1. 제어문

1. 제어문이란 조건에 따라서 특정 소스코드의 실행이나 반복을 제어하는 구문.
2. 제어문에는 조건문과 반복문이 포함되어 있다.
3. 조건문은 조건에 따라 특정 소스코드를 실행. if구문, switch구문.
4. 반복문은 조건에 따라 특정 소스코드의 반복 실행. for구문, while구문.

2. 조건문

1. if구문
    - if(A) { B; }: 조건 A가 true일 때만 B를 실행.

```
if(10 < 5) {
    //if 조건이 false기 때문에 실행되지 않음
    System.out.println("true1");
}

if(10 > 5) {
    //if 조건이 true이기 때문에 true2 출력
    System.out.println("true2");
}
```

    - if 구문 {} 블록안의 실행문이 한줄이면 {}을 생략할 수 있다. 실행문이 여러줄일 경우에는 무조건 {} 블록으로 묶어줘야한다. 이 규칙은 모든 제어문에 동일하게 적용된다.

```
if(10 < 5)
    System.out.println("true1");

if(10 > 5) {
    System.out.println("true2");
    System.out.println("true3");
}
```

2. else if구문
    - if(A) { C; } else if(B) { D; }: A가 true면 C가 실행되고 A가 false면서 B가 true일 때 D가 실행된다.
    - else if 구문은 하나가 아닌 여러개를 사용할 수 있다.

```
int score = 85;

if(score >= 90) {
    System.out.println("A");
} else if(score >= 80) {
    System.out.println("B");
} else if(score >= 70) {
    System.out.println("C");
}
```

- else if문은 항상 if과 함께 사용돼야 한다. 독립적으로는 사용할 수 없다.
- if구문의 조건과 else if 조건이 연관되어 있어야 한다.

```
int a = 10;
int b = 20;

int c;

//연관된 조건 사용
if(a < 9) {
    c = 100;
} else if(a < 100) {
    c = 50;
}

//조건이 연관되어 있지 않아서 안좋은 else if구문의 사용
if(a < 9) {
    c = 100;
} else if(b < 100) {
    c = 50;
}
```

3. else구문
    - else구문은 if 바로 다음에 사용할 수도 있고 마지막 else if구문뒤에 붙여서 사용할 수도 있다.
    - else 구문이 존재하는 조건문 끝에는 조건을 더 추가할 수 없다.
    - else 구문은 모든 조건이 false 일 경우 실행할 실행문을 작성하는 구문
    - else 구문에는 조건을 작성할 수 없다.

```
int year = 2022;

if(year == 2023) {
    System.out.println("올해");
} else {
    System.out.println("올해가 아님");
}

int score = 55;
Strint grade;

if(score >= 90) {
    grade = "A";
} else if(score >= 80) {
    grade = "B";
} else if(score >= 70) {
    grade = "C";
} else {
    grade = "D";
}
```

- **예제: chap04_controlstatement._01_IfStatement.java**
- **예제: chap04_controlstatement._03_IfEx.java**

4. switch~case구문
    - switch~case 구문은 변수의 값을 조건으로 나눠서 처리하는 조건문

```
int num = 2;

switch(num) {
    case 1:
        //num의 값이 1일 때 실행될 내용
        System.out.println("num의 값은 1입니다.");
        //case문마다 break를 항상 써줘야한다.
        //여기서 switch문 종료한다는 의미.
        break;
    case 2:
        //num의 값이 2일 때 실행될 내용
        System.out.println("num의 값은 2입니다.");
        break;
    //else구문과 마찬가지로 위 조건이 false일 때 실행될
    default:
        System.out.println("num의 값은 1, 2가 아닙니다
        break;
}
```

- break; 구문을 사용하지 않으면 조건이 맞는 case 구문 아래의 모든 case 구문과 default 구문이 실행된다.

- **예제: chap04_controlstatement._02_SwitchCase.java**
- **예제: chap04_controlstatement._04_SwitchEx.java**

3. 반복문

1. 특정 조건에 부합할 때는 계속해서 특정 소스 코드를 반복실행하는 구문
2. 반복문에는 for, while, do~while 구문이 대표적으로 존재한다.
3. for 구문
    - for문은 초기화식, 조건식, 증감식 포함되어있는 반복문

```
for(초기화식(변수선언 및 초기화); 조건식(초기화식에서 선
    반복실행될 구문;
}

int sum = 0;

//i = 0 일 때 조건식으로 입장, 조건식이 true이므로 중괄
//i = 1 일 때 조건식으로 입장, 조건식이 true이므로 중괄
//i = 10일 때까지 동일하게 동작하다가
//i = 11일 때 조건식이 false가 되면서 반복문이 종료.
for(int i = 0; i <= 10; i++) {
    sum += i;
}
```

    - for문의 변형: for문의 변형은 초기화식이나 조건식 또는 증감식을 생략하는 것이다. 세 개의 식을 모두 생략할 수도 있다.

```
//1. 초기화식이 없는 경우
//초기화식을 생략하기 위해서는 for문에서 사용할 변수가 [
int i = 1;
int sum = 0;

for( ; i <= 100; i++) {
    sum += i;
}

//2. 조건식이 없는 경우
//조건식이 없으면 for문이 무한 반복되기 때문에 for문안에
for(int j = 0; ; j++) {
    sum += j;

    if(j == 100) {
        //break; 명령어는 바로 반복문을 종료시키는 구문
        break;
    }
}

//3. 증감식이 없는 경우
//증감식이 없으면 변수의 값이 변하지 않고 for문이 계속 [
for(int h = 0; h <= 100; ) {
    //3의 배수의 합
    sum += h;
    h += 3;
}

//4. 초기화식, 조건식, 증감식 모두 생략한 경우
int l = 0;

//무한 루프(무한 반복되는 구문)
for( ; ; ) {
  sum += l;

  l++;

  if(l == 1000) {
    break;
  }
}
```

4. while구문
    - while구문은 조건식만 존재하는 반복문. for구문보다 사용하기 편하다.
    - while구문은 조건식이 true일 경우 특정 소스코드를 반복실행

```
while(조건식) {
    반복 실행될 소스코드;
}

//1~100까지의 합
int i = 1;
int sum = 0;

while(i <= 100) {
    sum += i++;
}
```

- while구문의 무한 루프

```
while(true) {
    //무한 반복되는 구문
    //특정 조건일 때 break; 구문을 사용해서 반복문을 종
}
```

5. do~while구문
    - 반복 실행될 코드가 먼저 한 번 실행되고 조건식으로 가서 조건을 따지는 구문

```
do {
    //반복 실행될 코드
} while(조건식);
```

6. 중첩 for구문
    - for구문 안에서 for구문을 열어서 다수의 for구문을 사용하는 구문

```
for(초기화식1; 조건식1; 증감식1) {
    for(초기화식2; 조건식2; 증감식2) {
        ....
    }
}

//내부의 for구문이 한 사이클이 실행됐을 때 외부 for구문
for(int i = 0; i < 5; i++) {
    //i = 0 내부의 for구문이 5회 진행
    //i = 1 내부의 for구문이 5회 진행
    for(int j = 0; j < 5; j++) {
        //총 25회 실행
        System.out.println(j);
    }
    //총 5회 실행
    System.out.println(i);
}
```

7. 반복문의 break, continue 구문
    - 반복문을 종료시키는 역할 break;

```
for( ; ; ) {
    //break구문이 실행되는 순간 반복문이 종료
    break;
}
```

    - 중첩 for문에서의 break구문. break구문의 위치에 따라서 종료되는 반복문이 달라진다.

```
for( ; ; ) {
    //외부의 for문을 종료시키는 break;
    break;
    for( ; ; ) {
        //내부의 for문을 종료시키는 break;
        //외부의 for문은 계속 반복 실행된다.
        break;
    }
}
```

    - continue구문: continue 구문을 만나는 순간 for문에서는 증감식으로 이동. while문에서는 조건식으로 이동.

```
for(int i = 0; i < 100; i++) {
    //i가 짝수 일 때는 출력하지 않고 바로 증감식으로 이
    if(i % 2 == 0) {
        continue;
    }

    System.out.println(i);
}

int j = 0;

while(j < 100) {
    j++;

    //j가 짝수일 때는 출력하지 않고 바로 조건식으로 이동
    if(j % 2 == 0) {
        continue;
    }

    System.out.println(j);
}
```