기본기 학습 – 배열과 클래스

✔ **참조형 변수**

① 배열(Array)

하나의 이름으로 여러 자리를 예약하는 기능

A[1] A[2] A[3] A[4]

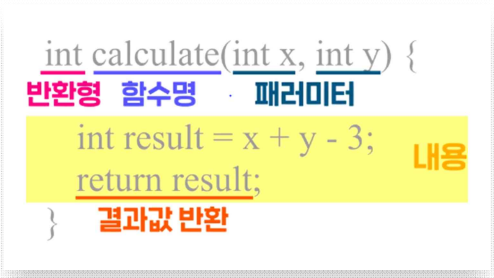
자바	int sample[10]; 자료형 배열이름[갯수] ►sample이라는 이름으로 10개의 int 공간을 예약
	sample[3] = 40; ►sample의 3번 자리에 숫자 40을 넣음
	int x = sample[3]; ► sample의 3번 자리에서 40이라는 값을 꺼내서 변수 x에 저장

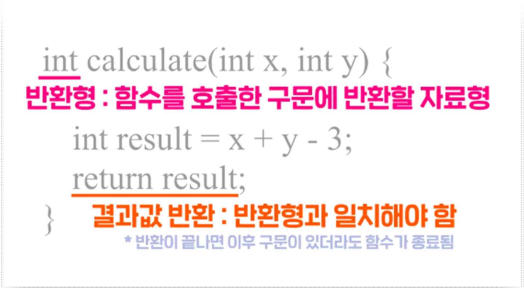
배열은 순서에 맞춰 값을 담고 순서에 해당하는 값을 바로 꺼내써야 하는 목록형 데이터에 사용

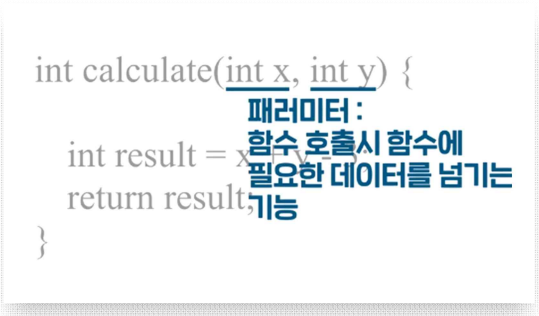
② 클래스

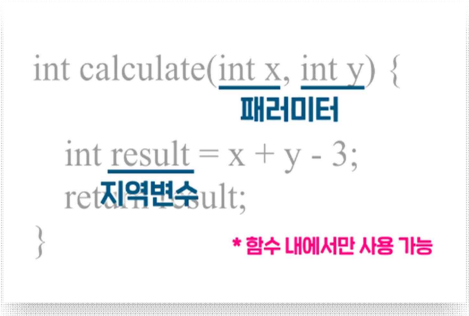
▪ **함수Funtion** 입력값을 넣으면 함수 내부에서 처리된 결과가 출력되는 구문

예) 두 개의 인티저 정수 값을 받아 두 수를 더한 값에서 3을 뺀 값을 호출한 뒤에 반환하는 함수









* 패러미터와 지역변수는 함수 바깥의 변수와 이름이 같더라도 서로 같은 변수가 아님

```
int z = calculate(6, 7);
```

calculate 함수에 6과 7을 패러미터로 넘기면 6+7-3 = 10이 반환됨
따라서 z에는 결과적으로 10이 할당됨

■ **클래스 = 쿠키커터와 같은 역할**

쿠키커터는 필요에 따라 달, 별, 사람으로 만들어 찍어낼 수 있음
클래스도 프로그램의 필요에 따라 원하는 기능을 가진 객체를 생성해 내는 데에 사용함

■ **객체를 생성한다는 것**

클래스라는 틀이 가진 기능과 객체 자신만의 속성을 가진 프로그래밍 요소

하트 모양의 커터를 클래스라고 하면 그것으로 찍어낸 개체가 하트모양의 쿠키라는 기본 기능은 같더라도 서로 다른 재료나 모양의 쿠키, 즉 속성이 다른 객체를 만들 수 있다는 의미

■ **클래스에 포함되는 요소**

함수(메서드) : 모든 객체에 공통적인 기능을 부여함
변수(필드) : 객체별 개별적인 속성/ 상태를 나타낼 수 있음

기본기 학습 - 클래스 패키지

✔ 클래스는 어떻게 만들까?

```
class MyClass {
    private int count;
    public boolean isActivated;
    public int addCount() {
        if(isActivated) count++;
    }
}
```

① class 뒤에 클래스 명을 기술

② 중괄호 안에 클래스에 필요한 함수와 변수를 구현

③ 필요에 따라 생성자constructor라는 특수한 함수를 만들어 줌

✔ 생성자

클래스를 통해 new 연산자로 객체를 생성할 때 처음 한번만 호출되는 함수

```
MyClass object = new MyClass();
                 이것이 생성자!
```

객체의 초기상태를 만드는 명령 구문을 실행함

```
public class MyClass {
    private int count;
    public boolean isActivated;
    public MyClass(boolean isActivated)
    {
        this.isActivated = isActivated;
    }                  이것이 생성자를 구현한 모습
    public int addCount() {
        if(isActivated) count++;
    }
}
```

← 반환형이 없음
클래스의 이름과 동일하게 만들어줘야 함
필요에 따라 파라미터를 받아 초기화 함

✔ 자바는 객체지향 언어

클래스로 생성한 객체들의 집합으로 프로그램을 만들어 냄
하나의 객체가 다른 객체들을 만들어내고 그 객체가 가진 함수나 변수를 사용하여 더 큰 기능을 가능하게 하는 프로그램을 완성하는 구조

그런데 클래스가 모든 함수와 변수를 사용할 수 있게 열어둔다면?
외부 클래스가 바꾸지 말아야 할 변수나 내부에서만 사용하는 함수까지 사용되면 프로그램 오류 발생

✔ 접근제한자 access modifier

public 모든 접근을 허용

private 내부에서만 접근

protected 내부, 같은 패키지 상속받은 클래스에서만 접근

(default) 내부, 같은 패키지에서만 접근

*클래스, 변수, 함수의 맨 앞에 붙여 사용함

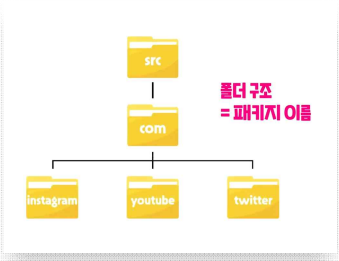
*변수와 함수에는 모든 접근자 사용이 가능하나 클래스에는 public과 기본 상태만 가능

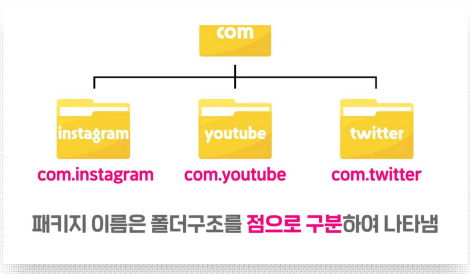
```
public class MyClass {
    private int count;
    public boolean isActivated;
    public MyClass(boolean isActivated)
    {
        this.isActivated = isActivated;
    }
    public int addCount() {
        if(isActivated) count++;
    }
}
```

클래스, 변수, 함수의 맨 앞에 붙여 사용함

✔ 패키지

클래스를 묶어 배포하는 단위





*1개의 클래스는 파일 하나로 이루어짐

*다른 패키지의 클래스를 사용할 때는 소스코드 최상단에 패키지 명을 import 해주면 됨

```
import com.youtube;

public class MyClass {
    private int count;
    public boolean isActivated;
```

*패키지 간에는 접근제한자로 사용을 제한할 수도 있음

✔ 상속

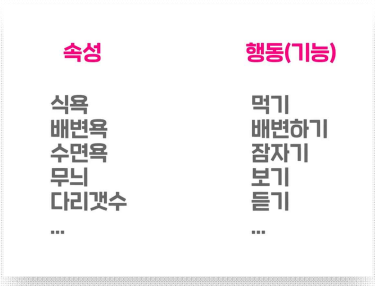
클래스의 변수와 함수들을 다른 클래스에게 물려주는 기능

슈퍼클래스 → **서브클래스**

멤버를 물려준 클래스 물려받은 클래스

```
public class MyClass extends SuperClass {
    private int count;
    public boolean isActivated;
    public MyClass(boolean isActivated)
    {
        this.isActivated = isActivated;
    }
    public int addCount() {
        if(isActivated) count++;
    }
}
```

클래스 명 뒤에 상속받을 수퍼클래스를 extends와 함께 기술



예시

동물의 기본적인 속성과 행동

*이런 공통적인 부분을 동물이라는 수퍼클래스로 만들어 두고 세부 동물의 특징적인 부분들만 각각의 서브클래스로 구현한다면 공통부분은 수퍼클래스에서 한번에 추가나 수정을 할 수 있고 다른 부분은 수퍼클래스를 상속받아 언제나 새로 만들 수 있다.

*협업에도 도움이 되는 점은 뼈대가 되는 부분만 수퍼클래스로 만들면 다른 사람들이 상속을 받아 살을 붙여 서브클래스를 만들 수 있음.

#3 코틀린의 형변환과 배열

✔ 형변환 type casting

하나의 변수에 지정된 자료형을 호환되는 다른 자료형으로 변경하는 기능

*논리형은 변환할 수 없음

```
var a: Int = 54321
var b: Long = a.toLong()
```

기본 자료형들은 자료형 간의 형변환을 지원하기 위해 형변환 함수를 제공하고 있다
코틀린은 명시적 형변환만 지원하고 암시적 형변환은 지원하지 않음.

명시적 형변환 변환될 자료형을 개발자가 직접 지정함
암시적 형변환 변수를 할당할 시 자료형을 지정하지 않아도 자동으로 형변환 됨

✔ 배열만들기


```
var intArr = arrayOf(1, 2, 3, 4, 5)
```

배열로 사용할 변수를 만들어 주고 **arrayOf 함수**를 통해 배열에 저장할 값들을 나열하면 됨

특정한 크기의 공간을 가지는 비어있는 배열을 만들고 싶으면 **arrayOfNulls 함수**에 크기를 지정하면 null로 채워진 배열이 만들어짐.

```
var nullArr = arrayOfNulls<Int>(5)
```

이때 arrayOfNulls 함수에는 꺾쇠가 들어가는데 꺾쇠 안에는 배열에 할당할 자료형을 지정해주면 된다. 이를 **Generic**이라고 한다.

```
intArr[2]
```

배열에 값을 할당하거나 사용하려면 배열 이름 뒤에 대괄호를 쓰고 그 안에 참조할 index를 쓰면 된다.

`intArr[2] = 8` 변수처럼 해당 index 위치에 값을 할당할 수도 있고

`println(intArr[4])` 할당된 값을 사용할 수도 있다.

배열은 처음 선언했을 때의 전체 크기를 변경할 수 없다는 단점이 있지만 한 번 선언을 해두면 다른 자료구조보다 빠른 입출력이 가능함