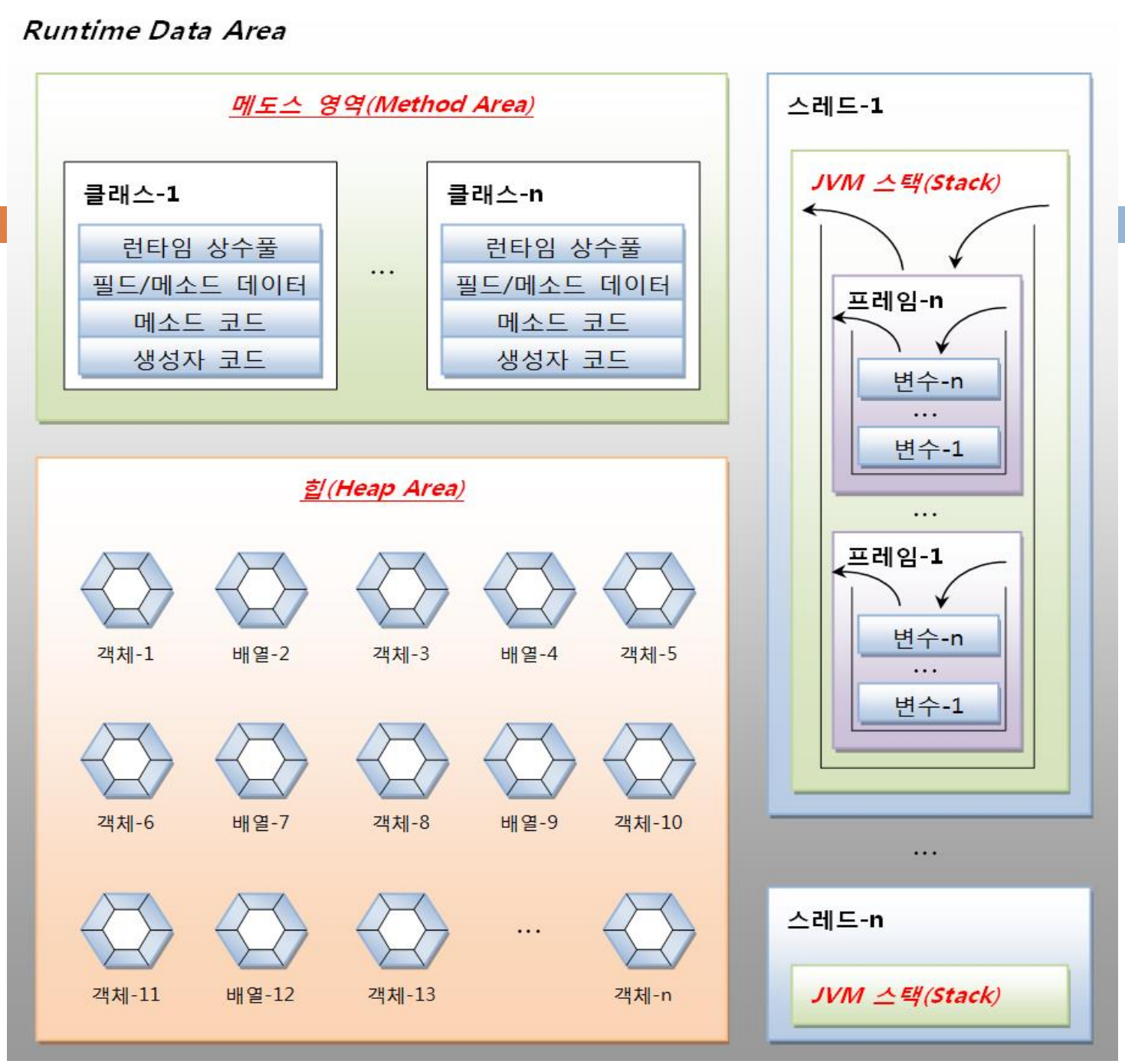
JVM 메모리 구조

- method area : JVM이 .class파일을 읽어 클래스별로 필드데이터, 메소드 데이터, 메소드 코드, 생성자 코드 등을 분류해 저장
  - 전역변수, static 붙은 메소드, 클래스 변수(static 붙은 변수) 저장
  - 클래스가 로딩될 때 생성되고, 모든 스레드가 공유하는 영역
  - 클래스가 실제로 호출될때 method area에 올라간다.
  - 프로그램의 시작부터 종료가 될 때까지 메모리에 남아있게 된다.
- call stack : 메소드를 위한 작업공간
  - main이 젤 먼저 수행되어 밑에 존재
  - 메소드가 실행되는 동안 객체 참조변수, 지역변수와 연산의 중간결과를 저장
  - 메소드가 끝나면 메모리 반환됨.
- Heap : new 연산자로 생성된 객체와 배열 저장
  - 클래스 영역에 선언된 인스턴스 변수들 저장
  - 자바에서는 garbage collector가 관리함.

Runtime Data Area



정적(static) vs. 비정적(non-static)

- 다른 클래스에서 정의된 메인 메소드에서
  - 비정적 소속변수나 비정적 소속메소드를 호출하기 위해서는 우선 객체가 생 성되어 있어야 한다
  - 정적 소속변수에 접근하거나 정적 소속메소드를 호출하는 경우에는 객체를 생성하지 않아도 됨.
  - 호출 방식
    - 비정적 소속변수: 객체이름 . 소속변수이름
      - Car c = new Car();
      - String str = c.color ;
    - 정적 소속변수: 클래스이름 . 소속변수이름
      - int n = Car.numberOfCars ;
    - 비정적 소속메소드 : 객체이름 . 메소드이름 (…)
      - Car c = new Car();
      - String str = c.toString();
    - 정적 소속메소드 : 클래스이름 . 메소드이름 (...)
      - Car.increase();

정적(static) vs. 비정적(non-static)

- 같은 클래스 내에서
  - 비정적 메소드
    - 같은 클래스 내부에 정의된 비정적/정적 소속변수 모두 참조 가능
    - 같은 클래스 내부에 정의된 비정적/정적 소속메소드 모두 호출 가능
    - 호출 시에는 메소드 이름만을 사용
  - 정적 메소드
    - 같은 클래스 내부에 정의된 **정적 소속변수만** 참조 가능
    - 같은 클래스 내부에 정의된 **정적 소속메소드만** 호출 가능
    - 호출 시에는 메소드 이름만을 사용

- 오류가 발생하는 이유는?

```
public class Test  {
    public static void main(String[] args)  {
        int tmpR = add(10, 20);  // 오류!!
        System.out.println(tmpR);
    }

    public int add(int x, int y)  {
        return (x + y);
    }
}
```

```java
public class Util {

    private int utilID;
    public static int fps = 20;

    public Util(int utilID) {
        this.utilID = utilID;
    }

    public static void setFps(int fps) {
        Util.fps = fps;
        utilID = 1;
    }
```

인스턴스 변수에 접근 불가