제3장 조건분기와 반복

조건분기와 반복

우리는 매일 조건에 대한 행동의 분기나 반복을 하고 있습니다.

예를 들어 “만약 일기예보가 비라면 우산을 가지고 간다" 라는 분기나,

“정답일 때 까지 몇 번이라도 문제를 푼다" 라는 반복입니다.

프로그램도 이것과 같은 조건분기와 반복을 수행하며 처리를 진행해 나갑니다.

이 장에서는 조건분기나 반복을 실현하기 위한 Java 의 문법을 배웁니다.

프로그램의 흐름

코드를 실행시키는 순서를 **제어구조** 라고 한다.

대표적인 것으로는 **“순차", “분기", “반복"** 의 3가지가 있다

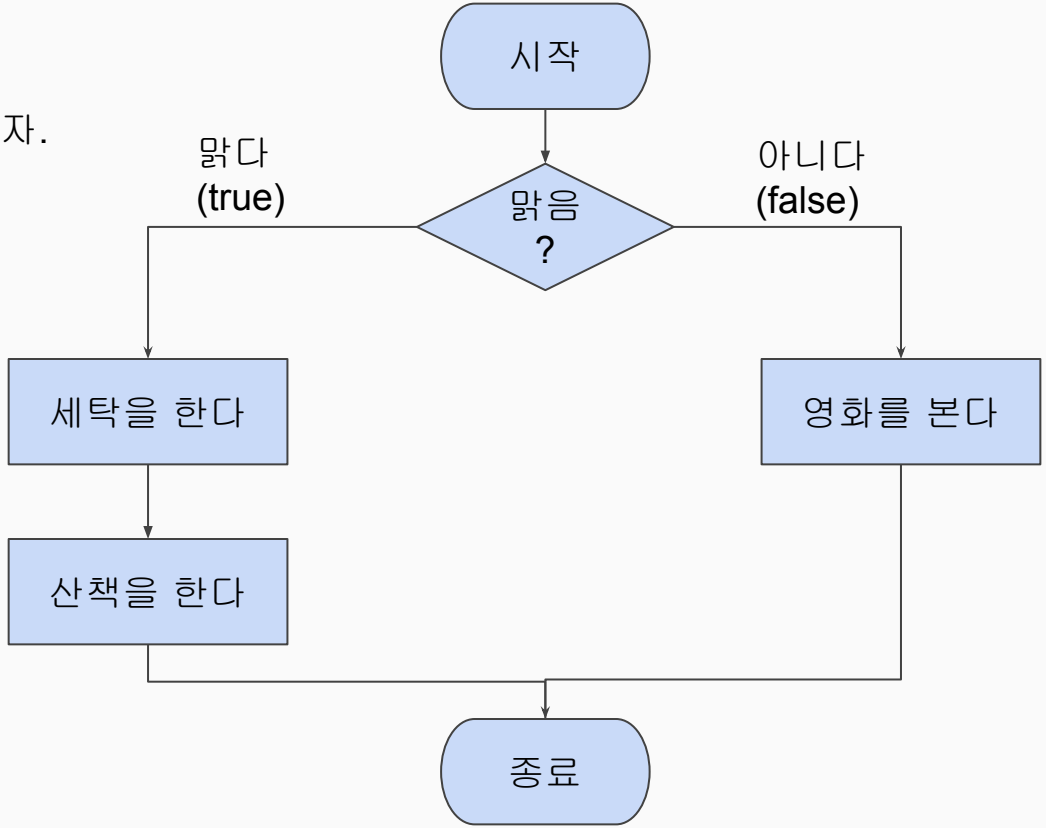
2장까지에 사용했던 “위에서 부터 순차적으로 한줄씩" 실행 것이 “순차" 이다

아무리 복잡한 프로그램이라도, 순차, 분기, 반복의 3가지 제어구조를 조합하면 만들 수 있다

분기를 체험하자

만약 내일이 맑으면 세탁하고 산책을 하자.

하지만, 내일 비가 온다면, 집에서 영화를 보자.



리스트 3-1 날씨에 의한 행동 변화를 Java로 표현

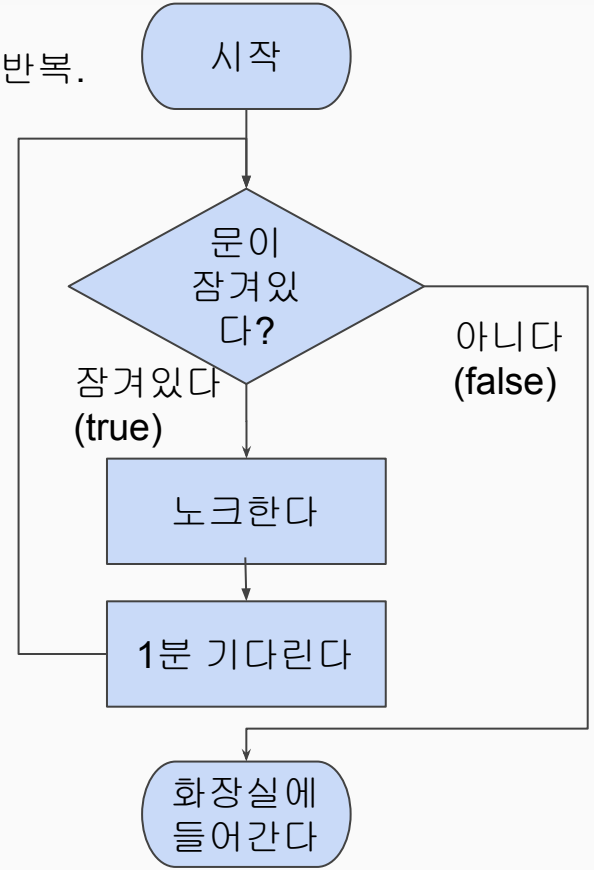

```java
public class Main {
    public static void main(String[] args) {
        boolean clear = true;
        if (clear == true) {
            System.out.println("세탁을 합니다");
            System.out.println("산책을 합니다");
        } else {
            System.out.println("영화를 봅니다");
        }
    }
}
```

분기 정리

- if 라는 명령어를 사용하면 “분기" 를 할 수 있습니다
- if 의 뒤에는 () 안에 “맑은가" 에 대해 “분기조건"을 씁니다
- 변수 clear 가 true 인지 아닌지를 체크를 할 때에는 “==” 을 사용
- 분기조건이 성립한다면, () 의 직후에 있는 블록 (“{“ 와 “}” 로 감싸여져 있는 부분) 의 안쪽 코드가 실행된다
- 분기조건이 성립하지 않는다면, else 의 뒤에 있는 블록의 안쪽 코드가 실행된다

반복을 체험하자

만약 화장실에 누군가 있다면 “문을 노크하고 1분 기다림" 을 반복.



리스트 3-2 화장실이 비는 것을 기다리는 반복 처리


```
public class Main {
    public static void main(String[] args) {
        boolean isDoorClose = true;
        while (isDoorClose == true) {
            System.out.println("노크한다");
            System.out.println("1분 기다린다");
        }
    }
}
```

반복문 정리

- while 을 사용하면 “반복" 제어를 행할 수 있다
- while 의 뒤에는 () 안에 “문이 닫혔는가" 등의 “반복하는 조건"을 씁니다
- 반복 조건이 성립한다면 몇번이라도 블록 안의 코드가 반복 실행 된다

블록의 작성 방법

블록이란 **복수의 문장을 하나로 묶는 역할을 하기 위한 것**

블록 작성에는 2가지 룰이 있음

룰 1 : 중괄호의 생략
내용이 1행 밖에 없을 때는 중괄호를 생략해도 됨

룰 2 : 블록 내에 선언한 변수의 수명
블록 안에서 선언한 변수는 그 블록이 종료와 동시에 소멸한다

리스트 3-3 중괄호의 생략


```
public class Main {
    public static void main(String[] args) {
        boolean clear = true;
        if (clear == true) {
            System.out.println("세탁을 합니다");
            System.out.println("산책을 합니다");
        } else
            System.out.println("영화를 봅니다");
    }
}
```

조건식 작성 방법

```
if (clear == true) {
```

```
if (age > 18) {
```

조건식 안에 사용 되는 ==, > 등의 기호를 **관계연산자** 라고 한다

관계연산자의 종류와 의미

연산자	의미
==	좌변과 우변이 같다
!=	좌변과 우변이 다르다
>	좌변이 우변보다 크다
<	좌변이 우변보다 작다
>=	좌변이 우변보다 크거나 같다
<=	좌변이 우변보다 작거나 같다

- sw != false
- deg - 273.15 < 0
- initial == '오'

age > 18

이 관계연산자는 계산되어 무엇으로 변하는가?

if 문과 while 문 정리

- if문은 “조건식의 평가 결과가 true 이면 제1블록을, false 이면 else 이후의 블록을 실행
  - 조건식은 반드시 true 또는 false 가 나오는 식이어야 함
- while 문은 “조건식의 평가 결과가 true이면, 블록을 반복 실행"

문자열의 비교

Java에서 String 형의 비교에는 **특수한 작성 방법** 이 있다.

예를 들어 “변수 **s** 의 내용이 “"저녁" 이라면 …” 을 코드로 작성 하면

```
if (s == "저녁") {
```

맞는 것 같지만, **Java 에서는 문자열의 비교는 “==”를 사용하면 안 됨**.

이유에 대해서는 뒤에 따로 설명 하겠습니다.

일단, **문자열의 비교를 할 때에는 반드시 다음과 같은 방법을 사용** 해야 합니다.

```
if (s.equals("저녁")) {
```

논리연산자를 사용한 복수의 조건식

"나이가 18세 이상이고, 성별이 남성" 과 같은 2개 이상의 조건의 조합의 조건식을 사용하고 싶을 때에는 **논리연산자**를 사용합니다.

연산자	의미
&&	그리고 (모두 참이면 true)
||	또는 (둘 중 하나가 참이면 true)

```
if (age >= 18 && gender == 1) {
```

```
if (name.equals("준석") || married == true) {
```

Mission : 나이가 18세 이상이면서 gender 가 1 이거나 나이가 16세 이상이면서 gender 가 0

부정연산자

만약 ~이 아니면 을 조건식에 표현하고 싶으면 식의 앞에 **부정연산자 !** 를 붙인다

```
if (!(age == 10)) {
```

수학의 표현과 Java 조건식의 표현

수학 :

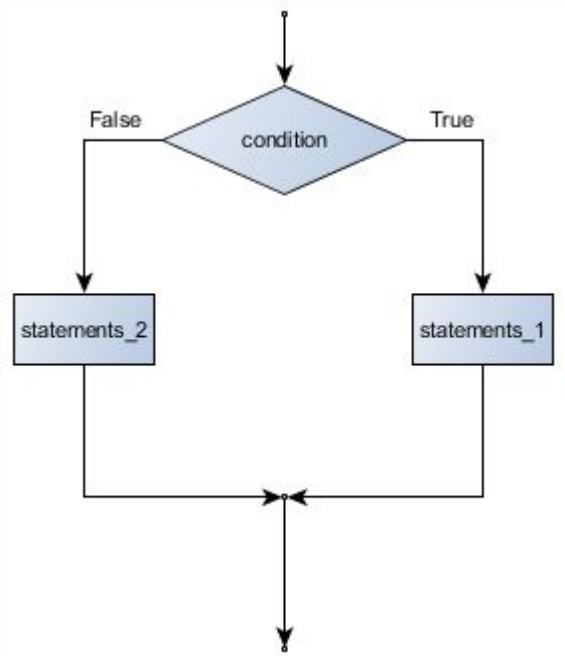
x 는 10보다 크고, 20보다 작다 => 10 < x < 20

Mission : Java 에서는 어떻게 써야 하나요?

분기문의 종류

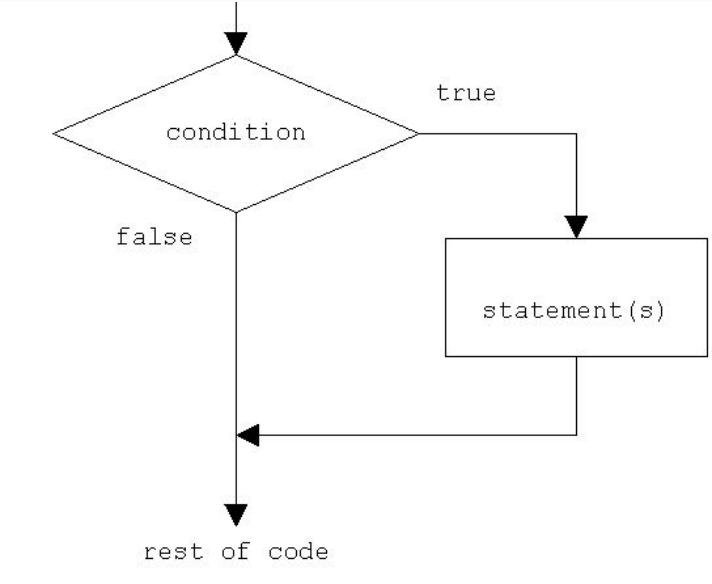
1. if - else 문
2. if 만 있는 문장
3. if - else if - else (false 일 때, 또 다시 조건으로 분기)

if - else 문 (기본형)



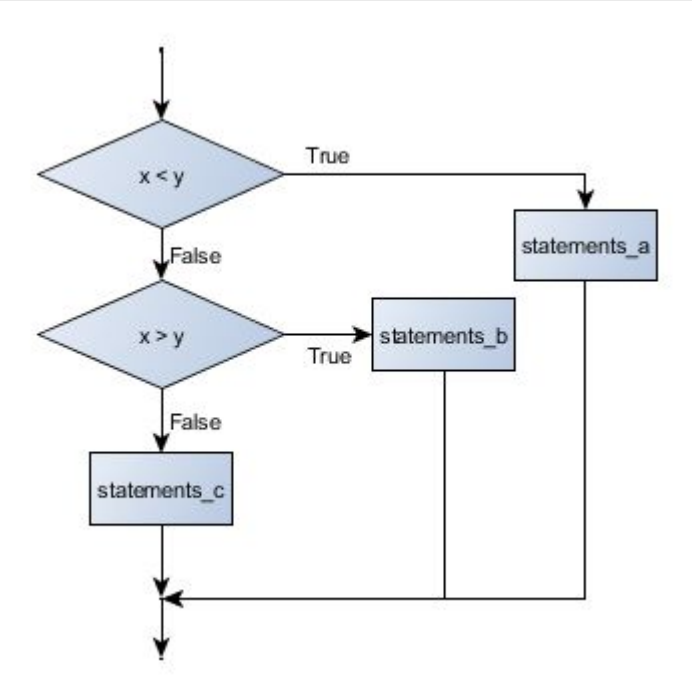

```
if (조건식) {
    블록1
} else {
    블록2
}
```

if 문만 있는 문장




```
if (조건식) {
    블록1
}
```

if - else if - else 문



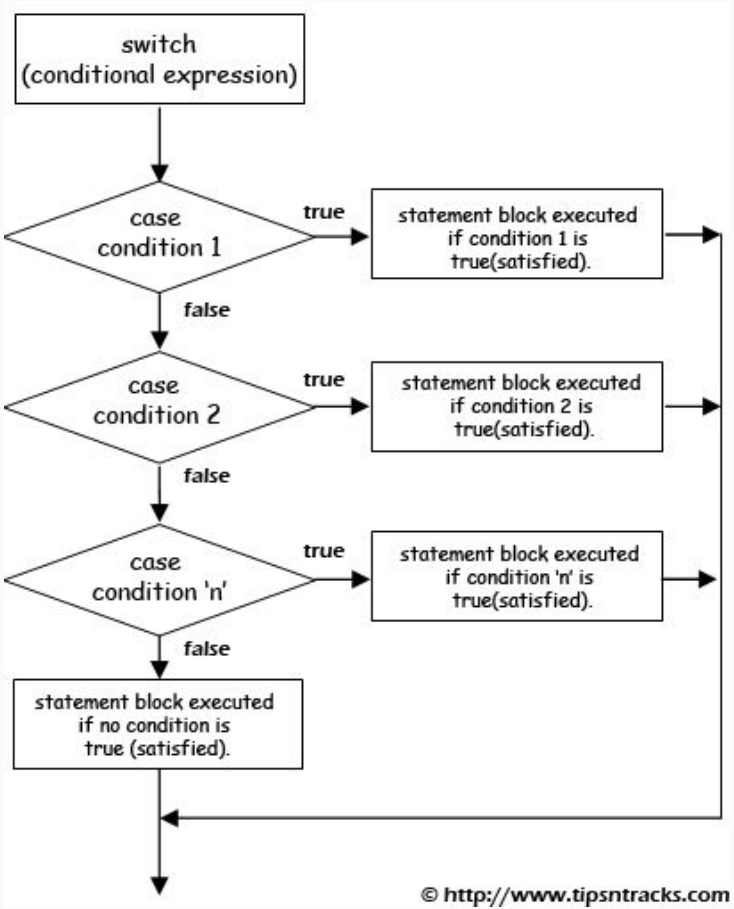

```
if (조건식1) {
    블록1
} else if (조건식2) {
    블록2
} else {
    최종블록
}
```

리스트 3-4 길고 뭔가 깔끔하지 않은 소스 코드


```
System.out.println("당신의 운세를 점쳐 드립니다");
int fortune = new java.util.Random().nextInt(4) + 1;

if (fortune == 1) {
    System.out.println("대길");
} else if (fortune == 2) {
    System.out.println("중길");
} else if (fortune == 3) {
    System.out.println("길");
} else {
    System.out.println("흉");
}
```

switch 문




```java
switch (fortune) {
    case 1:
        System.out.println("대길");
        break;
    case 2:
        System.out.println("중길");
        break;
    case 3:
        System.out.println("길");
        break;
    default:
        System.out.println("흉");
        break;
}
```

리스트 3-5 break 문을 잊으면..


```java
public class Main {
    public static void main(String[] args) {
        System.out.println("당신의 운세를 점쳐 드립니다");
        int fortune = 1;
        switch (fortune) {
            case 1:
                System.out.println("대길");

            case 2:
                System.out.println("중길");
                break;
            case 3:
                System.out.println("길");
                break;
            default:
                System.out.println("흉");
        }
    }
}
```

리스트 3-5 break 문을 쓰지 않는 기술 방법

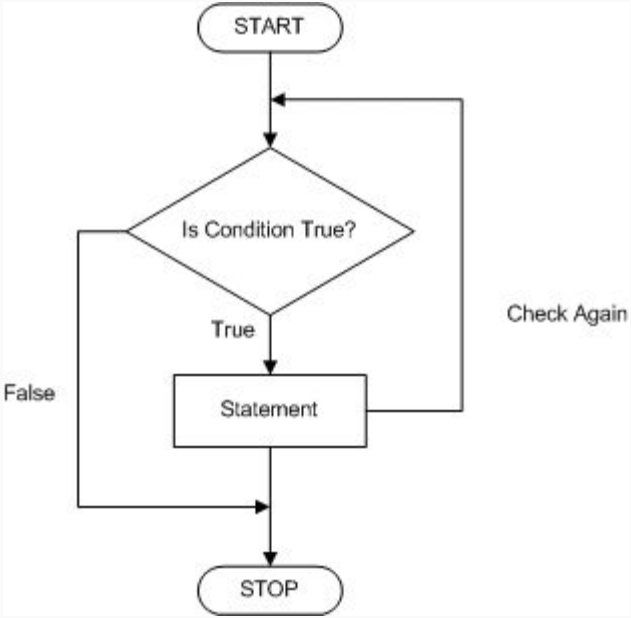

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("당신의 운세를 점쳐 드립니다");
        int fortune = new java.util.Random().nextInt(5) + 1;

        switch (fortune) {
            case 1:
            case 2:
                System.out.println("좋네요!");
                break;
            case 3:
                System.out.println("보통입니다");
                break;
            case 4:
            case 5:
                System.out.println("음...");
        }
    }
}
```

반복문의 종류

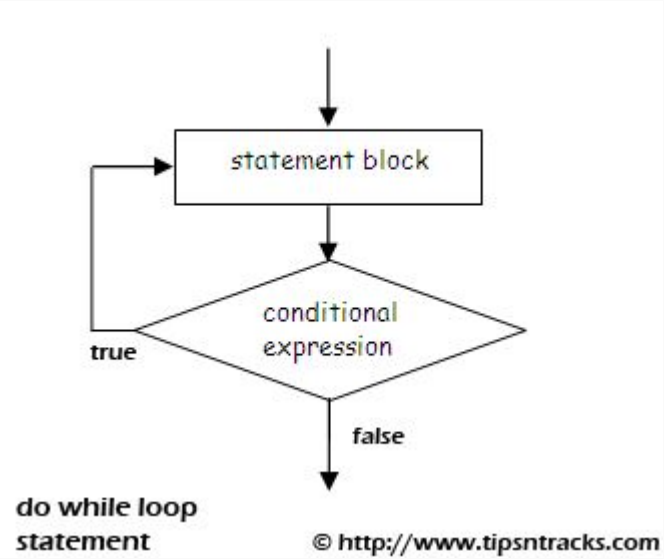
1. while 문
2. do - while 문
3. for 문

while 문 :
조건을 판단 후 반복



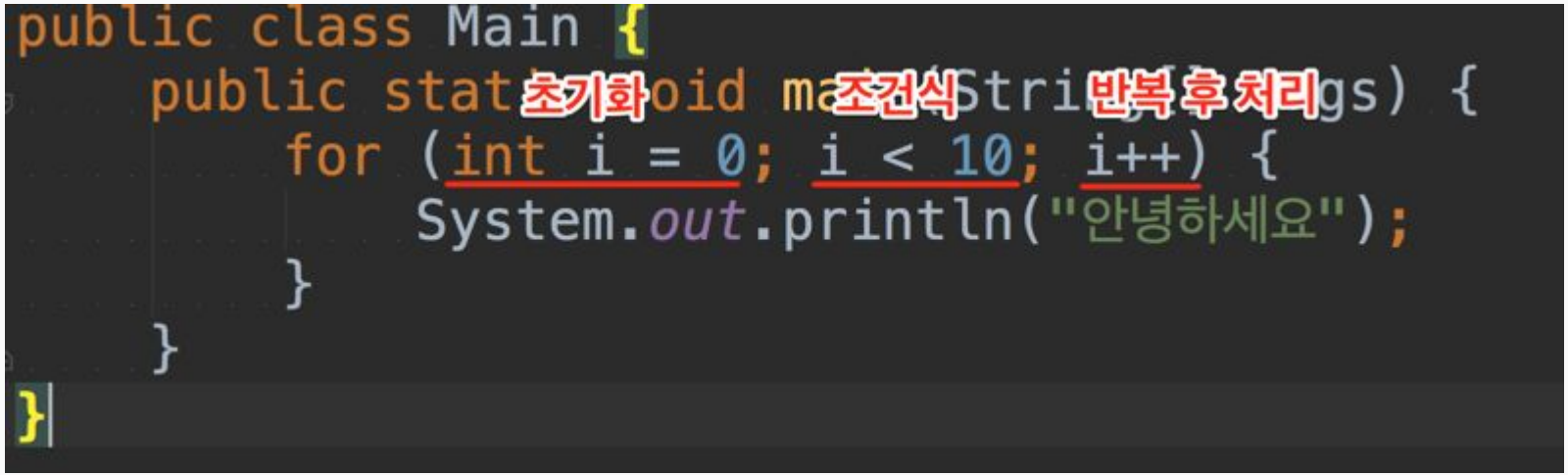

```
while (조건식) {
    블록
}
```

do - while 문 :
1회 실행 후 조건을 보고 반복




```
do {
  블록
} while (조건식)
```

for 문




```
public class Main {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.println("안녕하세요");
        }
    }
}
```

1. 루프 변수의 이름은 자유
2. 변수는 블록내에서 이용 가능
3. 블록 밖에서는 이용 불가

리스트 3-8 for 문의 루프 변수 i 의 내용을 표시

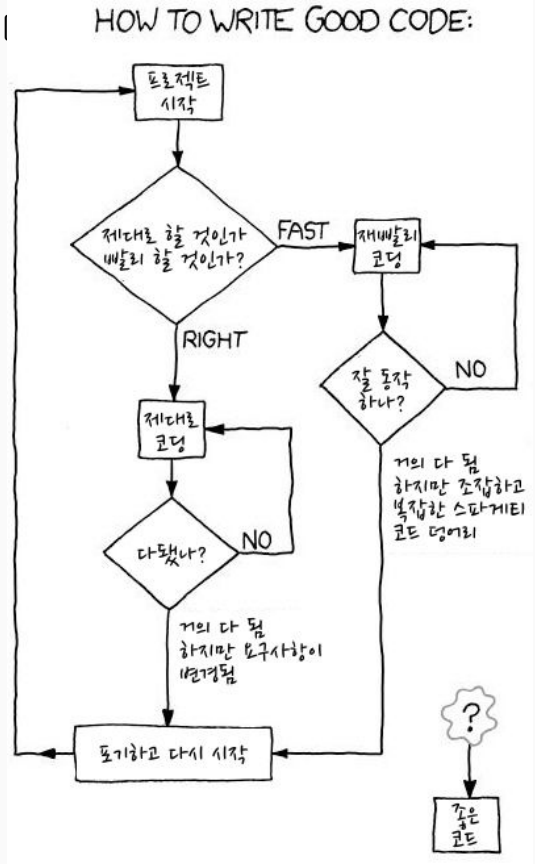

```
1  public class Main {
2      public static void main(String[] args) {
3          for (int i = 0; i < 3; i++) {
4              System.out.print("현재 " + (i + 1) + "회차->");
5          }
6      }
7  }
```

복잡한 for 문

- for (int i = 1; i < 10; i++) { … }
- for (int i = 0; i < 10; i += 2) { … }
- for (int i = 10; i > 0; i--) { … }
- for (; i < 10; i++) { … }
- for (int i = 0; i < 10;) { … }

제어구조의 네스트 (nest)

분기 안에 분기, 반복문 안에 분기 등 다중구조를 “네스트” (nest) 라고 한



리스트 3-9 for 문의 루프를 2중으로 하여 구구단 출력


```java
public class Main {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                System.out.print(i * j);      // 곱셈의 결과
                System.out.print(" ");        // 공백을 출력
            }
            System.out.println();             // 개행
        }
    }
}
```

반복문의 중단

경우에 따라 반복문 도중에 중단을 하고 싶을 때가 있다.
이럴 경우 Java 에서는 **break 문**과 **continue 문** 2종류의 중단 방법이 준비되어 있다.

```
for (int i = 1; i < 10; i++) {
    if (i == 3) {
        break;
    }
    System.out.println(i);
}
```

break 문

for 문을 종료

```
for (int i = 1; i < 10; i++) {
    if (i == 3) {
        continue;
    }
    System.out.println(i);
}
```

continue 문

이번만 중단하고, 계속 해서 루프를 진행

무한 루프

무한 루프란, 말 그대로 **무한하게 반복하는 제어구조** 를 말 한다.

프로그래밍 초보들은 실수로 무한 루프에 빠지게 되는 경우가 있다.

의도적으로 무한 루프를 만드는 경우 다음 2가지 경우가 일반적이다.

- while (true) { … }
- for (;;) { … }

제어구문

- 순차, 분기, 반복의 3가지 제어구조를 조합하여 어떤 프로그램도 만들 수 있다
- 분기와 반복은 “조건문" 과 “블록"으로 구성되어 있다
- 조건식의 평가결과는 true 또는 false 여야 한다
- 문자열을 비교 할 때는 “==”이 아니라 “equals”를 사용한다
- 블록 안에 정의한 변수는 블록 종료와 함께 소멸한다
- 제어구문은 네스트 가능

분기

- if 문 또는 switch 문을 사용해서 분기를 실현한다
- if 문은 “if 만의 구문", “if - else 구문", “if - else if - else 구문" 의 3종류
- switch 문의 블록은 break 문으로 벗어나는 것이 가능

반복

- while 문, do - while 문 또는 for 문을 사용하여 반복을 실현
- while 문의 블록은 최저 0회 이상, do - while 문의 블록은 최저 1회 이상 실행
- for 문은 루프 변수를 이용하여 “ ~회 반복” 하는 경우에 사용
- break 문을 실행하면 반복 자체를 중단하고, continue 문을 실행하면 이번만 중단하고 다음 반복으로 넘어간다

연습문제 3-1

다음 문장을 Java 조건식으로 기술하시오

1. 변수 weight 의 값이 60과 같다
2. 변수 age1 과 age2 의 합계를 2배 한 것이 60을 넘는다
3. 변수 age 가 홀수다
4. 변수 name 에 저장된 문자열이 “스마트"와 같다

연습문제 3-2

다음 중 조건식(if문의)으로 사용 할 수 있는 것을 고르시오

1. cost = 300 * 1.05
2. 3
3. age != 30
4. true
5. b + 5 < 20
6. gender = true

연습문제 3-3

Java 프로그램으로 작성하시오

1. int형 변수 gender 를 선언하고, 0 또는 1을 대입한다 (어떤 것이라도 상관없음)
   또한, int 형 변수 age 를 선언하고, 적당한 숫자를 대입한다.
2. 화면에 “안녕하세요” 를 표시한다
3. 만약 변수 gender 가 0이면 “나는 남자입니다", 그렇지 않으면 “나는 여자입니다" 를 표시한다
4. 만약 변수 gender 가 남자이면 age 변수의 값을 표시하고, 뒤에 “살입니다" 를 붙여서 표시한다.
5. 마지막에 “잘 부탁합니다" 를 표시한다

연습문제 3-4


```
1  public class Main {
2      public static void main(String[] args) {
3          boolean clear = true;
4          if (clear == true) {
5              System.out.println("세탁을 한다");
6              System.out.println("산책을 한다");
7          } else
8              System.out.println("영화를 본다");
9      }
10 }
```

```
7          } else
8              System.out.println("영화를 본다");
9              System.out.println("잔다");
10     }
11 }
```

3행의 clear 가 false 일 경우 “영화를 본다" 다음에 “잔다" 를 표시하기 위해, 9행에 “잔다"를 표시하는 행을 추가하였다.

하지만 프로그램이 의도한 대로 움직이지 않았다. 어느 부분의 오류 때문에 어떤 현상이 발생하는지 생각 해 보시오.
그리고 프로그램을 수정 하시오.

연습문제 3-5

switch 문을 이용하여 다음 조건을 만족하는 프로그램을 작성하시오

1. 화면에 “[메뉴] 1:검색 2:등록 3:삭제 4:변경 >” 을 표시한다
2. 키보드로 숫자를 입력하고, 변수 selected 에 대입한다.
3. 만약 변수 selected 가 1 이면 “검색합니다", 2이면 “등록합니다", 3이면 “삭제합니다", 4이면 “변경합니다"를 표시한다
4. selected 가 1 부터 4 사이의 값이 아니라면 아무것도 하지 않는다

연습문제 3-6

다음 조건을 만족하는 프로그램을 작성하시오

1. 화면에 “[숫자 맞추기 게임]” 을 표시한다
2. 0 부터 9 까지의 정수 중에서 랜덤하게 수를 1개 생성해서 변수 ans 에 대입한다
3. for 문을 이용해 “5회 반복 하는 루프"를 만든다. 아래의 4. ~ 7. 은 루프 안에 기술 한다
4. 화면에 “0 ~ 9 사이의 숫자를 입력 하세요” 를 표시한다
5. 숫자를 입력해, 변수 num 에 대입한다
6. 만약 변수 num 이 변수 ans 와 같으면 “정답!” 이라고 화면에 표시하고 반복을 종료
7. 만약 변수 num 이 변수 ans 와 같지 않으면 “다릅니다"를 표시한다
8. 반복 블록의 바깥에, “게임을 종료합니다" 라고 화면에 표시한다