
  default: 실행문 4;
               break;
  }
```

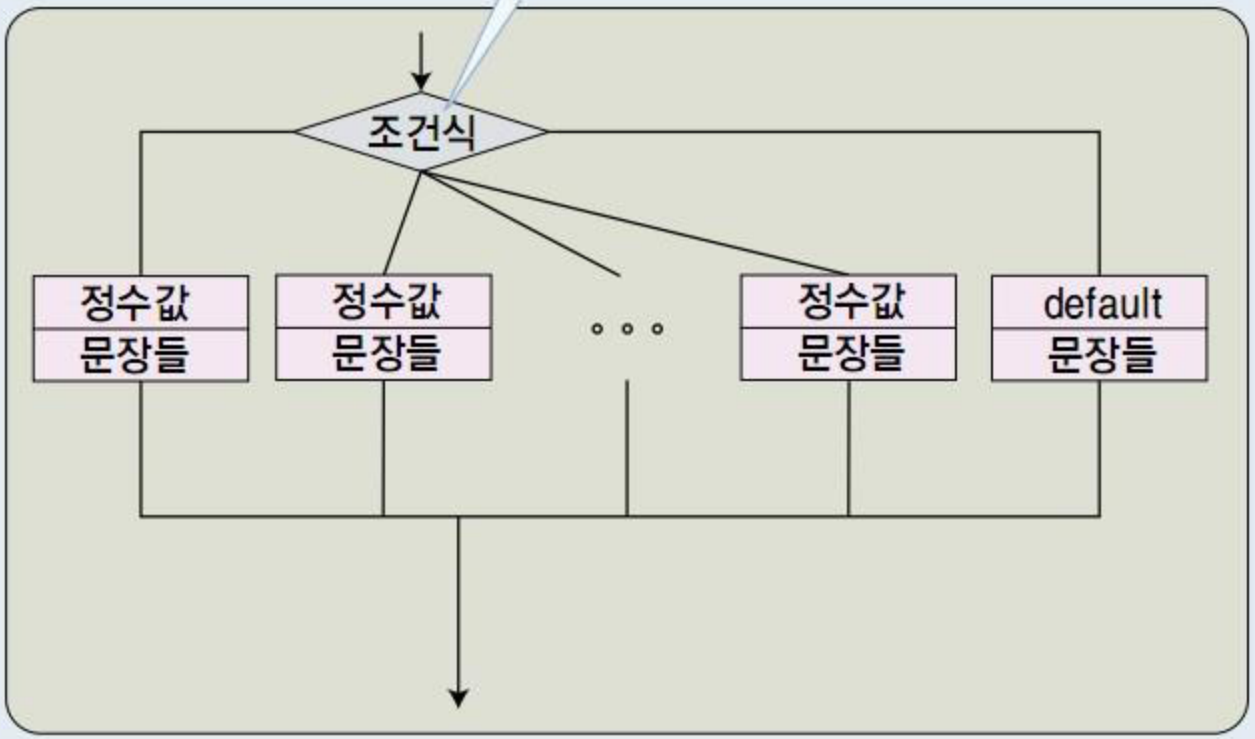
예제)

```
switch(Integer.parseInt(args[0]))
{
case 1:
        System.out.println("1인데요");
        break;
case 2:
        System.out.println("2인데요");
        break;
default:
        System.out.println("값이 넘었네요");
        break;
}
```

4] switch문2

정수식의 결과에 따라 적합한 case절을 수행하고, 없을 경우 default절을 수행한다.

```
switch (정수식)
{
case 정수값1:
        정수식의문장1;
        break;
case 정수값2:
        문장2;
        break;
case 정수값3:
        문장3;
        break;
        ..........
}
```



4] switch문 예시문

```
class SwithTest1{
    public static void main(String[] arg){
        int money=Integer.parseInt(arg[0]);

        switch(money){
            case 500:
            System.out.println("버스를 타시오");
            break;
            case 5000:
            System.out.println("일반택시를 타시오");
            break;
            case 50000:
            System.out.println("모범택시를 타시오");
            break;
            default:
            System.out.println("걸어 가시오");
        }
    }
}
```

4] 중첩 switch문 예시문

- switch문의 중첩횟수에는 거의 제한이 없다.
- switch문 안에 또 다른 switch문을 중첩해서 넣을 수 있다

```
switch(num) {
    case 1:
    case 7:
        System.out.println("SK");
        switch(num) {
            case 1:
                System.out.println("1");
                break;
            case 7:
                System.out.println("7");
                break;
        }
        break;
    case 6:
        System.out.println("KTF");
        break;
    case 9:
        System.out.println("LG");
        break;
    default:
        System.out.println("UNKNOWN");
}
```

4] if문과 switch문의 비교

- if문이 주로 사용되며, 경우의 수가 많은 경우 switch문을 사용할 것을 고려한다. 모든 switch문은 if문으로 변경이 가능하지만, if문은 switch문으로 변경할 수 없는 경우가 많다
- if문 보다 switch문이 더 간결하고 효율적이다

```
if(num==1) {
    System.out.println("SK");
} else if(num==6) {
    System.out.println("KTF");
} else if(num==9) {
    System.out.println("LG");
} else {
    System.out.println("UNKNOWN");
}
```

```
switch(num) {
    case 1:
        System.out.println("SK");
        break;
    case 6:
        System.out.println("KTF");
        break;
    case 9:
        System.out.println("LG");
        break;
    default:
        System.out.println("UNKNOWN");
}
```

5] 반복문 – for, while, do-while

- 문장 또는 문장들을 반복해서 수행할 때 사용
- for문과 while문은 서로 변경가능하다.
- 반복회수가 중요한 경우에 for문을 그 외에는 while문을 사용한다.
- do-while문은 while문의 변형으로 블럭{}이 최소한 한번은 수행될 것을 보장한다
- 조건식과 수행할 블럭{} 또는 문장으로 구성

```
System.out.println(1);
System.out.println(2);
System.out.println(3);
System.out.println(4);
System.out.println(5);
```

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

```
int i=1;

while(i<=5) {
    System.out.println(i);
    i++:
}
```

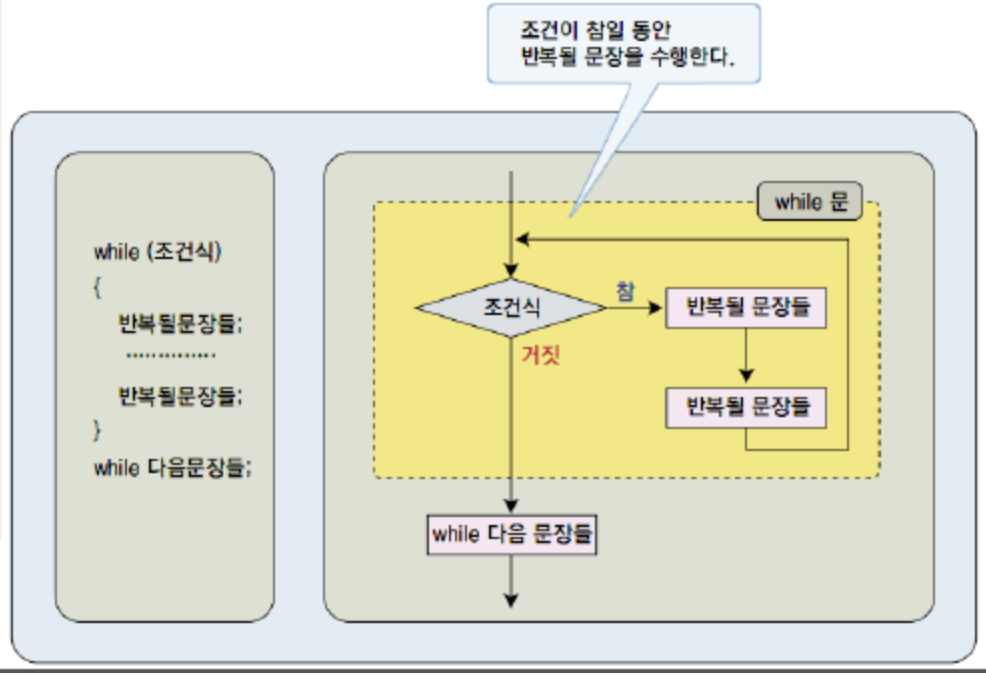
```
int i=0;

do {
    i++;
    System.out.println(i);
} while(i<=5);
```

5] 반복문 – while문

-조건을 잘못 주면 무한반복이 될 수도 있다.
-항상 조건의 값은 boolean자료 형인 true, false이다.
-실행 문이 한 줄일 때에는 { }를 생략해도 된다

- **while문**
  - **특정 조건이 만족하는 동안 지정된 영역을 반복할 수 있는 기능**



예제

```
a=6;
while( a > 5) {
 System.out.println("안녕");
 a--;
}
```

5] 반복문 – while문 예시

```
//1 부터 10까지의 합을 구한다.
class WhileTest1{
    public static void main(String[]  arg){
        int sum=0;
        int i=1;
        while(i<=10){
        sum  += i;
        i++;
    }
    System.out.println(sum);
  }
}
```

```
class WhileTest2  {
    public static void main(String[]  arg){
        int sum=0;
        int I =Integer.parseInt(arg[0]);
        while(i<=10){
            sum  += i;
            System.out.println("i가  "+i+"일때  sum은  "+sum+"이다");
             i++;
        }
    }
}
```

5] 반복문 – 중첩while문

-while문 안에 또 다른 while문을 포함시킬 수 있다.
- while문의 중첩횟수에는 거의 제한이 없다

```
int i=2;
while(i <= 9) {
     int j=1;
     while(j <= 9) {
          System.out.println(i+" * "+j+" = "+i*j);
          j++;
     }
     i++;
}
```

5] 반복문 – do-while문

-while문가 거의 비슷하지만 일단 먼저 한번은 실행 문을 실행하고 비교를 한다.

- 형식

do {
실행문;
} while(조건)

-항상 조건의 값은 boolean자료 형인 true, false이다.

예제

```
a=6;
do {
System.out.println("안녕");
a--;
}while( a > 5)
```

5] 반복문 – do-while문 예시 1

```
import java.io.IOException;
public class doWhileOperator {
	public static void main(String[] args) throws IOException {
	int a;
	do {
		System.out.println("숫자를 입력하시오");
		a = System.in.read() - '0';
		System.in.read();
		System.in.read();
	}while(a%2==1);
	System.out.println("a = " + a + "이고 짝수");
	}
}
```

5] Scanner 테스트

```
import java.util.Scanner;
public class ScannerTest {
 /** * Scanner 테스트 */
        public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       //next
       String nextString = sc.next();
       System.out.println("nextString = " + nextString);
       //nextInt
       int nextIntNum = sc.nextInt();
       System.out.println("nextIntNum = " + nextIntNum);
       }
}
```

5] Math.random() 테스트

- Math클래스에 정의된 난수(亂數) 발생함수
- 0.0과 1.0 사이의 double값을 반환한다.(0.0 <= Math.random() < 1.0)

예) 1~10범위의 임의의 정수를 얻는 식 만들기

1. 각 변에 10을 곱한다.

```
0.0 * 10 <= Math.random() * 10 < 1.0 * 10
    0.0 <= Math.random() * 10 < 10.0
```

2. 각 변을 int형으로 변환한다.

```
(int)0.0 <= (int)(Math.random() * 10) < (int)10.0
       0 <= (int)(Math.random() * 10) < 10
```

3. 각 변에 1을 더한다.

```
0 + 1 <= (int)(Math.random() * 10) + 1 < 10 + 1
    1 <= (int)(Math.random() * 10) + 1 < 11
```

5] 난수 테스트

```
import java.util.Scanner;
public class FlowTest21 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int number = (int)(Math.random()*100)+1;
		int inNumber = 0;
		do{ System.out.println("숫자를 입력하세요...");
			inNumber = sc.nextInt();
			if(inNumber == number){
			System.out.println("맞혔습니다.");
			break;
			} else if( inNumber < number ){
			System.out.println("숫자가 너무 작아요.");
			} else { System.out.println("숫자가 너무 커요."); }
		} while(true);
	}
}
```

5-1] 반복문 – for문

▶ for문

반복횟수를 정해주어서 정해진 횟수만큼 반복을 하는 문입니다.
아주 유용하게 잘 쓰이는 문이다.

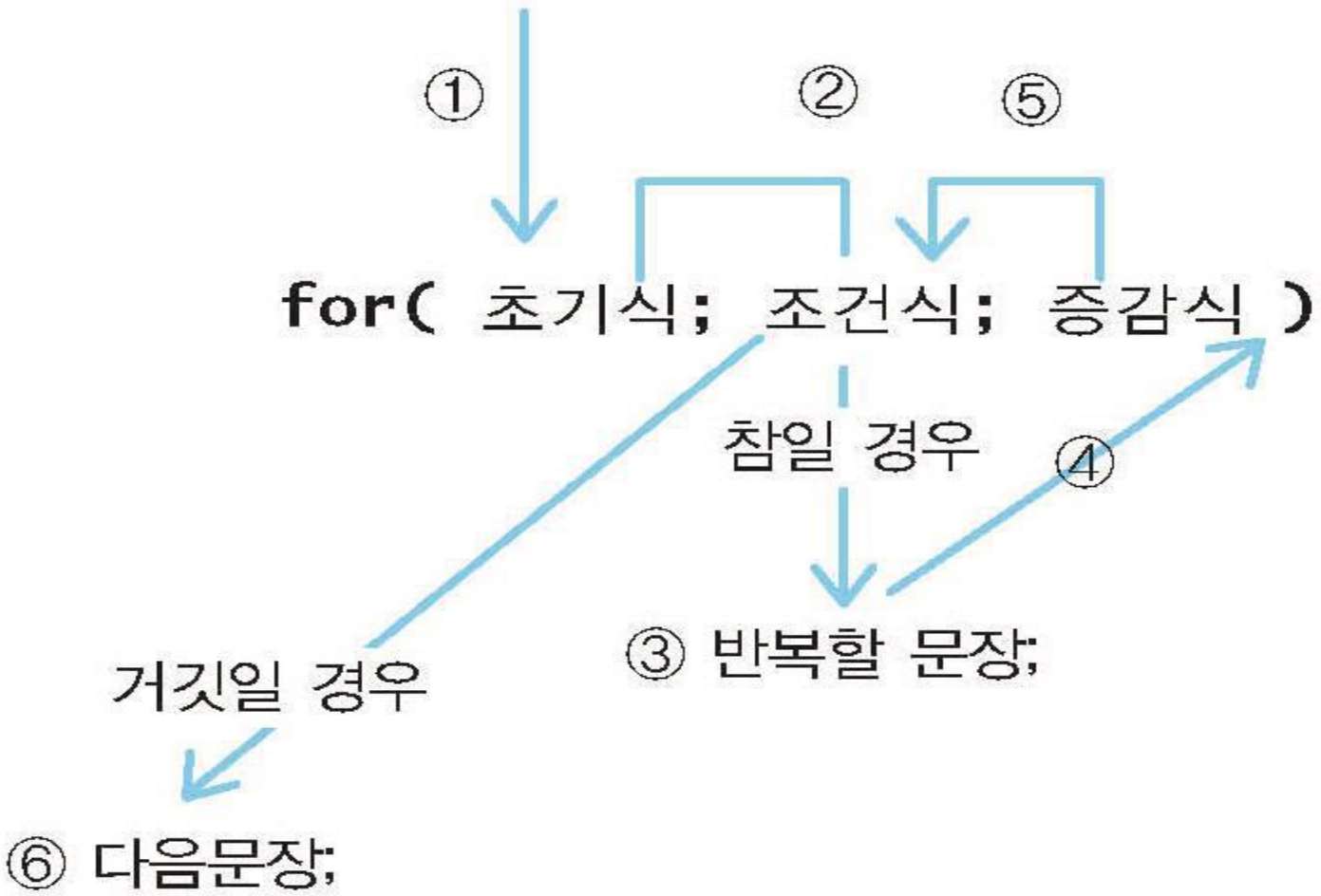
☞ 형식

```
for(반복변수 초기화; 반복변수 조건검사; 반복변수의 값 변화)
{
        실행문 ;
}
```

☞ 예제

```
for(int a=0; a<4; a++)
{
        System.out.println("현재 a의 갯수는 " + a);
}
```

5-1] 반복문 – for문



5-1] 반복문 – for문 기본 다루기

0 3 6 9

```
for(int i=0; i<=10; i+=3)
```

1 6 11 16

```
for(int i=1; i<=20; i+=5)
```

2 4 8 16 32 64

```
for(int i=2; i<=100; i*=2 )
```

5 4 3 2 1

```
for(int i=5; i>=1; i--)
```

5-1] 반복문 – for문 기본 다루기

```
//1부터 100까지의 합을 구하시오
class ForTest1 {
	public static void main(String[] arg){
		int sum=0;
		for(int i=1; i<=100; i++){
			sum +=i;
			System.out.println("i가 "+i+"일때 sum은 "+sum+"이다");
		}
	}
}
```

```
//1부터 100까지의 홀수의 합을 구하시오
class ForTest2 {
	public static void main(String[] arg){
		int sum=0;
		for(int i=1; i<=100; i+=2){
			sum +=i;
			System.out.println("i가 "+i+"일때 sum은 "+sum+"이다");
		}
	}
}
```

5-1] 반복문 – 이중 for문 다루기1

```
class gugudan1{
	public static void main(String[] a){
		int result;
		for(int i=1; i<10; i++){
			System.out.println("");
			for(int j=1; j<10; j++){
				result=i*j;
				System.out.print(" "+j +"*"+i+"="+result);
			}
		}
		System.out.println("");
	}
}
```

5-1] 반복문 – 이중 for문 다루기2

```
class ForTest{
	public static void main(String[] arg){
		for(int i=1; i<=5; i++){
			for(int j=1; j<=i; j++){
				System.out.print("*");
			}
			System.out.println("");
		}
	}
}
```

결과값

```
*
**
***
****
*****
```

```
class ForTest5 {
	public static void main(String[] arg){
		for(int i=1; i<=5; i++){
			for(int j=5;j>=i; j--){
				System.out.print("*");
			}
			System.out.println("");
		}
	}
}
```

결과값

```
*****
****
***
**
*
```

5-1] 반복문 – 이중 for문 다루기3

```
class ForTest6 {
	public static void main(String[] arg){
	for(int i=1; i<=5; i++){
		for(int j=1; j<=5-i; j++)
			System.out.print("  ");
		for(int z=1; z<=i; z++)
			System.out.print("*");
		System.out.println("");
		}
	}
}
```

결과값

```
    *
   **
  ***
 ****
*****
```

```
class ForTest7 {
	public static void main(String[] arg){
	  for(int i=1; i<=5; i++){
		for(int j=1; j<=i; j++) System.out.print(" ");
		for(int z=1; z<=6-i; z++) System.out.print("*");
		 System.out.println("");
		 }
	  }
	 }
}
```

결과값

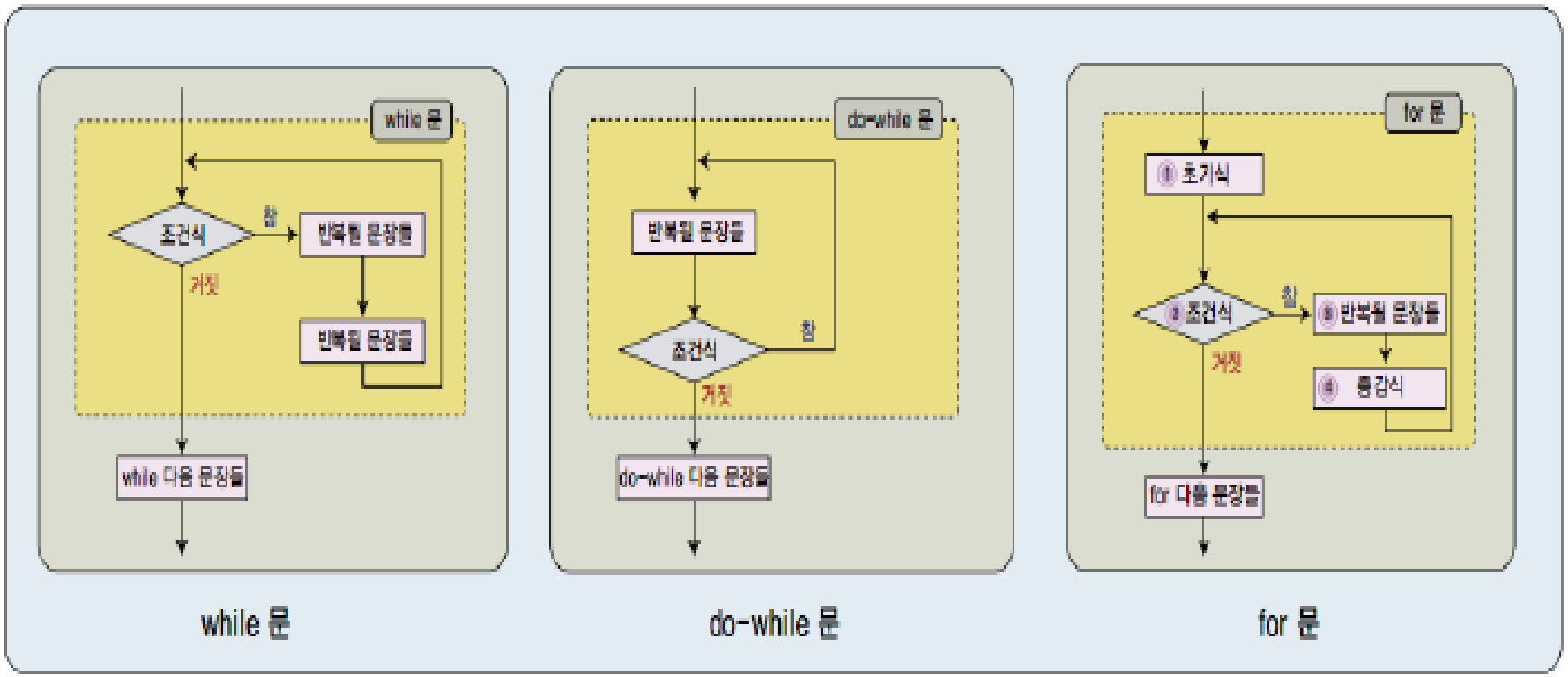
```
*****
****
***
**
*
```

6] 입출력 프로그램 참조

```
public class Exam_01 {
	public static void main(String[] ar) throws java.io.IOException{
		System.err.println("에러 출력 스트림...");
		int x = System.in.read();
		System.out.println("x = " + x);
		System.out.println("기본 출력 스트림...");
	}
}
```

6] 3개 반복문 구조

- 3개의 반복문의 구조



7] 제어문 – break문

◦▶ break문(1)

break문은 반복 문을 빠져 나오게 하는 문이고 반복 문 내의 어떤 곳에 위치할 수도 있다.

☞ 예제

```
Int a=6;
while( a > 5) {
        System.out.println("안녕");
        a--;
        break; // 반복문을 빠져나가게 한다.
}
```

7] 제어문 – break문

```
class BreakTest {
	public static void main(String[] arg){
		int sum=0;
		for(int i=1; i<=100; i++){
			if(i==51)    break;
			sum +=i;
			System.out.println("i가 "+i+"일때 sum은 "+sum+"이다");
		}
	}
}
```

7] 제어문 – continue문

▶ continue문

반복 문을 빠져 나가지 않고 반복문의 조건부로 실행위치를 옮긴다.
continue이후에 실행 문들은 실행이 되지 않는다.

☞ 예제

```
int a=1;
for(int x=1; x<=5;x++){
        System.out.println("나는 반복한다.");
       a++;
       if(a>3) continue;
       System.out.println("나두 반복되고 싶어");
}
```

7] 제어문 – label문

```
class LabelTest {
	public static void main(String[] arg){
		outer:
		for(int i=0;i<10;i++){
			for(int j=0;j<10;j++){
				if(j==3)  break outer;
				System.out.println("j="+j);
				}
				System.out.println("i="+i);
		}
	}
}
```