변수

안 화 수

변수

- 변수(Variable)
  변수는 메모리상에 데이터를 저장하기 위한 기억 공간의 이름

- 자바의 변수 생성법

  자료형 변수명 = 값;

```
int  age = 25;
double  left = 1.5;
char  c1 = '자';
String  str = "자바";
String  subject[] = {"자바", "JSP", "python", "spring", "Kotlin" };
```

변수 명명 규칙

- 변수이름 명명 규칙

1. 영문자(A~Z, a~z)와 숫자(0~9)와 언더바(_), $의 조합으로 만들어 진다.
   ex) apple3, a_b, $apple

2. 첫 글자는 반드시 영문자, 언더바(_), $로 시작해야한다.
   숫자로 시작해서는 안된다.
   ex) apple(0), _pear(0), $apple(0), 3banana(X)

3. 변수명은 대.소문자를 구별한다.
   ex) apple, Apple, APPLE (서로 다른 변수로 인식함)

4. 2개의 단어가 붙어있는 경우에는 첫 문자는 소문자로 시작한다. (카멜 표기법)
   ex) firstName, maxSpeed

5. 변수명의 길이에는 제한이 없다.

6. 자바에서 사용하는 예약어는 변수명으로 사용할 수 없다.
   ex) int, char, if, else, switch, for, while, break, void, return etc

식별자 표기법

- 카멜 표기법(Camel Case)

1. 단어의 첫 글자는 소문자로 시작한다.
2. 두 번째 단어부터는 첫 글자를 대문자로 표기한다.
3. 단어 사이에는 공백이나 특수 문자가 없고, 단어들을 연결하여 작성한다.

변수 이름: myVariable, counterValue, userName
메서드 이름: calculateTotalAmount(), getUserInfo(), openFileInputStream()
파일 이름: myFile.java, myDocument.docx, myImage.jpg

카멜 표기법은 가독성을 높이고, 식별자를 명확하게 구분할 수 있도록 도와준다.
Java, JavaScript, C#, Python, 등 다양한 프로그래밍 언어에서 사용되며, 각 언어에 따라 약간의 변형이 있을 수 있다

식별자 표기법

- 스네이크 표기법(Snake Case)

1. Snake case(스네이크 케이스)는 변수, 함수, 데이터베이스 테이블 또는 열의 이름을 작성할 때 사용되는 명명 규칙 중 하나이다.
2. Snake case에서는 단어와 단어 사이를 밑줄(_)로 구분한다.
   각 단어는 소문자로 작성하며, 밑줄로 연결된다.
3. Snake case에서는 모든 글자를 소문자로 작성한다.
   대문자를 사용하지 않고 모든 단어가 소문자로 작성되므로 가독성이 좋다.
4. Snake case는 데이터베이스 테이블 이름과 열 이름을 작성할 때 주로 사용된다.
5. 데이터베이스 테이블 및 열은 스네이크 케이스로 작성되며, 데이터베이스와 어플리케이션 코드 간의 일관성을 유지하기 위해 사용된다.

데이터베이스 테이블 및 열: user_profile, first_name, phone_number
변수 또는 함수 이름: total_score, calculate_average, is_active

Snake case는 가독성이 좋고 언더스코어를 사용하여 단어를 명확하게 구분하기 때문에 코드 또는 데이터베이스 스키마를 읽고 이해하기 쉽다. Snake case는 주로 데이터베이스와의 통합 또는 프로젝트에서 코드 작성의 일관성을 유지하기 위해 사용된다.

식별자 표기법

❖ 파스칼 표기법(Pascal Case)

1. 파스칼 표기법(Pascal Case)은 프로그래밍 및 식별자 이름 작성 규칙 중 하나로, 다양한 프로그래밍 언어 및 플랫폼에서 사용되는 네이밍 규칙 중 하나이다.
2. 모든 단어의 첫 글자를 대문자로 시작한다.
3. 단어 사이에는 공백이나 특수 문자가 없고, 단어들을 연결하여 작성한다.

클래스 이름: MyClass, CarModel, UserProfile
타입 이름: Integer, StringBuffer, LinkedListNode

파스칼 표기법은 카멜 표기법과는 달리 모든 단어의 첫 글자를 대문자로 표기하기 때문에, 클래스나 타입을 다른 식별자와 명확하게 구분할 수 있도록 도와준다.
Java, C#, C++, Pascal 등 다양한 프로그래밍 언어에서 사용되며, 주로 클래스와 타입 이름을 작성할 때 적용된다.

변수의 범위

- 변수의 사용 범위

1. 지역 변수

2. 멤버 변수(field, 전역변수)

3. 정적 멤버변수(정적 필드)

변수의 범위

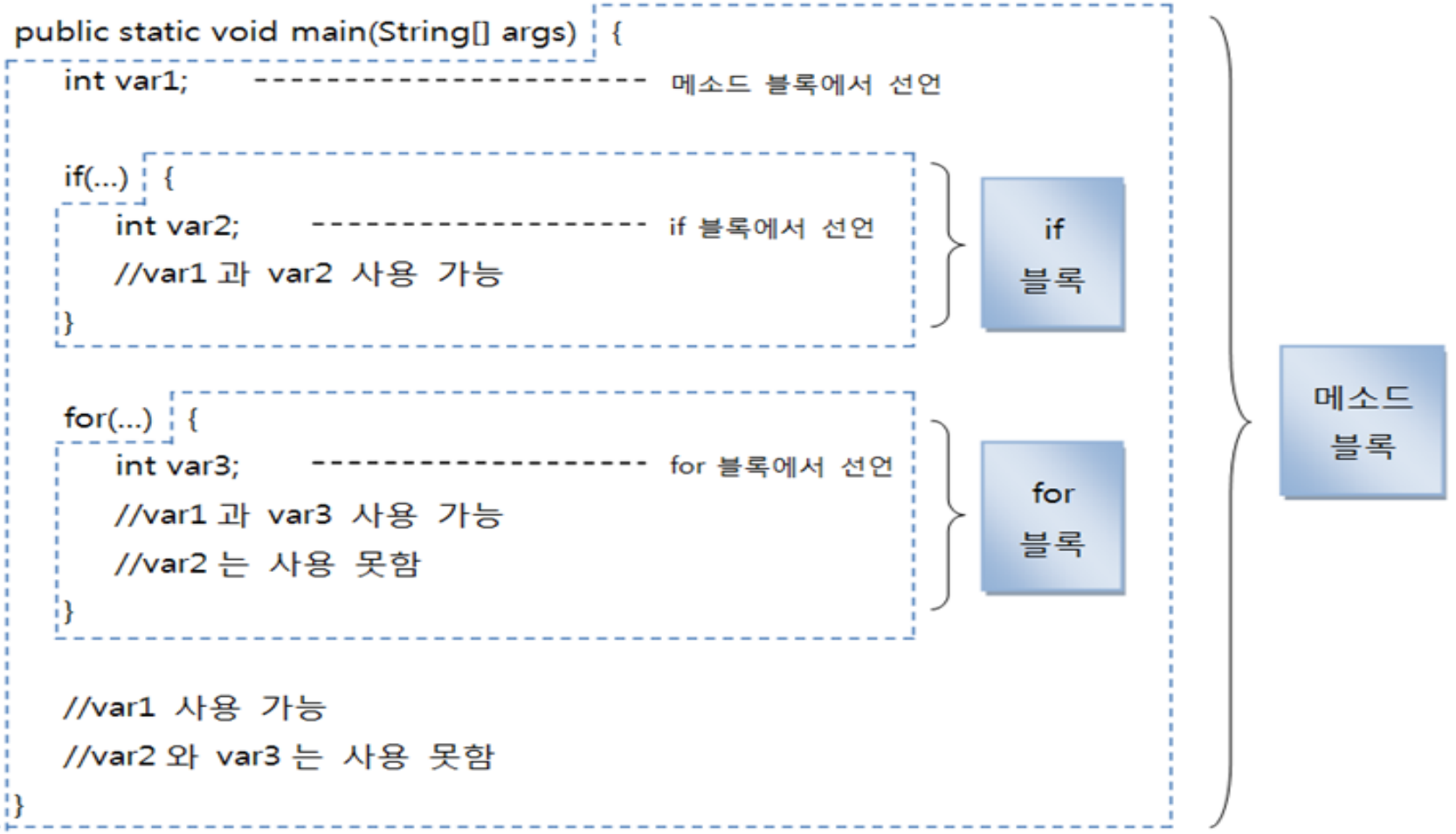
- 지역변수(Local Variable)

1. 메소드 안에서 정의 되는 변수
2. 매개 변수(parameter) : 메소드의 괄호안에서 사용되는 변수
3. 블럭문(조건문, 반복문) 안에서 사용되는 변수
4. 지역변수는 stack 메모리 영역에 저장된다.
5. 지역변수는 해당 메소드가 호출될 때 stack 메모리 영역에 할당되고, 메소드가 실행이 종료되면 자동으로 메모리에서 해제된다.

변수의 범위

- 지역변수(Local Variable)

  메소드 블록, if블록, for블록



변수의 범위

- 멤버변수(field, 전역변수)

1. 메소드 바깥쪽에 정의 되는 변수이다.
2. 멤버변수는 heap메모리 영역에 저장된다.
3. 클래스(생성자, 메소드) 안에서 모두 사용이 가능하다.
4. 참조형(클래스, 배열, 인터페이스)을 이용해서 new연산자로 객체를 생성할때 heap메모리 영역에 새로운 공간을 할당 받는다.

변수의 범위

❖ 정적 멤버변수(정적field)

1. 변수명 앞에 static을 붙어서 정적 멤버변수로 만들 수 있다. static int a = 10;

2. 공유를 목적으로 변수에 쉽게 접근할 수 있도록 만들 경우에만 정적 멤버변수를 사용한다.

int n = Integer.parseInt(String s);

3 . 정적 멤버변수는 클래스가 실행될 때 static영역(공유영역)에 자동으로 할당되고, 프로그램이 종료될 때 까지 값을 유지한다

4. 정적 멤버변수는 메모리를 할당 받을 때 자동으로 초기값이 설정된다.
int a=0, double b=0.0, boolean c = false

5. 정적 멤버변수에 접근하는 방법은 정적 멤버변수를 가지고 있는 클래스명으로 직접 접근 할 수 있다.

```
class StaticTest{
        static int a;                          // 정적 멤버변수
         public static void main(String[] args){
                    System.out.println( StaticTest.a );
         }
}
```

자료형

- 자바의 자료형

기본 자료형 - 수치형 - 정수형 byte (1Byte)
short (2Byte)
int (4Byte) ex) int a = 10;
long (8Byte)

실수형 float (4Byte)
double (8Byte) ex) double d=3.5;

문자형 char (2Byte) ex) char c1 = 'k';
char c2 = '안';

논리형 boolean ex) boolean b = true;

참조형(Reference Type) : 클래스, 배열, 인터페이스(Collection)
ex) String s1 = "50";
String s2 = new String("자바");

자료형

- 자바의 자료형 예제 (1/3)

```
public class Variable {

     public static void main(String[] args) {

// 변수 : 메모리상에 데이터를 저장하기 위한 기억 공간의 이름
// 변수를 만드는 방법 :  자료형  변수명 = 데이터(값);

// 기본 자료형 변수
// 1. 정수형 변수
        byte b1 = 10;              // -128 ~ 127
//      byte b2 = 130;             // 오버플로우 발생(오류 발생)
        short s = 100;             // -32768 ~ 32767
        int i = 1000;              // -21억 ~ 21억
        long l = 10000L;

        System.out.println("b1="+b1);
        System.out.println("s="+s);
        System.out.println("i="+i);
        System.out.println("l="+l);

// 2. 실수형 변수
        float ft1 = 3.14f;     // float형은  f를 붙여야 한다.
        float ft2 = 3.14F;
        float ft3 = (float)3.14;
        double d = 42.195;

        System.out.println("ft1="+ ft1);
        System.out.println("ft2="+ ft2);
        System.out.println("d="+ d);
        System.out.printf("%.1f\n", d);// 소수 첫째자리까지 출력 : 42.2
        System.out.printf("%.2f\n", d);// 소수 둘째자리까지 출력 : 42.20
```

자료형

- 자바의 자료형 예제 (2/3)

```
// 3. 문자형 변수
        char c1 = 'A';
        char c2 = '안';
        System.out.println("c1="+c1);
        System.out.println("c2="+c2);

// 4. 논리형 변수
        boolean bn1 = true;
        boolean bn2 = false;
        System.out.println("bn1="+bn1);
        System.out.println("bn2="+bn2);

// 참조형 변수 : 클래스
        String s1 = "자바";
        String s2 = new String("자바");
        if(s1 == s2) {          //  주소 비교
            System.out.println("같은 주소");
        }else {
            System.out.println("다른 주소");
        }
        if(s1.equals(s2)) {     //  데이터(값) 비교
            System.out.println("같은 값");
        }else {
            System.out.println("다른 값");
        }
```

자료형

- 자바의 자료형 예제 (3/3)

```
// 참조형 변수 : 배열 – 동일한 자료형의 데이터를 저장하는 정적인 자료구조
        int[] score  = {80, 90, 100};

        for(int j=0; j<score.length; j++) {
            System.out.print(score[j]+"\t");
        }
        System.out.println();


// 참조형 변수 : 인터페이스(List)
// 1. 순차적인 자료구조이다.
// 2. 여러가지 자료형의 데이터를 모두 저장할 수 있다.
// 3. 동적으로 공간의 크기를 늘릴 수 있다.

//      List list = new List();              //  오류발생
        List list = new ArrayList();
        list.add(30);
        list.add(3.14);
        list.add('j');
        list.add(true);
        list.add("자바");

        for(int k=0; k<list.size(); k++) {
            System.out.print(list.get(k)+"\t");
        }
    }
}
```

자료형 변환

- 자바의 자료형 변환

1. 기본 자료형 변환
   ex) double <---> int

2. Wrapper를 이용한 자료형 변환 (기본자료형 <---> 참조형)
   ex) int <---> String Wrapper 클래스(박싱과 언박싱)
   int n = Integer.parseInt("20");

3. 레퍼런스 형변환
   두 클래스 사이에 상속 관계가 있을 경우에만 레퍼런스 형변환이 가능하다.

자료형 변환

❖ 기본자료형 변환

자바의 기본 자료형 변수들 사이의 자료형 변환을 의미한다.

➢ 자동 형변환 : 컴파일러가 자동으로 자료형 변환해주는 변환을 의미
작은 자료형 데이터를 큰 자료형 변수에 저장하는 경우

ex) char -> int
int -> float
int -> double
float -> double

자료형 크기 순서

byte(1) < short(2) < int(4) < long(8) < float(4) < double(8)

자료형 변환

❖ 기본자료형 변환

자바의 기본 자료형 변수들 사이의 자료형 변환을 의미한다.

➢ 강제 형변환 : 프로그래머가 직접 자료형 변환을 해야 되는 변환을 의미
큰 자료형 데이터를 작은 자료형 변수에 저장하는 경우

ex) int -> char ex) char c = (char)97;
double -> int ex) int a = (int) 3.14;
double -> float