인스턴스와 static

프로그램

- *컴퓨터가 수행할 명령의 집합*
- **현실의 문제를 해결하기 위한 목적**으로 컴퓨터에게 일을 시키는 것

객체

- 현실의 개념을 **추상화** 하여 프로그램 상에서 다룰 수 있도록 나타낸 것
- `class`로 **상태**, **행위**를 정의한다.

추상화

- 현실의 개념은 아주 복잡한 성질을 가지고 있다.
- 여러 성질 중 내 프로그램에서 필요한 성질만 추려내는 것

ex) 학사관리시스템

학생 : 나이, 키, **이름**, 피부색, 모발 개수, **전공**, **학번**, ...

Class

- 객체를 정의하기 위한 문법

```
class Student {
        // 상태 : 멤버변수
        int no;
        String name;
        String major;
```

```
        // 행위 : 메서드
        void listenLecture() {
                // 강의를 듣는다
        }

        String writeReport() {
                // 논문을 쓴다
                String report = "논문";
                return report;
        }
}
```

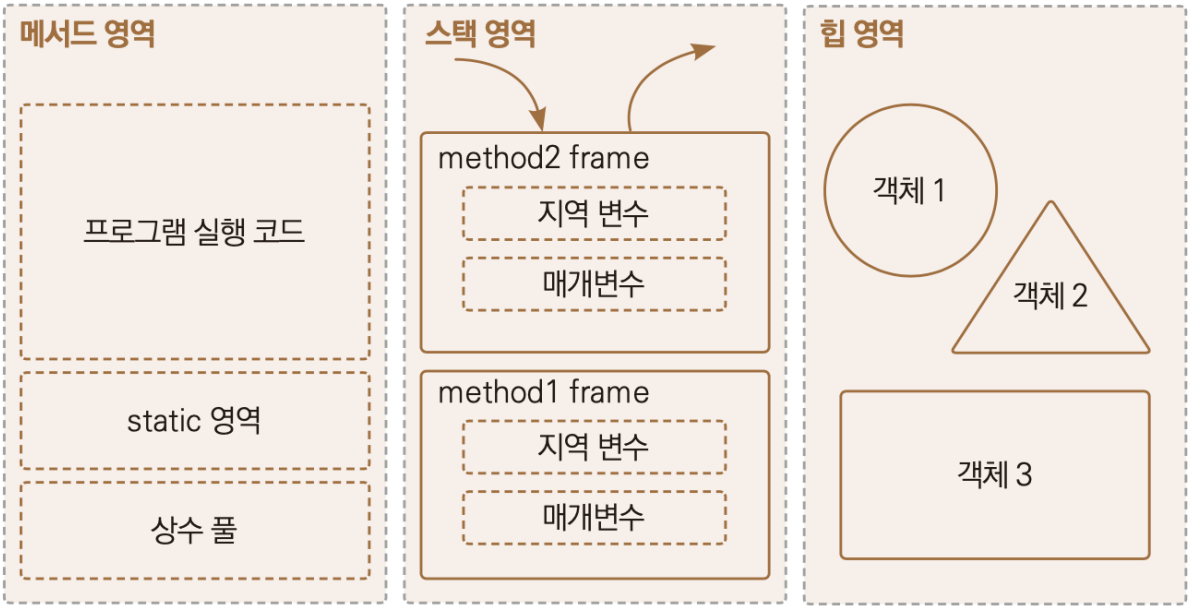
인스턴스

- 정의된 `class`로부터 만들어진 객체
- 만들어진 객체는 `JVM Heap`영역에 할당된다.

class : 붕어빵 틀 → instance : 붕어빵

- 슈크림 붕어빵
- 팥 붕어빵
- 피자 붕어빵

▼ 런타임 데이터 영역들



프로그램에서 `동적` 과 `정적` 의미

동적

- 프로그램이 실행되는 시점에 바뀔 수 있는 것
- runtime

정적

- 프로그램이 실행되기 전 결정되는 것
- compiletime

static

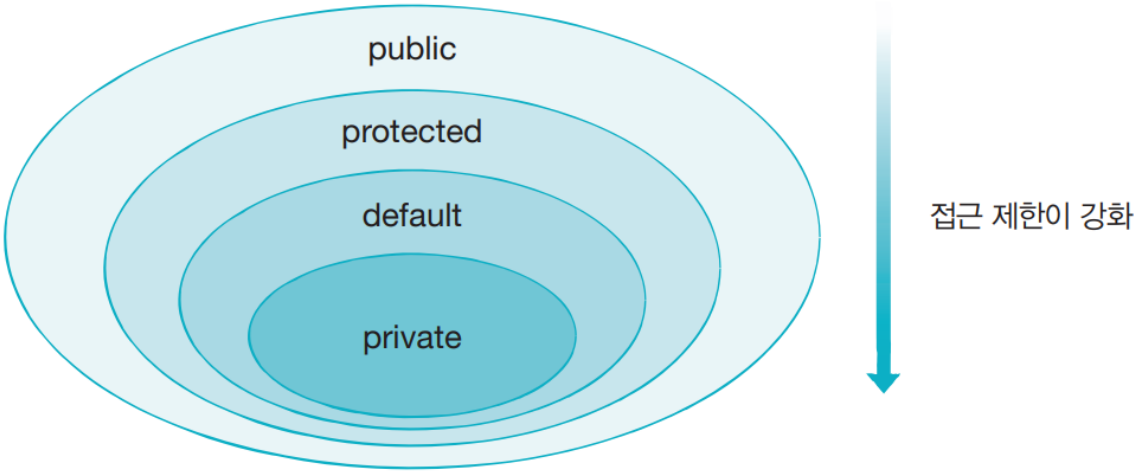
- `정적인` 이라는 의미
- static 키워드가 붙으면 compile타임에 `한 번` 메모리에 할당되어 프로그램이 종료될 때까지 유지된다.

```
# 각각의 인스턴스는 동적으로 만들어져서 별개의 heap 메모리를 할당 받는데,

# static 변수는 인스턴스가 만들어지기 전에도 메모리 공간이 할당되므로
```

```
인스턴스 없이 접근가능하다.

# static 메서드 역시 인스턴스 없이 호출가능하다.
```

접근제한자(접근지정자)

- 객체의 `멤버변수`, `메서드`에 접근할 수 있는 범위를 제한하는 키워드



디자인 패턴

- 선배 개발자들이 자주 맞닥뜨린 문제의 해결방법을 정리해서 패턴화해 놓은 것

싱글턴 (일종의 디자인 패턴)

- 프로그램 실행 중 한 타입의 인스턴스를 하나만 만들 수 있도록 하는 디자인 패턴

- 생성자 private
- 나 자신의 인스턴스를 관리하고 있는 멤버변수
- 외부에서 인스턴스를 접근할 수 있도록 하는 메소드