

4. 조건문

#0.강의/1.자바로드맵/1.자바-입문

- /if문1 - if, else
- /if문2 - else if
- /if문3 - if문과 else if문
- /switch문
- /삼항 연산자
- /문제와 풀이1
- /문제와 풀이2
- /정리

if문1 - if, else

조건문 시작

지금까지 살펴본 프로그램은 단순히 위에서 아래로 순서대로 한 줄씩 실행되었다.

특정 조건에 따라서 다른 코드를 실행하려면 어떻게 해야할까? 예를 들어서 만약 18살 이상이면 "성인입니다"를 출력하고, 만약 18살 미만이라면 "미성년자입니다."를 출력해야 한다.

아마도 다음과 같이 코딩을 해야할 것 같다.

```
만약 (나이 >= 18)면 "성인입니다"  
만약 (나이 < 18)면 "미성년자입니다"
```

영어로 하면 다음과 같다.

```
if (age >= 18) "성인입니다"  
if (age < 18) "미성년자입니다"
```

이렇게 특정 조건에 따라서 다른 코드를 실행하는 것을 조건문이라 한다.

조건문에는 if 문, switch 문이 있다. 둘다 특정 조건에 따라서 다른 코드를 실행하는 것이라 생각하면 된다.

먼저 if 문부터 알아보자.

if문

if 문은 특정 조건이 참인지 확인하고, 그 조건이 참(true)일 경우 특정 코드 블록을 실행한다.

```
if (condition) {  
    // 조건이 참일 때 실행되는 코드  
}
```

코드 블록: {} (중괄호) 사이에 있는 코드

If1

```
package cond;  
  
public class If1 {  
    public static void main(String[] args) {  
        int age = 20; // 사용자 나이  
  
        if (age >= 18) {  
            System.out.println("성인입니다.");  
        }  
  
        if (age < 18) {  
            System.out.println("미성년자입니다.");  
        }  
    }  
}
```

실행 결과

성인입니다.

age = 20 값을 주면 처음 if 문에서 참이 된다.

if (age >= 18) 분석

```
//age = 20  
if (age >= 18) {"성인입니다"}
```

```
if (20 >= 18) {"성인입니다"} //age의 값은 20이다.  
if (true) {"성인입니다"} //조건이 참으로 판명된다.  
{"성인입니다"} //if문에 있는 코드 블록이 실행된다.
```

조건이 참이므로 "성인입니다."가 화면에 출력된다.

이후에 다음 코드가 실행된다.

if (age < 18) 분석

```
if (age < 18) {"미성년자입니다"}  
if (20 < 18) {"미성년자입니다"} //age의 값은 20이다.  
if (false) {"미성년자입니다"} //조건이 거짓으로 판명된다.  
//해당 코드 블록은 실행되지 않는다.
```

조건이 거짓이므로 if 문 블록을 실행하지 않고, 빠져나온다. 따라서 "미성년자입니다"는 화면에 출력되지 않는다.

int age = 20의 값을 15로 변경하면 미성년자입니다.가 출력되는 것을 확인할 수 있다.

else문

else 문은 if 문에서 만족하는 조건이 없을 때 실행하는 코드를 제공한다.

```
if (condition) {  
    // 조건이 참일 때 실행되는 코드  
} else {  
    // 만족하는 조건이 없을 때 실행되는 코드  
}
```

else 문을 사용하면 앞서 진행했던 프로그램을 다음과 같이 더 간략하게 바꿀 수도 있다.

- 기존: 만약 18살 이상이면 "성인입니다"를 출력하고, 만약 18살 미만이라면 "미성년자입니다."를 출력해야 한다.
- 변경: 만약 18살 이상이면 "성인입니다"를 출력하고, 그렇지 않으면 "미성년자입니다."를 출력해야 한다.

쉽게 이야기해서 18살이 넘으면 성인이고, 그렇지 않으면 모두 미성년자이다.

아마도 다음과 같이 코딩을 해야할 것 같다.

```
만약 (나이 >= 18)면 "성인입니다"  
그렇지 않으면 "미성년자입니다"
```

영어로 하면 다음과 같다.

```
if (age >= 18) "성인입니다"
else "미성년자입니다"
```

If2

```
package cond;

public class If2 {
    public static void main(String[] args) {
        int age = 20; // 사용자의 나이

        if (age >= 18) {
            System.out.println("성인입니다."); //참일 때 실행
        } else {
            System.out.println("미성년자입니다."); //만족하는 조건이 없을 때 실행
        }
    }
}
```

실행 결과

성인입니다.

`int age = 20`의 값을 `15`로 변경하면 `미성년자입니다.`가 출력되는 것을 확인할 수 있다.

if문2 - else if

다음 문제를 코드로 풀어보자.

문제

당신은 연령에 따라 다른 메시지를 출력하는 프로그램을 작성해야 한다.

이 프로그램은 `int age` 라는 변수를 사용해야 하며, 연령에 따라 다음의 출력을 해야 한다.

- 7세 이하일 경우: "미취학"
- 8세 이상 13세 이하일 경우: "초등학생"
- 14세 이상 16세 이하일 경우: "중학생"
- 17세 이상 19세 이하일 경우: "고등학생"
- 20세 이상일 경우: "성인"

`if` 문을 사용해서 코드를 작성해보자.

If3

```
package cond;

public class If3 {
    public static void main(String[] args) {
        int age = 14;

        if(age <= 7) { //~7: 미취학
            System.out.println("미취학");
        }
        if(age >= 8 && age <= 13) { //8~13: 초등학생
            System.out.println("초등학생");
        }
        if(age >= 14 && age <= 16) { //14~16: 중학생
            System.out.println("중학생");
        }
        if(age >= 17 && age <= 19) { //17~19: 고등학생
            System.out.println("고등학생");
        }
        if(age >= 20) { //20~: 성인
            System.out.println("성인");
        }
    }
}
```

이 코드는 다음과 같은 단점이 있다.

- 불필요한 조건 검사: 이미 조건을 만족해도 불필요한 다음 조건을 계속 검사한다. 예를 들어서 나이가 5살이라면 미취학이 이미 출력이 된다. 그런데 나머지 `if` 문을 통한 조건 검사도 모두 실행해야 한다.
- 코드 효율성: 예를 들어서 나이가 8살인 초등학생이라면 미취학을 체크하는 조건인 `age <= 7` 을 통해 나이가

이미 8살이 넘는다는 사실을 알 수 있다. 그런데 바로 다음에 있는 초등학생을 체크하는 조건에서 `age >= 8` && `age <= 13` 라는 2가지 조건을 모두 수행한다. 여기서 `age >= 8`이라는 조건은 이미 앞의 `age <= 7`이라는 조건과 관련이 있다. 결과적으로 조건을 중복 체크한 것이다.

이런 코드에 `else if`를 사용하면 불필요한 조건 검사를 피하고 코드의 효율성을 향상시킬 수 있다.

else if

`else if` 문은 앞선 `if` 문의 조건이 거짓일 때 다음 조건을 검사한다. 만약 앞선 `if` 문이 참이라면 `else if`를 실행하지 않는다.

if-else 코드

```
if (condition1) {  
    // 조건1이 참일 때 실행되는 코드  
} else if (condition2) {  
    // 조건1이 거짓이고, 조건2가 참일 때 실행되는 코드  
} else if (condition3) {  
    // 조건2이 거짓이고, 조건3이 참일 때 실행되는 코드  
} else {  
    // 모든 조건이 거짓일 때 실행되는 코드  
}
```

쉽게 이야기해서 이렇게 전체 `if` 문을 하나로 묶는다고 보면 된다. 이렇게 하면 특정 조건이 만족하면 해당 코드를 실행하고 `if` 문 전체를 빠져나온다. 특정 조건을 만족하지 않으면 다음 조건을 검사한다. 여기서 핵심은 순서대로 맞는 조건을 찾아보고, 맞는 조건이 있으면 딱 1개만 실행이 되는 것이다.

참고로 `else`는 생략할 수 있다.

else 생략 코드

```
if (condition1) {  
    // 조건1이 참일 때 실행되는 코드  
} else if (condition2) {  
    // 조건1이 거짓이고, 조건2가 참일 때 실행되는 코드  
}
```

이제 앞서 만든 코드를 `else if`를 사용해서 완성해보자.

If4

```
package cond;

public class If4 {
    public static void main(String[] args) {
        int age = 14;

        if(age <= 7) { //~7: 미취학
            System.out.println("미취학");
        } else if(age <= 13) { //~8~13: 초등학생
            System.out.println("초등학생");
        } else if(age <= 16) { //~14~16: 중학생
            System.out.println("중학생");
        } else if(age <= 19) { //~17~19: 고등학생
            System.out.println("고등학생");
        } else { //~20~: 성인
            System.out.println("성인");
        }
    }
}
```

age = 7인 경우

`if(age <= 7)`의 조건이 참이다. "미취학"을 출력하고 전체 `if` 문 밖으로 나간다.

age = 13인 경우

`if(age <= 7)`의 조건이 거짓이다. 다음 조건으로 넘어간다.

`else if(age <= 13)`의 조건이 참이다. "초등학생"을 출력하고 전체 `if` 문 밖으로 나간다.

age = 50인 경우

`if(age <= 7)`의 조건이 거짓이다. 다음 조건으로 넘어간다.

`else if(age <= 13)`의 조건이 거짓이다. 다음 조건으로 넘어간다.

`else if(age <= 16)`의 조건이 거짓이다. 다음 조건으로 넘어간다.

`else if(age <= 19)`의 조건이 거짓이다. 다음 조건으로 넘어간다.

`else` 만족하는 조건 없이 `else` 까지 왔다. `else`에 있는 "성인"을 출력하고 전체 `if` 문 밖으로 나간다.

if문3 - if문과 else if문

if 문에 else if를 함께 사용하는 것은 서로 연관된 조건일 때 사용한다. 그런데 서로 관련이 없는 독립 조건이면 else if를 사용하지 않고 if 문을 각각 따로 사용해야 한다.

예시

```
// 예시1. if-else 사용: 서로 연관된 조건이어서, 하나로 묶을 때
if (condition1) {
    // 작업1 수행
} else if (condition2) {
    // 작업2 수행
}

// 예시2. if 각각 사용: 독립 조건일 때
if (condition1) {
    // 작업1 수행
}
if (condition2) {
    // 작업2 수행
}
```

예시 1은 작업1, 작업2 둘 중 하나만 수행된다. 그런데 예시 2는 조건만 맞다면 둘다 수행될 수 있다.

if 문에 여러 조건이 있다고 항상 if-else로 묶어서 사용할 수 있는 것은 아니다. 조건이 서로 영향을 주지 않고 각각 수행해야 하는 경우에는 else if 문을 사용하면 안되고, 대신에 여러 if 문을 분리해서 사용해야 한다. 여러 독립적인 조건을 검사해야 하는 경우가 그런 상황의 대표적인 예시이다. 즉, 각 조건이 다른 조건과 연관되지 않고, 각각의 조건에 대해 별도의 작업을 수행해야 할 때 이런 상황이 발생한다.

예제를 통해 자세히 이해해보자.

문제

온라인 쇼핑몰의 할인 시스템을 개발해야 한다. 한 사용자가 어떤 상품을 구매할 때, 다양한 할인 조건에 따라 총 할인 금액이 달라질 수 있다.

각각의 할인 조건은 다음과 같다.

- 아이템 가격이 10000원 이상일 때, 1000원 할인
- 나이가 10살 이하일 때 1000원 할인

이 할인 시스템의 핵심은 **한 사용자가 동시에 여러 할인을 받을 수 있다는 점**이다.

예를 들어, 10000원짜리 아이템을 구매할 때 1000원 할인을 받고, 동시에 나이가 10살 이하이면 추가로 1000원 더 할인을 받는다. 그래서 총 2000원 까지 할인을 받을 수 있다.

If5

```
package cond;

public class If5 {
    public static void main(String[] args) {
        int price = 10000; // 아이템 가격
        int age = 10; // 나이
        int discount = 0;

        if (price >= 10000) {
            discount = discount + 1000;
            System.out.println("10000원 이상 구매, 1000원 할인");
        }

        if (age <= 10) {
            discount = discount + 1000;
            System.out.println("어린이 1000원 할인");
        }

        System.out.println("총 할인 금액: " + discount + "원");
    }
}
```

실행 결과

```
//price = 10000, age = 10
10000원 이상 구매, 1000원 할인
어린이 1000원 할인
총 할인 금액: 2000원
```

- 이 코드에서는 각각 독립된 if 문이 있다. 따라서 해당하는 모든 할인을 적용한다.
- 만약 else if 를 쓰면, 첫 번째로 충족하는 조건만 할인이 적용되고 나머지는 무시된다. 따라서 사용자는 나머지 할인을 놓칠 수 있다.

if 문을 사용해야 하는 곳에 else if 를 사용해서 어떤 문제가 발생하는지 확인해보자.

If6 - else if문 적용

```
package cond;

public class If6 {
    public static void main(String[] args) {
        int price = 10000; // 아이템 가격
        int age = 10; // 나이
        int discount = 0;

        if (price >= 10000) {
            discount = discount + 1000;
            System.out.println("10000원 이상 구매, 1000원 할인");
        } else if (age <= 10) {
            discount = discount + 1000;
            System.out.println("어린이 1000원 할인");
        } else {
            System.out.println("할인 없음");
        }

        System.out.println("총 할인 금액: " + discount + "원");
    }
}
```

실행 결과

```
//price = 10000, age = 10
10000원 이상 구매, 1000원 할인
총 할인 금액: 1000원
```

- 첫 번째로 충족되는 조건인 1000원 할인만 적용되고, if 문을 빠져나온다. 따라서 사용자는 나머지 할인을 놓치게 된다.

정리

if 문을 각각 사용할지, if와 else if를 함께 묶어서 사용할지는 요구사항에 따라 다르다. 둘의 차이를 이해하고 적절하게 사용하면 된다.

참고 - if문 {} 중괄호 생략

다음과 같이 if 문 다음에 실행할 명령이 하나만 있을 경우에는 {} 중괄호를 생략할 수 있다. else if, else도 마

찬가지이다.

```
if (true)
    System.out.println("if문에서 실행됨");
```

다음과 같은 경우에는 두번째 문장은 if 문과 무관하다. 만약 둘다 if 문 안에 포함하려면 {} 를 사용해야 한다.

```
if (true)
    System.out.println("if문에서 실행됨");
    System.out.println("if문에서 실행 안됨");
```

만약 둘다 if 문 안에 포함하려면 다음과 같이 {} 를 사용해야 한다.

```
if (true) {
    System.out.println("if문에서 실행됨");
    System.out.println("if문에서 실행 안됨");
}
```

프로그래밍 스타일에 따라 다르겠지만, 일반적으로는 if 문의 명령이 한개만 있을 경우에도 다음과 같은 이유로 중괄호를 사용하는 것이 좋다.

- **가독성:** 중괄호를 사용하면 코드를 더 읽기 쉽게 만들어 준다. 조건문의 범위가 명확하게 표시되므로 코드의 흐름을 더 쉽게 이해할 수 있다.
- **유지보수성:** 중괄호를 사용하면 나중에 코드를 수정할 때 오류를 덜 발생시킬 수 있다. 예를 들어, if 문에 또 다른 코드를 추가하려고 할 때, 중괄호가 없으면 이 코드가 if 문의 일부라는 것이 명확하지 않을 수 있다.

switch문

다음 문제를 코드로 풀어보자

당신은 회원 등급에 따라 다른 쿠폰을 발급하는 프로그램을 작성해야 한다.

이 프로그램은 int grade 라는 변수를 사용하며, 회원 등급(grade)에 따라 다음의 쿠폰을 발급해야 한다.

- 1등급: 쿠폰 1000
- 2등급: 쿠폰 2000

- 3등급: 쿠폰 3000
- 위의 등급이 아닐 경우: 쿠폰 500

각 쿠폰이 할당된 후에는 "발급받은 쿠폰 " + 쿠폰값 을 출력해야 한다.

2등급 사용자 출력 예)

```
발급받은 쿠폰 2000
```

if 문을 사용해서 코드를 작성해보자.

Switch1

```
package cond;

public class Switch1 {

    public static void main(String[] args) {
        //grade 1:1000, 2:2000, 3:3000, 나머지: 500
        int grade = 2;

        int coupon;
        if (grade == 1) {
            coupon = 1000;
        } else if (grade == 2) {
            coupon = 2000;
        } else if (grade == 3) {
            coupon = 3000;
        } else {
            coupon = 500;
        }
        System.out.println("발급받은 쿠폰 " + coupon);
    }
}
```

실행 결과

```
발급받은 쿠폰 2000
```

switch 문

switch 문은 앞서 배운 if 문을 조금 더 편리하게 사용할 수 있는 기능이다.

참고로 if 문은 비교 연산자를 사용할 수 있지만, switch 문은 단순히 값이 같은지만 비교할 수 있다.

switch 문은 조건식에 해당하는 특정 값으로 실행할 코드를 선택한다.

```
switch (조건식) {  
    case value1:  
        // 조건식의 결과 값이 value1일 때 실행되는 코드  
        break;  
    case value2:  
        // 조건식의 결과 값이 value2일 때 실행되는 코드  
        break;  
    default:  
        // 조건식의 결과 값이 위의 어떤 값에도 해당하지 않을 때 실행되는 코드  
}
```

- 조건식의 결과 값이 어떤 case의 값과 일치하면 해당 case의 코드를 실행한다.
- break 문은 현재 실행 중인 코드를 끝내고 switch 문을 빠져나가게 하는 역할을 한다.
- 만약 break 문이 없으면, 일치하는 case 이후의 모든 case 코드들이 순서대로 실행된다.
- default는 조건식의 결과값이 모든 case의 값과 일치하지 않을 때 실행된다. if 문의 else와 같다. default 구문은 선택이다.
- if, else-if, else 구조와 동일하다.

앞서 작성한 코드를 switch 문으로 변경해보자.

Switch2

```
package cond;  
  
public class Switch2 {  
  
    public static void main(String[] args) {  
        //grade 1:1000, 2:2000, 3:3000, 나머지: 500  
        int grade = 2;  
  
        int coupon;  
        switch (grade) {  
            case 1:  
                coupon = 1000;  

```

```

        break;
    case 2:
        coupon = 2000;
        break;
    case 3:
        coupon = 3000;
        break;
    default:
        coupon = 500;
    }
    System.out.println("발급받은 쿠폰 " + coupon);
}
}

```

실행 결과

발급받은 쿠폰 2000

break 문이 없으면?

만약 break 문이 없으면 어떻게 되는지 확인하기 위해 조건을 변경해보자.

비즈니스 요구사항이 변경되었다. **2등급도 3등급과 같이 3000원 쿠폰을 준다고** 해보자.

Switch3

```

package cond;

public class Switch3 {

    public static void main(String[] args) {
        //grade 1:1000, 2:3000(변경), 3:3000, 나머지: 500
        int grade = 2;

        int coupon;
        switch (grade) {
            case 1:
                coupon = 1000;
                break;
            case 2:

```

```

        case 3:
            coupon = 3000;
            break;
        default:
            coupon = 500;
            break;
    }
    System.out.println("발급받은 쿠폰 " + coupon);
}
}

```

- 예를 들어서 grade가 2등급이면 먼저 case 2가 실행된다.
- 그런데 case 2에는 break문이 없다. 그러면 중단하지 않고 바로 다음에 있는 case 3의 코드를 실행한다. 여기서 coupon = 3000;을 수행하고 break문을 만나서 switch문 밖으로 빠져나간다.
- "발급받은 쿠폰 3000"이 출력된다.

if문 vs switch문

switch문의 조건식을 넣는 부분을 잘 보면 `x > 10`과 같은 참 거짓의 결과가 나오는 조건이 아니라, 단순히 값만 넣을 수 있다.

switch문은 조건식이 특정 case와 같은지만 체크할 수 있다. 쉽게 이야기해서 값이 같은지 확인하는 연산만 가능하다. (문자도 가능)

반면에 if문은 참 거짓의 결과가 나오는 조건식을 자유롭게 적을 수 있다. 예) `x > 10`, `x == 10`

정리하자면 switch문 없이 if문만 사용해도 된다. 하지만 특정 값에 따라 코드를 실행할 때는 switch문을 사용하면 if문보다 간결한 코드를 작성할 수 있다.

자바 14 새로운 switch문

switch문은 if문보다 조금 덜 복잡한 것 같지만, 그래도 코드가 기대보다 깔끔하게 나오지는 않는다.

이런 문제를 해결하고자 자바14부터는 새로운 switch문이 정식 도입되었다.

기존 코드를 새로운 switch문으로 개발하면 다음과 같다.

```

package cond;

public class Switch3 {

```

```

public static void main(String[] args) {
    //grade 1:1000, 2:2000, 3:3000, 나머지: 500
    int grade = 2;

    int coupon = switch (grade) {
        case 1 -> 1000;
        case 2 -> 2000;
        case 3 -> 3000;
        default -> 500;
    };
    System.out.println("발급받은 쿠폰 " + coupon);
}
}

```

기존 switch문과 차이는 다음과 같다.

- -> 를 사용한다.
- 선택된 데이터를 반환할 수 있다.

새로운 switch문은 더 많은 내용을 담고 있다. 지금 이해하기에 어려운 내용들이 있으므로, 자세한 내용은 별도로 다룬다.

삼항 연산자

if 문을 사용할 때 다음과 같이 단순히 참과 거짓에 따라 특정 값을 구하는 경우가 있다.

CondOp1

```

package cond;

public class CondOp1 {

    public static void main(String[] args) {
        int age = 18;
        String status;
        if (age >= 18) {
            status = "성인";
        } else {

```



```

        status = "미성년자";
    }
    System.out.println("age = " + age + " status = " + status);
}
}

```

실행 결과, age = 18

```
age = 18 status = 성인
```

실행 결과, age = 17

```
age = 17 status = 미성년자
```

이 예제는 참과 거짓에 따라 `status` 변수의 값이 달라진다.

이렇게 단순히 참과 거짓에 따라서 특정 값을 구하는 경우 **삼항 연산자** 또는 **조건 연산자**라고 불리는 `?:` 연산자를 사용할 수 있다.

이 연산자를 사용하면 `if` 문과 비교해서 코드를 단순화 할 수 있다.

우선 코드부터 보자. 삼항 연산자를 사용하면 다음과 같이 코드를 간결하게 만들 수 있다.

CondOp2

```

package cond;

public class CondOp2 {

    public static void main(String[] args) {
        int age = 18;
        String status = (age >= 18) ? "성인" : "미성년자";
        System.out.println("age = " + age + " status = " + status);
    }
}

```

실행 결과 분석

```
String status = (age >= 18) ? "성인" : "미성년자"; //age=18
String status = (true) ? "성인" : "미성년자"; //조건이 참이므로 참 표현식 부분이 선택된다.
String status = "성인"; //결과
```

삼항 연산자

(조건) ? 참_표현식 : 거짓_표현식

- 삼항 연산자는 항이 3개라는 뜻이다. 조건, 참_표현식, 거짓_표현식 이렇게 항이 3개이다. 자바에서 유일하게 항이 3개인 연산자여서 삼항 연산자라 한다. 또는 특정 조건에 따라 결과가 나오기 때문에 조건 연산자라고도 한다.
- 조건에 만족하면 참_표현식이 실행되고, 조건에 만족하지 않으면 거짓_표현식이 실행된다. 앞의 if, else 문과 유사하다.
- if 문 처럼 코드 블럭을 넣을 수 있는 것이 아니라 단순한 표현식만 넣을 수 있다.

삼항 연산자 없이 if 문만 사용해도 된다. 하지만 단순히 참과 거짓에 따라서 특정 값을 구하는 삼항 연산자를 사용하면 if 문 보다 간결한 코드를 작성할 수 있다.

문제와 풀이1

코딩이 처음이라면 필독!

프로그래밍이 처음이라면 아직 코딩 자체가 익숙하지 않기 때문에 문제와 풀이에 상당히 많은 시간을 쓰게 될 수 있다. 강의를 들을 때는 다 이해가 되는 것 같았는데, 막상 혼자 생각해서 코딩을 하려니 잘 안되는 것이다. 이것은 아직 코딩이 익숙하지 않기 때문인데, 처음 코딩을 하는 사람이라면 누구나 겪는 자연스러운 현상이다.

문제를 스스로 풀기 어려운 경우, 너무 고민하기 보다는 먼저 **강의 영상의 문제 풀이 과정을 코드로 따라하면서 이해하자. 반드시 코드로 따라해야 한다.** 그래야 코딩하는 것에 조금씩 익숙해질 수 있다. 그런 다음에 정답을 지우고 스스로 문제를 풀어보면 된다. 참고로 강의를 듣는 시간만큼 문제와 풀이에도 많은 시간을 들여야 제대로 성장할 수 있다!

문제: "학점 계산하기"

학생의 점수를 기반으로 학점을 출력하는 자바 프로그램을 작성하자. 다음과 같은 기준을 따른다.

- 90점 이상: "A"

- 80점 이상 90점 미만: "B"
- 70점 이상 80점 미만: "C"
- 60점 이상 70점 미만: "D"
- 60점 미만: "F"

점수는 변수(`int score`)로 지정하고, 해당 변수를 기반으로 학점을 출력하자.

출력 예시

score: 95
출력: 학점은 A입니다.

score: 85
출력: 학점은 B입니다.

score: 75
출력: 학점은 C입니다.

score: 65
출력: 학점은 D입니다.

score: 55
출력: 학점은 F입니다.

정답: "학점 계산하기"

```
package cond.ex;

public class ScoreEx {
    public static void main(String[] args) {
        int score = 85;

        if (score >= 90) {
            System.out.println("학점은 A입니다.");
        } else if (score >= 80) {
            System.out.println("학점은 B입니다.");
        } else if (score >= 70) {
            System.out.println("학점은 C입니다.");
        } else if (score >= 60) {
            System.out.println("학점은 D입니다.");
        }
    }
}
```

```

    } else {
        System.out.println("학점은 F입니다.");
    }
}
}

```

문제: "거리에 따른 운송 수단 선택하기"

주어진 거리에 따라 가장 적합한 운송 수단을 선택하는 프로그램을 작성하자. 다음과 같은 기준을 따른다.

- 거리가 1km 이하이면: "도보"
- 거리가 10km 이하이면: "자전거"
- 거리가 100km 이하이면: "자동차"
- 거리가 100km 초과이면: "비행기"

거리는 변수(`int distance`)로 지정하고, 해당 변수를 기반으로 운송 수단을 출력하자.

출력 예시

```

distance: 1
출력: 도보를 이용하세요.

```

```

distance: 5
출력: 자전거를 이용하세요.

```

```

distance: 25
출력: 자동차를 이용하세요.

```

```

distance: 150
출력: 비행기를 이용하세요.

```

정답: "거리에 따른 운송 수단 선택하기"

```

package cond.ex;

public class DistanceEx {
    public static void main(String[] args) {

```

```

int distance = 25;

if (distance <= 1) {
    System.out.println("도보를 이용하세요.");
} else if (distance <= 10) {
    System.out.println("자전거를 이용하세요.");
} else if (distance <= 100) {
    System.out.println("자동차를 이용하세요.");
} else {
    System.out.println("비행기를 이용하세요.");
}
}
}

```

문제: "환율 계산하기"

특정 금액을 미국 달러에서 한국 원으로 변환하는 프로그램을 작성하자. 환율은 1달러당 1300원이라고 가정하자. 다음과 같은 기준을 따른다.

- 달러가 0미만이면: "잘못된 금액입니다."
- 달러가 0일 때: "환전할 금액이 없습니다."
- 달러가 0 초과일 때: "환전 금액은 (계산된 원화 금액)원입니다."

금액은 변수(`int dollar`)로 지정하고, 해당 변수를 기반으로 한국 원으로의 환전 금액을 출력하자.

출력 예시

dollar: -5

출력: 잘못된 금액입니다.

dollar: 0

출력: 환전할 금액이 없습니다.

dollar: 10

출력: 환전 금액은 13000원입니다.

정답: "환율 계산하기"

```
package cond.ex;

public class ExchangeRateEx {
    public static void main(String[] args) {
        int dollar = 10;

        if (dollar < 0) {
            System.out.println("잘못된 금액입니다.");
        } else if (dollar == 0) {
            System.out.println("환전할 금액이 없습니다.");
        } else {
            int won = dollar * 1300;
            System.out.println("환전 금액은 " + won + "원입니다.");
        }
    }
}
```

문제와 풀이2

문제: "평점에 따른 영화 추천하기"

요청한 평점 이상의 영화를 찾아서 추천하는 프로그램을 작성하자.

- 어바웃타임 - 평점9
- 토이 스토리 - 평점8
- 고질라 - 평점7

평점 변수는 `double rating` 을 사용하세요. `if` 문을 활용해서 문제를 풀자.

출력 예시

- rating: 9
- 출력:
- '어바웃타임' 을 추천합니다.
- rating: 8

- 출력:
 - '어바웃타임'을 추천합니다.
 - '토이 스토리'를 추천합니다.
-
- rating: 7.1
 - 출력:
 - '어바웃타임'을 추천합니다.
 - '토이 스토리'를 추천합니다.
-
- rating: 7
 - 출력:
 - '어바웃타임'을 추천합니다.
 - '토이 스토리'를 추천합니다.
 - '고질라'를 추천합니다.

정답: "평점에 따른 영화 추천하기"

```
package cond.ex;

public class MovieRateEx {
    public static void main(String[] args) {
        double rating = 7.1;

        if (rating <= 9) {
            System.out.println("'어바웃타임'을 추천합니다.");
        }

        if (rating <= 8) {
            System.out.println("'토이 스토리'를 추천합니다.");
        }

        if (rating <= 7) {
            System.out.println("'고질라'를 추천합니다.");
        }
    }
}
```

문제: "학점에 따른 성취도 출력하기"

`String grade` 라는 문자열을 만들고, 학점에 따라 성취도를 출력하는 프로그램을 작성하자. 각 학점은 다음과 같은 성취도를 나타낸다.

- "A": "탁월한 성과입니다!"
- "B": "좋은 성과입니다!"
- "C": "준수한 성과입니다!"
- "D": "향상이 필요합니다."
- "F": "불합격입니다."
- 나머지: "잘못된 학점입니다."

`switch` 문을 사용해서 문제를 해결하자.

출력 예시

```
grade: "B"
출력: "좋은 성과입니다!"
```

```
grade: "A"
출력: "탁월한 성과입니다!"
```

```
grade: "F"
출력: "불합격입니다."
```

정답: "학점에 따른 성취도 출력하기"

```
package cond.ex;

public class GradeSwitchEx {
    public static void main(String[] args) {
        String grade = "B";

        switch(grade) {
            case "A":
                System.out.println("탁월한 성과입니다!");
                break;
            case "B":
                System.out.println("좋은 성과입니다!");
                break;
            case "C":
```



```

        System.out.println("준수한 성과입니다!");
        break;
    case "D":
        System.out.println("항상이 필요합니다.");
        break;
    case "F":
        System.out.println("불합격입니다.");
        break;
    default:
        System.out.println("잘못된 학점입니다.");
    }
}
}

```

문제: 더 큰 숫자 찾기

여러분은 두 개의 정수 변수 `a`와 `b`를 가지고 있다. `a`의 값은 10이고, `b`의 값은 20이다. 삼항 연산자를 사용하여 두 숫자 중 더 큰 숫자를 출력하는 코드를 작성하자.

출력 예시

더 큰 숫자는 20입니다.

정답: 더 큰 숫자 찾기

```

package cond.ex;

public class CondOpEx {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        int max = (a > b) ? a : b;

        System.out.println("더 큰 숫자는 " + max + "입니다.");
    }
}

```

문제: 홀수 짝수 찾기

정수 x 가 주어지면 x 가 짝수이면 "짝수"를, x 가 홀수이면 "홀수"를 출력하는 프로그램을 작성하자
삼항 연산자를 사용해야 한다.

참고로 $x \% 2$ 를 사용하면 홀수, 짝수를 쉽게 계산할 수 있다.

출력 예시

```
x: 2
출력: x = 2, 짝수

x: 3
출력: x = 3, 홀수
```

정답 홀수 짝수 찾기

```
package cond.ex;

public class EvenOddEx {
    public static void main(String[] args) {
        int x = 2;
        String result = (x % 2 == 0) ? "짝수" : "홀수";
        System.out.println("x = " + x + ", " + result);
    }
}
```

정리