Static

인스턴스멤버와 정적멤버

멤버(member)

- 멤버(member)는 영어로 "구성원" 이라는 뜻이다.
- 객체에도 구성원이 있는데 이를 멤버라고 한다.
  - 변수(필드)
  - 메소드
- 이런 멤버들의 종류가 인스턴스 멤버, 정적 멤버로 구분된다.

인스턴스 변수

- 인스턴스 변수는 클래스에서 일반적으로 선언한 변수(필드)
- 인스턴스 변수는 프로그램 실행 시 new 라는 키워드와 함께 클래스가 객체화(인스턴스화)**될 때**, 메모리영역을 차지하게 된다.

```java
class Car {
    String model;
    int maxSpeed;
}

public class Main {
    public static void main(String args[]) {
        // 여러 실행문~~~
        Car car = new Car();
    }
}
```

new 키워드가 있는 라인이 **실행될 때** 메모리 영역을 차지한다!!

정적 변수 (클래스 변수)

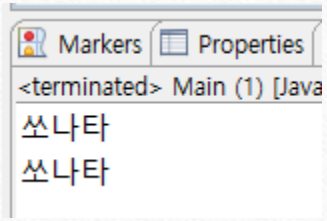
- 정적 변수 또는 클래스 변수 라고 한다.
- 정적 변수(클래스 변수)는 소스코드가 **컴파일 되는 시점에 바로** 클래스 로더에 의해서 메모리를 차지하게 된다.
- 요약
  - 인스턴스 변수: **객체화(인스턴스화) 될 때**, 메모리에 적재
  - 정적 변수: **컴파일 시점에 바로** 메모리에 적재
- 즉, **타이밍이 다르다!**

정적 변수 (클래스 변수)

- `static` 이라는 키워드를 앞에 추가한다.
- 일반적인 멤버 변수를 사용하듯이 **변수 이름**으로 사용한다.

```
class Car {
    static String model;
    int maxSpeed;
}

public class Main {
    public static void main(String args[]) {
        Car car = new Car();
        Car car2 = new Car();
        car.model = "쏘나타";
        System.out.println(car.model);
        System.out.println(car2.model);
    }
}
```



분명 car 객체의 멤버변수에 "쏘나타" 를 저장했는데 car2 역시 "쏘나타" 가 출력된다!! 이유는??

정적 변수 (클래스 변수)

- 정적 변수는 객체간에 데이터를 공유한다.
- 따라서, 객체(인스턴스)에 소속된 변수 라기 보다, **클래스 자체에 소속된 변수**라고 볼 수 있다! (그래서 클래스 변수)
- 그러므로, 정적 변수는 객체가 사라지더라도 **시스템이 종료되기 전까지 계속 메모리영역을 차지**하기 때문에 메모리에 고정됨

  (값이 고정되어 변하지 않는다는 말이 아니라 메모리에 고정됨)
- 일반적으로 객체 생성없이 바로 **클래스이름.변수명** 으로 접근!

정적 메소드

- 메소드도 멤버 중 하나 이기 때문에 정적 메소드 선언 가능
- 정적 메소드 또한 앞에 `static` 이라는 키워드 추가!
- 정적 메소드도 메모리영역을 차지하는 타이밍이 일반적인 인스턴스 메소드와 다르다! (더 먼저 메모리에 적재된다.)
- 따라서, 정적 멤버와 마찬가지로 객체 생성없이 바로

  클래스이름.메소드명() 으로 접근하여 사용!

정적 메소드

- 정적 메소드 내에서 this 사용불가!
- 정적 메소드 내에서 인스턴스 멤버 접근 불가!

(메모리 적재 타이밍이 다름!)

```
class Car {
    static String model;
    int maxSpeed;

    static void print(String model) {
        System.out.println( model );
        System.out.println( this.model );
        System.out.println( Car.model);
        System.out.println( maxSpeed );
    }
}
```

(번외) 싱글톤

- 전체 프로그램 내에서 **단 하나의 객체만 생성**되도록 해야하는 경우가 있을 수 있다.
- 하나의 객체만 생성되도록 클래스를 설계한 디자인 패턴을 **싱글톤**(singleton)이라고 한다.

```
class Car {
    private static Car singleton = new Car();
    private Car() {}
    static Car getInstance() {
        return singleton;
    }
}

public class Main {
    public static void main(String args[]) {
        Car car = Car.getInstance();
    }
}
```