Java 기초

조건문

조건문

조건문

조건문

- 조건문
  - 로직의 분기를 태울 경우 사용하는 문장.
  - Java에서 사용되는 조건문으로 if와 switch ~ case 문장이 있다.
  - 기존의 삼항연산자 또한 분기가 가능하지만 로직이 아닌 값을 반환하는 것이 차이점.
  - 이런 로직의 분기를 통해 상황에 따라 어떤 로직을 실행시킬 지 정할 수 있다.
  - Java에서 가장 중요한 명령어 중 하나.

IF

- if
  - Java의 조건문 중 하나의 형태
  - if문을 통해 어떤 분기를 태워야 할지 정할 수 있다.
  - if문 안에 if문을 넣을 수 있고 또 if문 안에 if문을 넣을 수 있다.
  - 또한 if문의 조건을 만족시키지 못할 경우 if문의 조건에 반대되는 로직을 세울 수도 있다.
  - 이처럼 조건에 맞춰 다양한 분기를 시행할 수 있는 것이 바로 이 IF문이다.

IF

- 단순 IF문

```
if(조건식 & 논리값){ 조건식/값이 참일 때 실행되는 로직;  }
```

  - if문의 조건식이 참이면 if문 {}안에 있는 로직이 실행되고 아닐 경우 if문을 무시하고 바로 아래의 로직을 실행시키는 문장
  - 가장 단순하면서 기본적인 if문
  - 조건식의 여부에 따라서 해당 로직을 실행시킬 수도 있고 아닐수도 있을 경우에 이 로직을 사용한다.
  - if문이 참이었을 경우 실행되는 로직을 {}안에 작성한다.

IF

단순 if문 예제

```
Run | Debug
public static void main(String[] args) {
    // 단순 if문 예제
    int a = 1;
    if(a > 0){
        System.out.println("a는 1보다 큽니다.");
    }
    // if문 안에 조건식이 아닌 논리값을 사용하여
    // if문을 강제로 태우거나 if문을 아예 못태우도록 만들 수는 있지만.
    // 크게 의미는 없다.
    if(true){
        System.out.println("이 값은 무조건 실행됩니다.");
    }
    // if문의 조건이 끝나면 반드시 바로 아래의 로직을 실행한다.
    System.out.println("이 문장은 if문에 관계없이 바로 다음에 실행되는 문장입니다.");
    // if문에서의 더 복잡한 조건식을 만들고 싶다면 &&나 ||, !과 같은 논리연산자를
    // 활용해서 복잡한 조건식을 만들 수도 있다.
    int b = 5;
    if( (a > 0)  && (b < 10)){
        System.out.println("i는 0보다 크고 j는 10보다 작다.");
    }

    // 산술연산을 활용해서 조건식 안에 산술 연산을 넣을 수도 있다.
    if( ((a+=1)>0) && ((b+=1)<10)){
        System.out.println("변형된 a & b의 값");
        System.out.println("a : "+a+"   b : "+b);
    }
}
```

IF

- if ~ else 문

```
if(조건식 & 논리값){
    조건식이나 논리값이 참일 경우 실행되는 로직
}else{
    조건식이나 논리값이 거짓일 경우 실행되는 로직
}
```

  - 이지선다형 조건문
  - 해당 조건이 참일 경우 if문의 로직을 해당 조건이 거짓일 경우 else 문 장안의 로직을 실행시킨다.
  - 동전의 앞 뒷면 같이 정하고자 할 경우 해당 로직을 쓸 수 있다.

IF

if ~ else 문 예제

```java
// if ~ else 문 예제
// if ~ else 문 사용시 조건식에 따라 하나의 로직만 사용된다.
int a = 3;
if (a < 5) {
    System.out.println("a는 5보다 작습니다");
}else{
    System.out.println("a는 5보다 큽니다");
}

// if ~ else 문을 통한 가감산 예제
int b = 5;
if(b >5){
    b--;
}else{
    b++;
}

System.out.println("현재 출력하려는 b의 값 : "+b);

// 간단한 로직은 삼항 연산자로 대체가 가능하다.
int c = 5;
c = (c > 5)? --c : ++c ;
System.out.println(c);
```

IF

- 중첩 if문

```
if(조건1){
        if(조건2){
                if(조건3){
                        .....
```

  - if문 안에 if문을 넣을 수 있고 또 그 안에 if문을 중첩 시킬 수 있다.
  - 이러한 문장을 중첩 if문이라고 한다.
  - if문을 중첩 시킬 때 if문의 중첩 개수는 상관이 없다.
  - 단 너무 과용을 할 경우 로직을 알아보기 힘들 수 있으므로 자제하여 사용할 것

IF

중첩 if문 예제

```
Run | Debug
public static void main(String[] args) {
    // 중첩 if문 예제
    int a = 3;
    if(a > 0){
        if(a > 1){
            if(a > 2){
                System.out.println("a는 2보다 큽니다.");
            }
        }
    }

    // 중첩 if문 시에 if문을 순차적으로 들어가서 순차적으로 빠져나간다.
    // 단 안에 조건식을 만족시키지 못하면 거기서 중지하고 로직을
    // 다시 빠져나온다.
    int b = 4;
    if(b > 0){
        System.out.println("첫번째 if문 시작지점");
        if(b > 1){
            System.out.println("두번째 if문 시작지점");
            if (b > 2) {
                System.out.println("세번째 if문 시작지점");
                if (b > 7) {
                    System.out.println("네번째 if문 시작지점");
                    System.out.println("네번째 if문 끝지점");
                }
                System.out.println("세번째 if문 끝 지점");
            }
            System.out.println("두번째 if문 끝지점");
        }
        System.out.println("첫번째 if문 끝 지점");
    }
}
```

IF

중첩 if문 예제

```
Run | Debug
public static void main(String[] args) {
    // if문 뿐만이 아닌 if else문에서도 사용이 가능하다.
    int a = 5;
    if (a > 6) {
        System.out.println("a는 6보다 큽니다");
    } else {
        if (a > 5) {
            System.out.println("a는 5보다 큽니다");
        } else {
            if (a > 4) {
                System.out.println("a는 4보다 큽니다");
            } else {
                System.out.println("a는 4보다 같거나 작습니다.");
            }
        }
    }

}
```

IF

- if~ else if ~ else 문

```
if(조건식){ 조건에 해당하는 로직...
}else if(조건식){조건에 해당하는 로직...
}else if(조건식){조건에 해당하는 로직...
}else{조건에 해당하는 로직... }
```

  - 다수개의 조건식을 넣어 비교문을 작성할 경우 사용하는 문장.
  - 좀 더 세밀한 조건을 줄 경우 이 문장을 사용한다.
  - 맨 위의 조건부터 비교하여 만약 그 조건이 참이면 해당 문장을 실행하고 if ~ else if ~ else 문 밖의 로직을 실행, 그렇지 않을 경우 다음 문장의 조건식을 비교한다.
  - 모든 조건식을 비교 했음에도 불구하고 만약 조건에 일치하는 것이 없다면 마지막의 else 로 간다.
  - if ~ else if ~ else 문 안에도 중첩 if문 사용이 가능하다.
  - 끝의 else문은 쓰지 않아도 크게 상관은 없다.

IF

if ~ else if ~ else 문 예제

```
Run | Debug
public static void main(String[] args) {
    // if ~ else if ~ else 예제
    int a = 10;
    if (a == 7) {
        System.out.println("a 는 7 입니다.");
    }else if(a == 8){
        System.out.println("a 는 8 입니다.");
    }else if(a == 9){
        System.out.println("a 는 9 입니다.");
    }else if(a == 10){
        System.out.println("a 는 10 입니다.");
    }else{
        System.out.println("a에는 해당되는 값이 없습니다.");
    }

    // if ~ else if ~ else 시 else문은 생략해도
    // 크게 로직에 상관이 없다. 단 모든 조건을 거치지 않을경우
    // if문을 스킵하고 if문 아래 다음 로직으로 이동한다.
    int b = 8;
    if (b == 1) {
        System.out.println("b 는 1 입니다.");
    }else if(a == 2){
        System.out.println("b 는 2 입니다.");
    }else if(a == 3){
        System.out.println("b 는 3 입니다.");
    }else if(a == 4){
        System.out.println("b 는 4 입니다.");
    }

    System.out.println("b는 해당되는 값이 존재하지 않습니다.");
}
```

IF

if ~ else if ~ else 문 예제

```
Run | Debug
public static void main(String[] args) {
    // 학점 계산기 예제
    int score = 72;

    if(score >= 90){
        System.out.println("A 학점입니다.");
    }else if((score < 90) && (score >= 80)){
        System.out.println("B 학점입니다.");
    }else if((score < 80) && (score >= 70)){
        System.out.println("C 학점입니다.");
    }else if((score < 70) && (score >= 60)){
        System.out.println("D 학점입니다.");
    }else{
        System.out.println("F 학점입니다.");
    }

    // 위에서부터 조건문을 비교하므로 위의 조건식을
    // 통과하게 되면 다음 조건식은 위의 조건식 자체의
    // 범위가 배제되었다고 이야기 할 수 있다.
    if(score >= 90){
        System.out.println("A 학점입니다.");
    }else if(score >= 80){
        System.out.println("B 학점입니다.");
    }else if(score >= 70){
        System.out.println("C 학점입니다.");
    }else if(score >= 60){
        System.out.println("D 학점입니다.");
    }else{
        System.out.println("F 학점입니다.");
    }
}
```

Switch~Case

- Switch~Case 문
  - Switch~Case문은 기존의 if ~ else if ~ else 문을 대체할 수 있는 조건 문이다.
  - Switch~Case문의 기본적인 문장은 다음과 같다.

```
switch(변수 / 값){
case 값1 :
        해당 분기에서 실행되는 로직;
        break;
case 값2 :
        해당 분기에서 실행되는 로직;
        break;
case 값3 :
        ……
default :
        분기를 타지 않았을 때의 로직;
```

Switch~Case

Switch~Case 예제

```java
Run | Debug
public static void main(String[] args) {
    // switch~case 예제
    int a = 3;
    switch (a) {
        case 1:
            System.out.println("a의 값은 1입니다.");
            break;
        case 2:
            System.out.println("a의 값은 2입니다.");
            break;
        case 3:
            System.out.println("a의 값은 3입니다.");
            break;
        case 4:
            System.out.println("a의 값은 1입니다.");
            break;
        default:
            System.out.println("a에 해당하는 값이 없습니다.");
            break;
    }
}
```

Switch~Case

- Switch~Case
  - switch~case 문은 if ~ else if ~ else 문과 동일하게 사용 가능하지만 몇 가지의 차이점이 있다
  - 첫번째는 switch~case문은 값의 동등 비교만 가능하다.
  - switch() 괄호 안에 들어갈 내용은 조건식이 아니라 변수 혹은 단일 값, 산술연산으로 나온 결과 말고는 들어갈 수가 없다.
  - 단지 case문에서의 값과 switch에서의 값을 비교해 동등하면 해당 로직을 태우고 그렇지 않으면 로직을 태우지 않는다.
  - default는 else의 생략처럼 생략 가능하다.

Switch~Case

Switch ~ Case 예제

```
// Switch ~ Case 예제 2
int a = 2;
// 값을 넣을 수 있다.
// 하지만 switch case문을 쓰는데 의미는 없다
switch (3) {
    case 1:
        break;
    default:
        break;
}
// 변수는 쓸 수 있다.
switch (a) {
    case 2:
        break;
    default:
        break;
}
// 산술연산은 쓸 수 있다.
switch (a+3) {
    case 1:
        break;
    default:
        break;
}
```

Switch~Case

Switch ~ Case 예제

```
// 대입 산술 연산자도 가능
switch(a+=4){
    case 7:
        break;
    default:
        break;
}
// 삼항 연산자도 가능
switch ((a>3)?5:1) {
    case 5:
        break;
    default:
        break;
}
```

```
// 비교연산은 불가
// switch (a>3) {
//     case true:
//         break;
//     default:
//         break;
// }
// 논리값도 불가
// switch (true) {
//     case true:
//         break;
//     default:
//         break;
// }
```

Switch~Case

Switch ~ Case 예제

Switch~Case

- break
  - Switch~Case문의 경우 각각의 Case의 맨 아래에 break를 적는다.
  - break를 쓰지 않게 되면 해당 로직만 실행되는 것이 아닌 아래있는 문장들도 전부 실행이 된다.
  - 이 실행은 다음 break를 만나기 전까지 계속 된다.
  - 오히려 이러한 형질을 이용해 범위 색인을 할 수도 있다.

Switch~Case

break 예제

```
Run | Debug
public static void main(String[] args) {
    int a = 2;
    switch (a) {
        case 1:
            System.out.println("a는 1입니다.");
        case 2:
            System.out.println("a는 2입니다.");
        case 3:
            System.out.println("a는 3입니다.");
        case 4:
            System.out.println("a는 4입니다.");
        default:
            System.out.println("a는 해당되는 값이 없습니다.");
    }
}
```

```
a는 2입니다.
a는 3입니다.
a는 4입니다.
a는 해당되는 값이 없습니다.
```

Switch~Case

break 예제

```
Run | Debug
public static void main(String[] args) {
    // Switch ~ Case에서 break문의 특성을 활용한 예제
    int a = 5;
    switch (a) {
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
            System.out.println("a는 1부터 7사이의 숫자입니다");
            break;
        case 8:
        case 9:
        case 10:
            System.out.println("a는 8부터 10사이의 숫자입니다.");
            break;
        default:
            System.out.println("a는 1부터 10사이의 숫자가 아닙니다");
            break;
    }
}
```

```
a는 1부터 7사이의 숫자입니다
```

Switch~Case

- 중첩 Switch~Case
  - Switch~Case문 또한 중첩이 가능하다.
  - Switch~Case문 안에 if문을 넣는 것 또한 가능하고 또한 if문 안에 Switch~Case문을 넣는 것 또한 가능하다.
  - 주의해야 할 점은 Switch~Case문을 너무 중첩해서 사용할 경우 로직을 읽기가 어려워지므로 주의할 것

Switch~Case

중첩 Switch~Case 예제

```
Run | Debug
public static void main(String[] args) {
    // 중첩 Switch ~ Case 문 예제
    int a = 7;
    switch (a) {
        case 7:
            switch((a>5)?5:3){
                case 3:
                    System.out.println("a는 5보다 작습니다.");
                    break;
                case 5:
                    System.out.println("a는 5보다 큽니다."); // 실행
                    break;
            }
            break;
        default:
            System.out.println("a에 해당하는 값이 없습니다");
            break;
    }
}
```

Switch~Case

중첩 Switch~Case 예제

```java
// if 문과 switch~case문 조합 예제
int a = 5;

if (a > 7) {
    switch (a) {
    case 10:
        System.out.println("a는 10 입니다.");
        break;
    case 9:
        System.out.println("a는 9 입니다.");
        break;
    case 8:
        System.out.println("a는 8 입니다.");
        break;
    }
}else if(a > 4){
    switch (a) {
    case 7:
        System.out.println("a는 7 입니다.");
        break;
    case 6:
        System.out.println("a는 6 입니다.");
        break;
    case 5:
        System.out.println("a는 5 입니다.");
        break;
    }
}else{
    System.out.println("a에 해당되는 값은 없습니다.");
}
```

```
a는 5 입니다.
```

조건문의 순서 따라가기

- 조건문의 순서 따라가기
  - 중첩 if문이 씌워진 경우 종종 로직을 따라 읽기 힘들 수 있다.
  - 기존의 if문의 성질을 이해하면 어떠한 복잡한 조건문이 오더라도 읽을 수 있다.
  - 가장 좋은 방법은 메모장과 필기도구를 이용해 옆에다 어떤 로직을 탈지를 적어가며 추적하는 것도 좋은 방법이다.

조건문의 순서 따라가기

조건문의 순서 따라가기 예제

```
Run | Debug
public static void main(String[] args) {
    // 중첩 if문 읽는 법
    // 중요한건 if문을 통한 블록화에 주의할 것.
    int a = 7;
    // 항상 조건은 위의 조건이 우선이며
    // 맨 위의 조건을 만족시키지 못할 경우 다음 아래 조건을
    // 비교한다. 아래 예제에서는 맨 위의 조건을 만족시키지
    // 못하므로 바로 다음 조건으로 이동한다.
    if(a < 5){

    // 다음 조건에서는 조건을 만족하므로 해당 분기의 로직을 실행한다.
    // 여기서 중요한 점은 어차피 해당 로직을 실행하고 if문을 빠져나오기
    // 때문에 다른 조건들을 고려할 필요는 없다.
    }else if(a < 10){
        // 이 부분에서도 마찬가지 결과가 나온다.
        // if문의 조건을 위에서부터 비교하여 맞는 조건이 나올때까지 내려간다.
        // 이후 맞는 조건이 나오면 그 부분의 로직만을 실행하고 바로 다음 아래의
        // 로직을 살펴보면 된다.
        if(a==9){

        }else if(a==8){

        }else if(a==7){

        }else{

        }
        // if문 밖에 있는 전 후의 문장들은 실행되는 문장들이니 주의할 것
        System.out.println("이것은 무조건 실행되는 문장입니다.");
    }else{

    }
}
```

조건문의 순서 따라가기

조건문의 순서 따라가기 예제

```
Run | Debug
public static void main(String[] args) {
    // if와 switch문 결합
    // 위의 if문과 큰 차이는 없으나 하나 주의할 점은
    // switch문은 break가 없을 경우 아래의 로직까지 전부 다 실행시킨다.
    // 이부분만 유의하면 충분히 읽고 판단이 가능하다.
    int a = 7;
    if (a>5) {
        switch (a) {
            case 6:
                System.out.println("a는 6입니다");
            case 7:
                System.out.println("a는 7입니다");
            case 8:
                System.out.println("a는 8입니다");
        }
    } else {

    }
}
```

랜덤(Random)

- 랜덤 함수
  - java에서는 랜덤하게 값을 발생시킬 수 있는 랜덤 함수가 존재한다.
  - 방법은 두가지가 있으며 하나는 Math라는 클래스를 이용해 사용하는 방법이다.
  - 또 한가지는 Random() 이라는 클래스를 활용해 랜덤 값을 도출하는 방법이 있다.
  - 두개 중 어느 것을 써도 상관이 없다.

랜덤(Random)

Math.random 예제

```
Run | Debug
public static void main(String[] args) {
    // Math.random은 결과 값이 무조건 double 타입이다.
    // 값은 0부터 0.99999999999... 까지의 값을 지닌다.
    // Math.random을 그냥 사용하기 보다는 연산을 통한 가공을
    // 통해 사람들이 많이 사용하고 있다.
    double d = Math.random();
    System.out.println(d);

    // 1부터 6까지의 랜덤 숫자 추출 예제
    int i = (int)(Math.random()*6)+1;
    System.out.println(i);
}
```

랜덤(Random)

Random() 클래스 이용

```java
import java.util.Random;

Run | Debug
public static void main(String[] args) {
    // Random 클래스 예제
    // Random 클래스는 java.util에 있으며 기존의 Math.random보다
    // 더 많은 기능을 가지고 있다. 그렇기에 연산을 통한 전처리를
    // 하고 싶지 않고 기능의 편의를 추구한다면 Random 클래스를
    // 사용할 수 있다.
    Random ran = new Random();

    // 0 ~ 0.9999.. 까지 값 출력
    System.out.println(ran.nextDouble());
    // 정수형 int의 최소부터 최대까지 출력
    System.out.println(ran.nextInt());
    // 정수형 int의 0부터 범위만큼 출력
    System.out.println(ran.nextInt(5));
    // 홀 짝 처럼 논리형 두가지 값만을 출력할 경우 사용
    System.out.println(ran.nextBoolean());

}
```