

조건문

2

자바 기본 프로그래밍

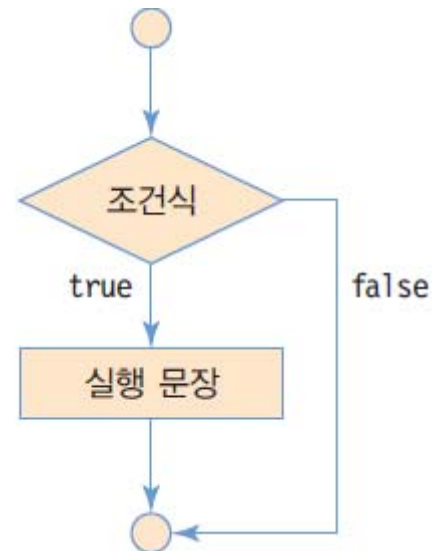
조건문 - if문

2

- 단순 if 문
 - ▣ if 다음의 괄호 안에는 조건식(논리형 변수나 논리 연산)
 - ▣ 조건식의 값
 - true인 경우, if문을 벗어나 다음 문장이 실행된다.
 - false의 경우에는 if 다음의 문장이 실행되지 않고 if 문을 빠져 나온다.
 - ▣ 실행문장이 단일 문장인 경우 둘러싸는 { } 생략 가능

if(조건식) {
...실행 문장...
}

if 키워드



예제 2-10 : if문 사용하기

3

시험 점수가 80점이 이상이면 합격 판별을 하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class SuccessOrFail {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.print("점수를 입력하시오: ");
        int score = in.nextInt();
        if (score >= 80)
            System.out.println("축하합니다! 합격입니다.");
    }
}
```

점수를 입력하시오: 95
축하합니다! 합격입니다.

조건문 – if-else

4

□ if-else 문

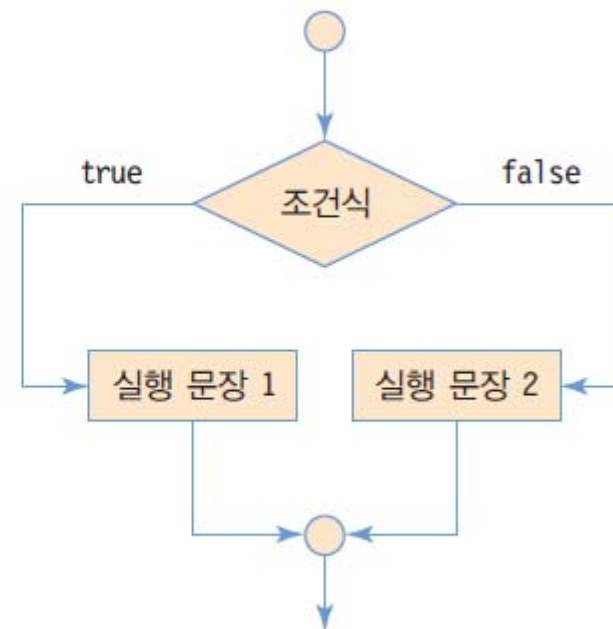
- 조건식이 true면 실행문장1 실행 후 if-else문을 벗어남
- false인 경우에 실행문장2 실행후, if-else문을 벗어남

>

```
if(조건식) {  
    ...실행 문장 1...  
}  
else {  
    ...실행 문장 2...  
}
```

if 키워드

else 키워드



예제 2-11 : if-else 사용하기

5

입력된 수가 3의 배수인지 판별하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class MultipleOfThree {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.print("수를 입력하시오: ");
        int number = in.nextInt();

        if (number % 3 == 0)
            System.out.println("3의 배수입니다.");
        else
            System.out.println("3의 배수가 아닙니다.");
    }
}
```

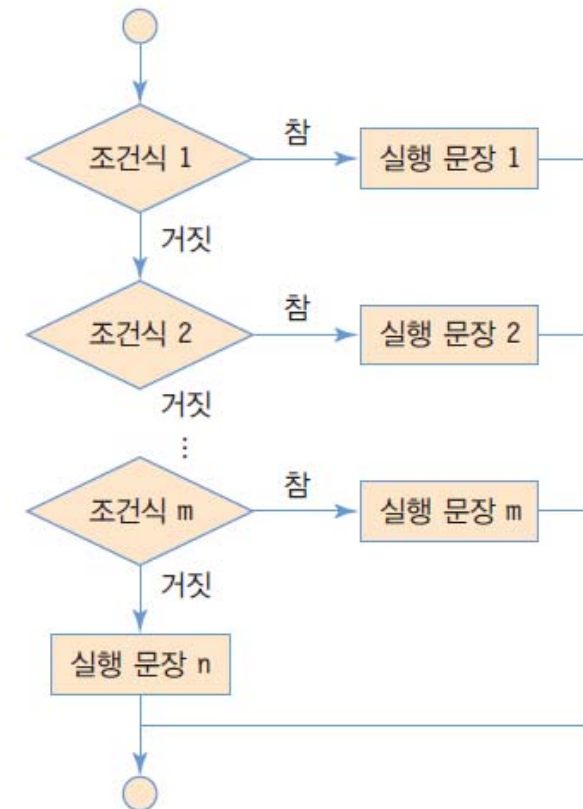
수를 입력하시오: 129
3의 배수입니다.

조건문 - 중첩 if

6

□ 중첩 if문

- ▣ 조건문이 너무 많은 경우, switch 문 사용 권장



예제 2-12 : 학점 매기기

7

if-else문을 이용하여 키보드 입력된 성적에 대해 학점을 부여하는 프로그램을 작성해보자.

```
import java.util.Scanner;

public class Grading {
    public static void main (String[] args) {
        char grade;
        Scanner a = new Scanner(System.in);
        while (a.hasNext()) {
            int score = a.nextInt();
            if(score >= 90) // score가 90 이상인 경우
                grade = 'A';
            else if(score >= 80) // score가 80 이상이면서 90 미만인 경우
                grade = 'B';
            else if(score >= 70) // score가 70 이상이면서 80 미만인 경우
                grade = 'C';
            else if(score >= 60) // score가 60 이상이면서 70 미만인 경우
                grade = 'D';
            else // score가 60 미만인 경우
                grade = 'F';
            System.out.println("학점은 "+grade+"입니다");
        }
    }
}
```

키가 입력될 때까지 기다리며, 입력된 키가 있는 경우 true 리턴. 라인의 첫 문자로 ctrl-z 키가 입력되면 false 리턴

80
학점은 B입니다
90
학점은 A입니다
76
학점은 C입니다

Tip: if문과 조건 연산자 ?:

8

- 조건 연산자 ?::는 if-else로 바꿀 수 있음

```
i = a>b?a-b:b-a;
```

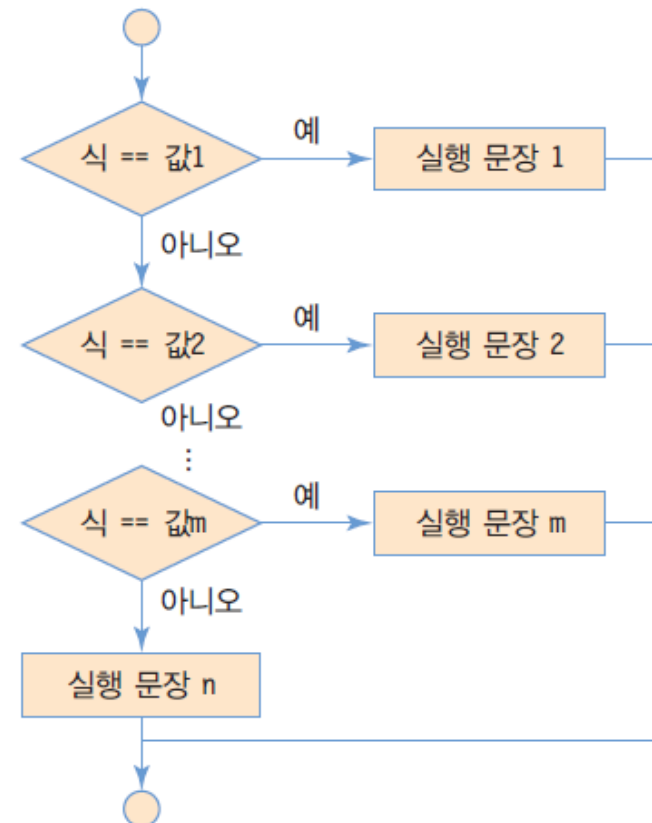
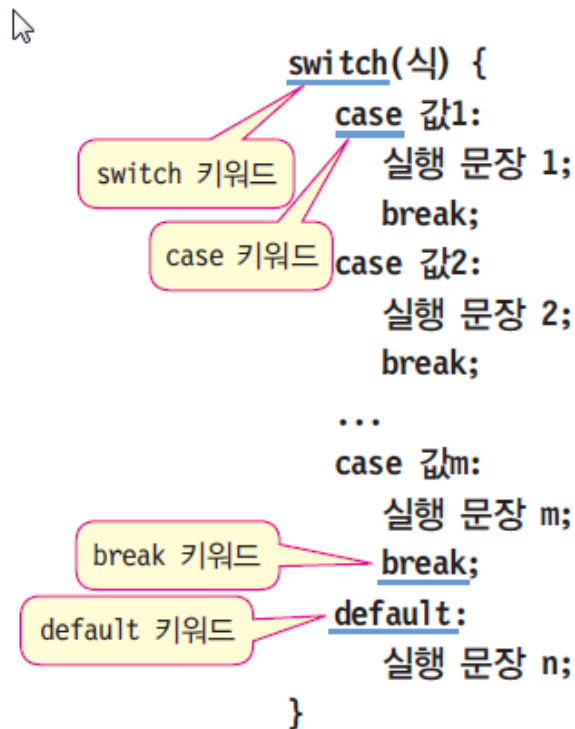


```
if (a>b)
    i = a - b;
else
    i = b - a;
```


switch문

9

- switch문은 식과 case 문의 값과 비교
 - ▣ case의 비교 값과 일치하면 해당 case의 실행문장 수행
 - break를 만나면 switch문을 벗어남
 - ▣ case의 비교 값과 일치하는 것이 없으면 default 문 실행
- default문은 생략 가능



switch문에서 벗어나기

10

- switch문 내의 break문
 - ▣ break문 만나면 switch문 벗어남
 - ▣ case 문에 break문이 없다면, 다음 case문으로 실행 계속
 - 언젠가 break를 만날 때까지 계속 내려 가면서 실행

```
char grade='A';
switch (grade) {
    case 'A':
        System.out.println("90 ~ 100점입니다.");
        break;
    case 'B':
        System.out.println("80 ~ 89점입니다.");
        break;
    case 'C':
        System.out.println("70 ~ 79점입니다.");
        break;
}
```

90 ~ 100점입니다.
80 ~ 89점입니다.

예제 2-13 : switch문의 break 사용하기

11

학점이 A, B 인 학생에게는 "참 잘하였습니다.", 학점이 C, D인 학생에게는 "좀 더 노력하세요.", 학점이 F인 학생에게는 "다음 학기에 다시 수강하세요."를 출력하는 프로그램을 switch문의 break를 잘 활용하여 작성하여라.

```
public class GradeSwitch {
    public static void main(String[] args) {
        char grade='C';
        switch (grade) {
            case 'A':
            case 'B':
                System.out.println("참 잘하였습니다.");
                break;
            case 'C':
            case 'D':
                System.out.println("좀 더 노력하세요.");
                break;
            case 'F':
                System.out.println("다음 학기에 다시 수강하세요.");
                break;
            default:
                System.out.println("잘못된 학점입니다.");
        }
    }
}
```

좀 더 노력하세요.

case 문의 값

12

- case 문의 값의 특징
 - ▣ switch 문은 식의 결과 값을 case 문과 비교
 - ▣ 사용 가능한 case문의 값
 - 문자, 정수, 문자열 리터럴(JDK 1.7부터)만 허용
 - 실수 리터럴은 허용되지 않음

```
int c = 25;
switch(c%2) {
    case 1 :           // 정수 리터럴
        ...;
        break;
    case 2 :           // 정수 리터럴
        ...;
        break;
}

String s = "예";
switch(s) {
    case "예" :         // 문자열 리터럴. JDK1.7부터 적용
        ...;
        break;
    case "아니요" :     // 문자열 리터럴. JDK1.7부터 적용
        ...;
        break;
}
```

정상적인
case 문

```
char grade='C';
switch (grade) {
    case 'A' :          // 문자 리터럴
        ...;
        break;
    case 'B' :          // 문자 리터럴
        ...;
        break;
}
```

정상적인
case 문

```
switch(a) {
    case a :            // 오류. 변수 사용 안됨
    case a > 3 :         // 오류. 수식 안됨
    case a == 1 :       // 오류. 수식 안됨
}
```

잘못된
case 문

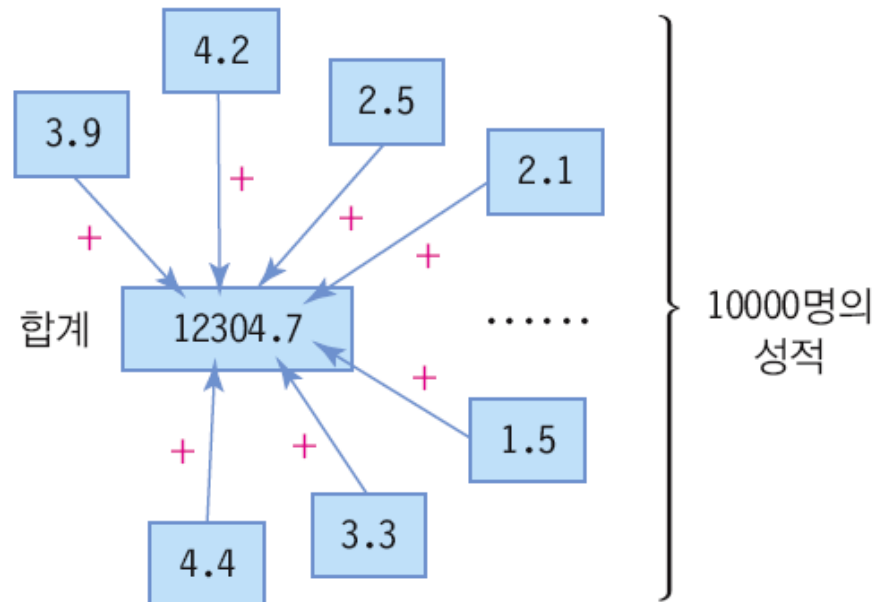
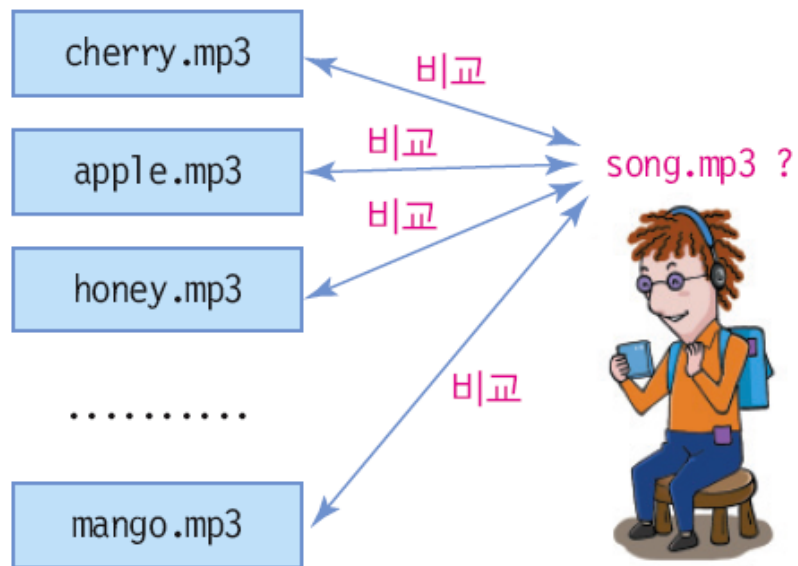
반복문

2

반복문의 특징

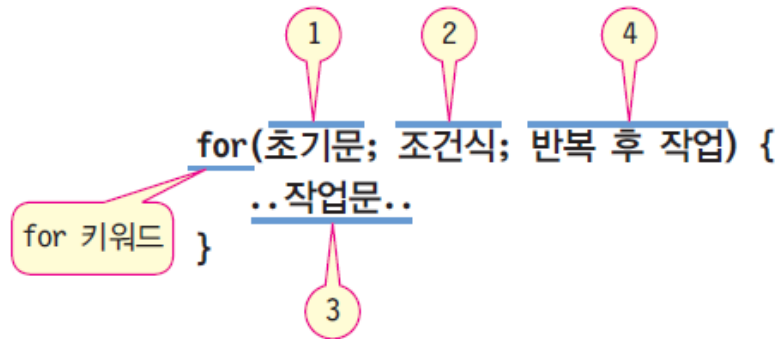
14

- 자바 반복문의 종류
 - ▣ for 문
 - ▣ while 문
 - ▣ do while 문

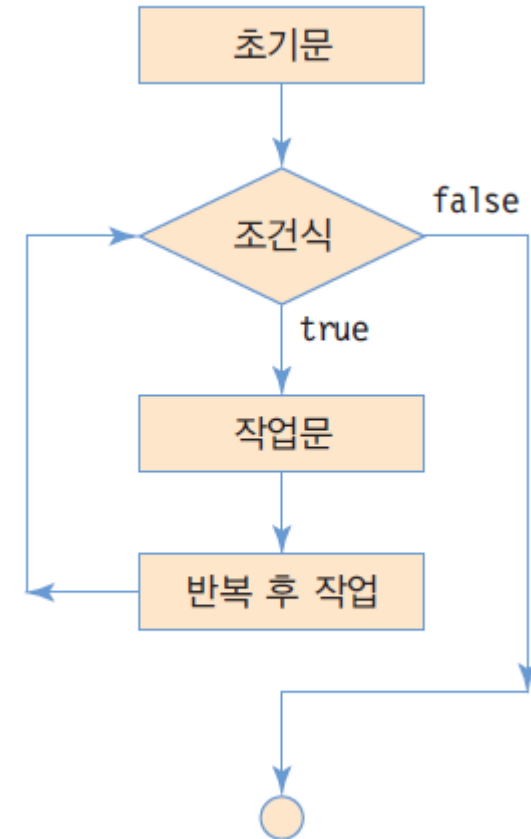


for 문의 구성

15

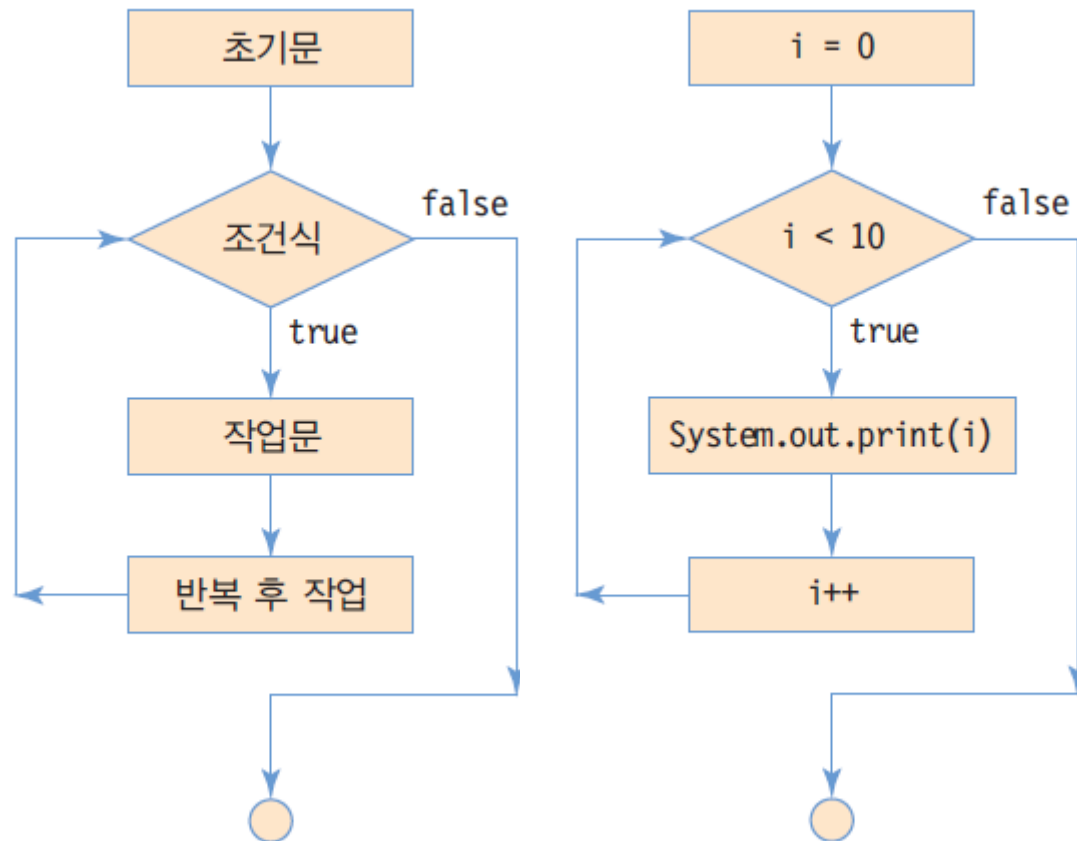


- ①
 - for 문이 실행한 후 오직 한번만 실행되는 초기화 작업
 - 콤마(',')로 구분하여 여러 문장 나열 가능
 - 초기화할 일 없으면 비어둘 수 있음
- ②
 - 논리형 변수나 논리 연산만 가능
 - 반복 조건이 true이면 반복 계속, false이면 반복 종료
 - 반복 조건이 true 상수인 경우, 무한 반복
 - 반복 조건이 비어 있으면 true로 간주
- ④
 - 반복 작업 문장들의 실행 후 처리 작업
 - 콤마(',')로 구분하여 여러 문장 나열 가능



for문의 실행 과정을 나타내는 순서도

16



```
for(i=0; i<10; i++) {  
    System.out.print(i);  
}
```


for문의 예시

17

- 0에서 9까지 정수 출력

```
int i;  
for(i = 0; i < 10; i++) {  
    System.out.print(i);  
}
```

```
int i;  
for(i = 0; i < 10; i++)  
    System.out.print(i);
```

- 반복문에 변수 선언 가능

```
for(int i = 0; i < 10; i++) // 변수 i는 for문을 벗어나서 사용할 수 없음  
    System.out.print(i);
```

- 0에서 100까지의 합 구하기

```
int sum = 0;  
for(int i = 0; i <= 100; i++)  
    sum += i;
```

```
int sum;  
for(int i = 0, sum=0; i <= 100; i++)  
    sum += i;
```

```
int sum = 0;  
for(int i = 100; i >= 0; i--)  
    sum += i;
```

for문의 특이한 형태

18

```
for(초기작업; true; 반복후작업) { // 반복 조건이 true이면 무한 반복  
.....  
}
```

```
for(초기작업; ; 반복후작업) { // 반복조건이 비어 있으면 true로 간주, 무한 반복  
.....  
}
```

```
// 초기 작업과 반복후작업은 ';'로 분리하여 여러 문장 나열 가능  
for(i=0; i<10; i++, System.out.println(i)) {  
.....  
}
```

```
// for문 내에 변수 선언  
for(int i=0; i<10; i++) { // 변수 i는 for문 내에서만 사용 가능  
.....  
}
```

예제 3-1 : 1부터 10까지 숫자의 합을 출력

19

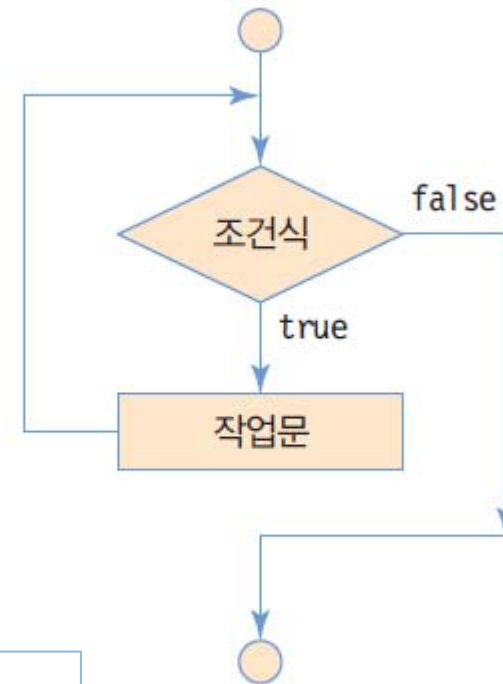
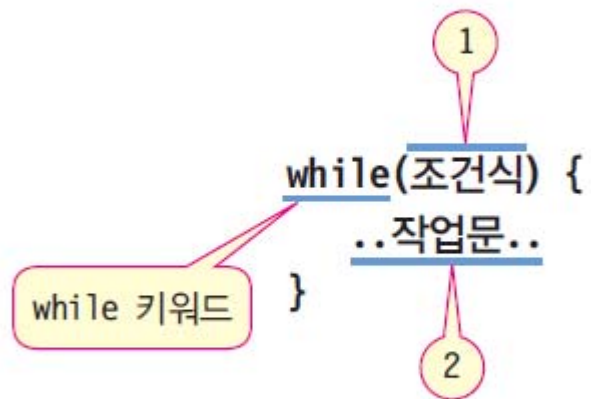
for문을 이용하여 1부터 10까지 덧셈을 표시하고 합을 구하시오.

```
public class ForSample {  
    public static void main (String[] args) {  
        int i, j;  
  
        for (j=0,i=1; i <= 10; i++) {  
            j += i;  
            System.out.print(i);  
            if(i==10) {  
                System.out.print("=");  
                System.out.print(j);  
            }  
            else  
                System.out.print("+");  
        }  
    }  
}
```

1+2+3+4+5+6+7+8+9+10=55

while 문의 구성

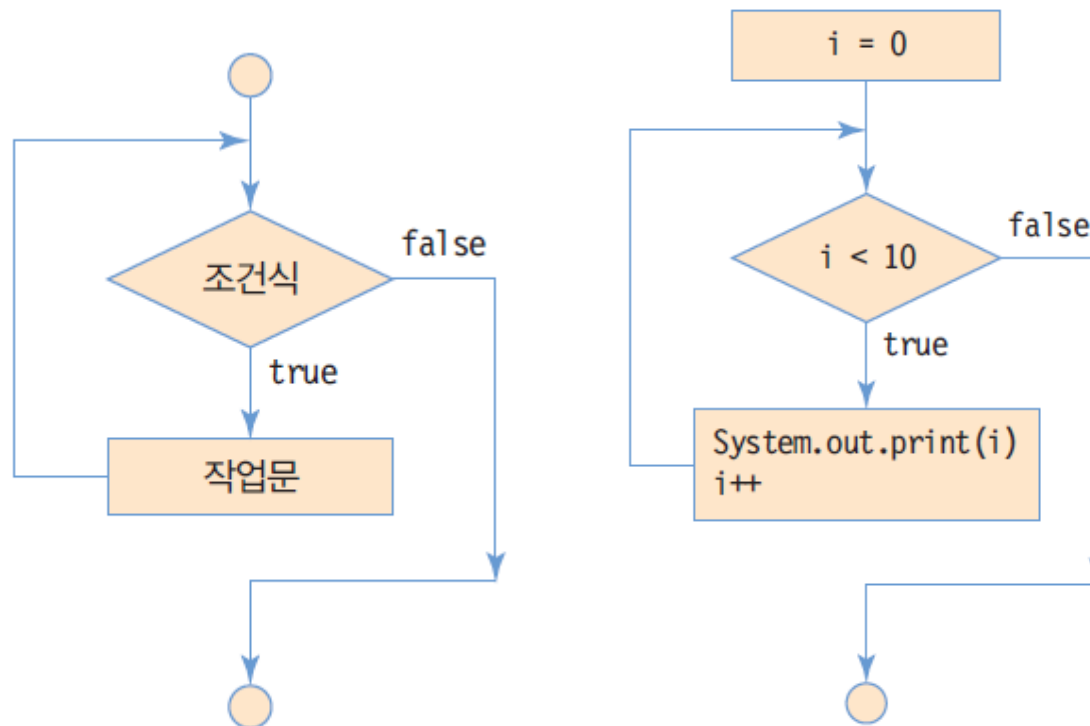
20



- 반복 조건이 true이면 반복, false이면 반복 종료
- 반복 조건이 없으면 컴파일 오류
- 처음부터 반복조건을 통과한 후 작업문 수행

while문의 실행 과정을 나타내는 순서도

21



```
i = 0;
while(i<10) {
    System.out.print(i);
    i++;
}
```

예제 3-2 : 입력된 수의 평균 구하기

22

while문을 이용하여 키보드에서 숫자를 입력 받아 입력 받은 모든 수의 평균을 출력하는 프로그램을 작성해보자.
0이 입력되면 입력이 종료되고 평균을 구하여 출력한다.

```
import java.util.Scanner;
public class WhileSample {
    public static void main (String[] args) {
        Scanner rd = new Scanner(System.in);
        int n = 0;
        double sum = 0;
        int i=0;
        while ((i = rd.nextInt()) != 0) {
            sum += i;
            n++;
        }
        System.out.println("입력된 수의 개수는 " + n + "개이며 평균은 " + sum / n + "입니다.");
    }
}
```

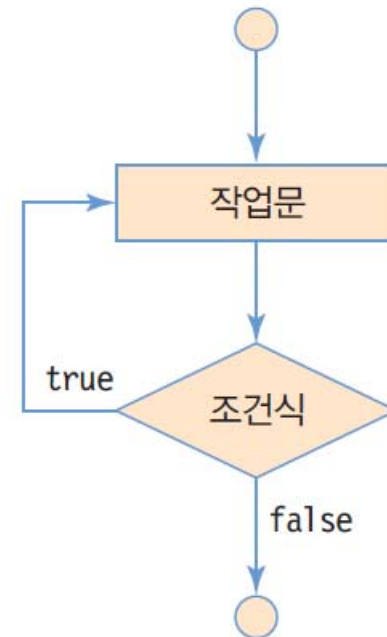
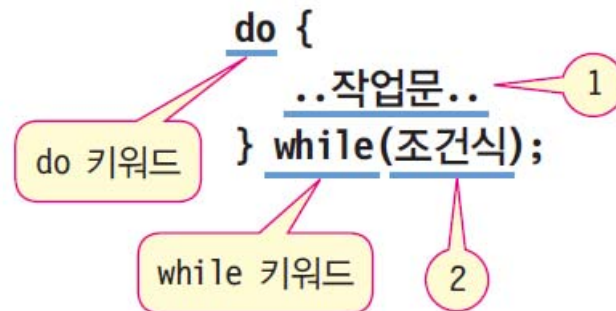
10
20
30
40
0

마지막 입력을 뜻함

입력된 수의 개수는 4개이며 평균은 25.0입니다.

do-while 문의 구성

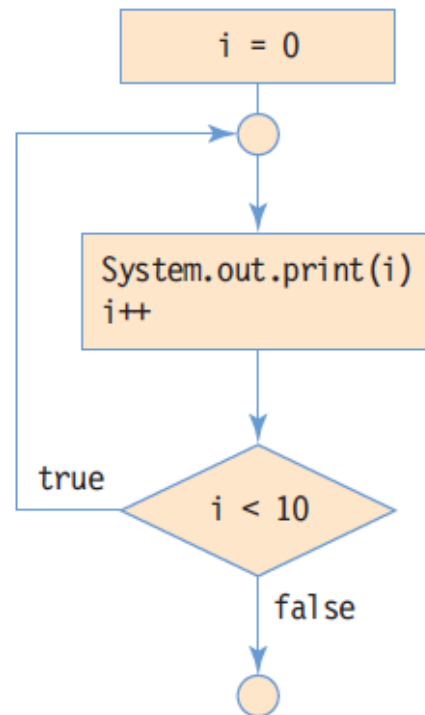
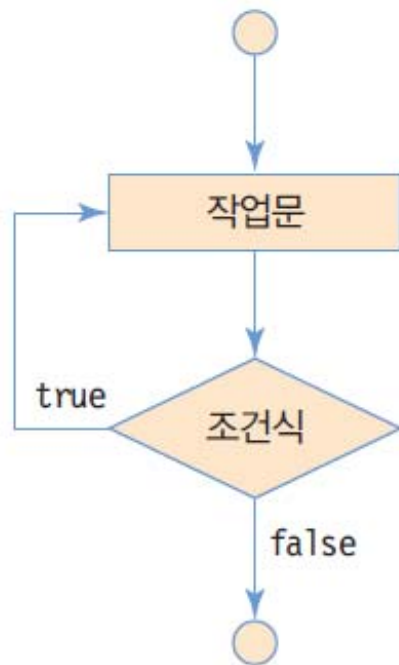
23



- ① • 무조건 최소 한번은 실행
- ② • 반복 조건이 true이면 반복, false이면 반복 종료
• 반복 조건이 없으며 컴파일 오류

do-while문의 실행 과정을 나타내는 순서도

24



```
i = 0;  
do {  
    System.out.print(i);  
    i++;  
} while(i<10);
```


예제 3-3 : a-z까지 출력

25

do-while문을 이용하여 'a'부터 'z'까지 출력하는 프로그램을 작성하시오.

```
public class DoWhileSample {  
    public static void main (String[] args) {  
        char a = 'a';  
  
        do {  
            System.out.print(a);  
            a = (char) (a + 1);  
        } while (a <= 'z');  
    }  
}
```

abcdefghijklmnopqrstuvwxyz

중첩 반복

26

- 중첩 반복
 - ▣ 반복문이 다른 반복문을 내포하는 구조
 - ▣ 이론적으로는 몇 번이고 중첩 반복 가능
 - ▣ 너무 많은 중첩 반복은 프로그램 구조를 복잡하게 하므로 2중 또는 3중 반복이 적당

```
for(i=0; i<100; i++) { // 100 개의 학교 성적을 모두 더한다.  
    ....  
    for(j=0; j<10000; j++) { // 10000 명의 학생 성적을 모두 더한다.  
        ....  
        ....  
    }  
    ....  
}
```

10000명의 학생이 있는 100개 대학의 모든 학생 성적의 합을 구할 때,
for 문을 이용한 이중 중첩 구조

예제 3-4 : 구구단

27

2중 중첩된 for문을 사용하여 구구단을 출력하는 프로그램을 작성하시오.
한 줄에 한 단씩 출력한다.

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

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

continue문

28

□ continue 문

- 반복문을 빠져 나가지 않으면서
- 반복문 실행 도중 다음 반복을 진행

```
for (초기문; 조건식; 반복후작업) {  
    .....  
    continue;  
    .....  
}
```

분기

```
while (조건식) {  
    .....  
    continue;  
    .....  
}
```

조건식으로
분기

```
do {  
    .....  
    continue;  
    .....  
} while (조건식);
```


조건식으로
분기

예제 3-5 : 1부터 100까지 짝수의 합

29

for와 continue문을 사용하여 1부터 100까지 짝수의 합을 구해보자.

```
public class ContinueExample {  
    public static void main (String[] args) {  
        int sum = 0;  
        for (int i = 1; i <= 100; i++) {  
            if (i%2 == 1)  
                continue;  
            else  
                sum += i;  
        }  
        System.out.println("1부터 100까지 짝수의 합은 " + sum);  
    }  
}
```



1부터 100까지 짝수의 합은 2550

break문

30

□ break 문

- ▣ 반복문 하나를 완전히 빠져 나갈 때 사용
- ▣ break문은 하나의 반복문만 벗어남
 - 중첩 반복의 경우 안쪽 반복문의 break 문이 실행되면 안쪽 반복문만 벗어남

예제 3-6 : 입력된 숫자 개수 세기

31

while문과 break문을 사용하여 -1이 입력될 때까지 입력된 숫자의 개수를 출력하시오.

```
import java.util.Scanner;
public class BreakExample {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);
        int num = 0;

        while (true) {
            if (in.nextInt() == -1)
                break;
            num++;
        }
        System.out.println("입력된 숫자 개수는 " + num);
    }
}
```

10
8
9
5
-1
입력된 숫자 개수는 4

마지막 입력을 뜻함

Tip: 라벨로 분기

32

□ 라벨로 분기하는 경우

▣ **continue** 라벨;

- 특정 라벨의 다음 반복으로 분기
- 중첩 반복(nested loop)에서 바깥의 반복문으로 빠져 나갈 때 주로 사용

▣ **break** 라벨;

- 라벨이 붙은 반복문을 벗어남.
- 중첩 반복문을 한 번에 벗어날 때 주로 사용

```
LABEL:
for (초기 작업; 반복 조건;반복 후 작업) {
    for (초기 작업; 반복 조건;반복 후 작업) {
        .....
        continue LABEL;
        .....
    }
}
```

```
LABEL:
for (초기 작업; 반복 조건;반복 후 작업) {
    for (초기 작업; 반복 조건;반복 후 작업) {
        .....
        break LABEL;
        .....
    }
}
.....
```