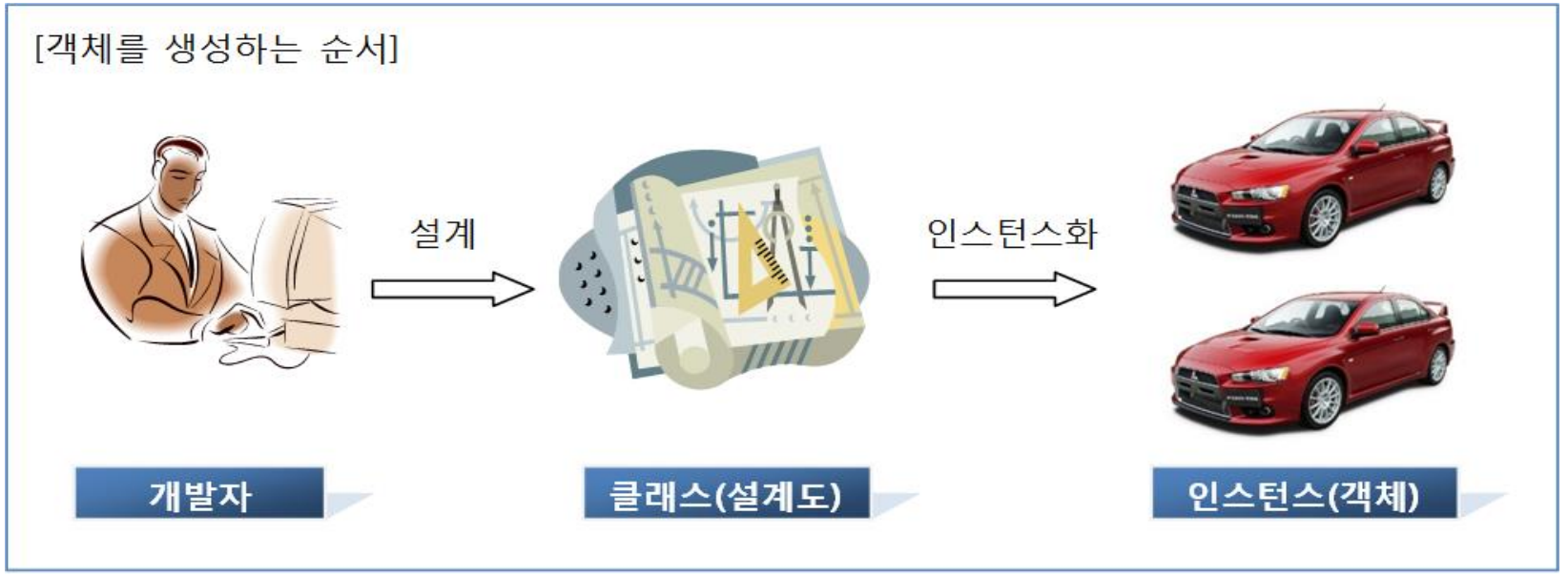
객체와 클래스

객체와 클래스

❖ 객체(Object)와 클래스(Class)

- 현실세계: 설계도 → 객체
- 자바: 클래스 → 객체

- 클래스에는 객체를 생성하기 위한 필드와 메소드가 정의
- 클래스로부터 만들어진 객체를 해당 클래스의 인스턴스(instance)
- 하나의 클래스로부터 여러 개의 인스턴스를 만들 수 있음



클래스 선언

❖ 클래스의 이름

- 자바 식별자 작성 규칙에 따라야

번호	작성 규칙	예
1	하나 이상의 문자로 이루어져야 한다.	Car, SportsCar
2	첫 번째 글자는 숫자가 올 수 없다.	Car, 3Car(x)
3	'$', '_' 외의 특수 문자는 사용할 수 없다.	$Car, _Car, @Car(x), #Car(x)
4	자바 키워드는 사용할 수 없다.	int(x), for(x)

- 한글 이름도 가능하나, 영어 이름으로 작성
- 알파벳 대소문자는 서로 다른 문자로 인식
- 첫 글자와 연결된 다른 단어의 첫 글자는 대문자로 작성하는 것이 관례

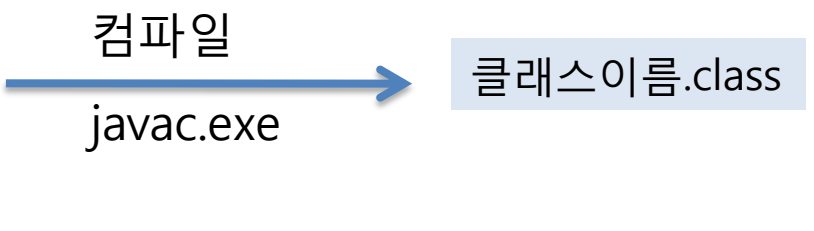
클래스 선언

❖ 클래스 선언과 컴파일

- 소스 파일 생성: 클래스이름.java (대소문자 주의)
- 소스 작성

```
public class 클래스이름 {


}
```



클래스 선언

❖ 클래스 선언과 컴파일

- 소스 파일당 하나의 클래스를 선언하는 것이 관례
  - 두 개 이상의 클래스도 선언 가능
  - 소스 파일 이름과 동일한 클래스만 public으로 선언 가능
  - 선언한 개수만큼 바이트 코드 파일이 생성

Car.java

```
public class Car {

}

class Tire {

}
```

Car.class

Tire.class

객체 생성과 클래스 변수

❖ new 연산자

- ○ 객체 생성 역할

```
new 클래스();
```

  - ▪ 클래스()는 생성자를 호출하는 코드
  - ▪ 생성된 객체는 힙 메모리 영역에 생성

- ○ new 연산자는 객체를 생성 후, 객체 생성 번지 리턴

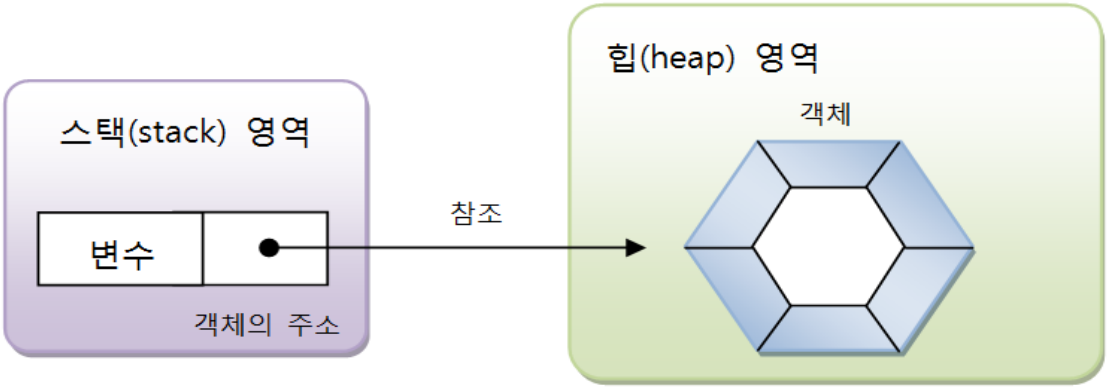
객체 생성과 클래스 변수

❖ 클래스 변수

- new 연산자에 의해 리턴 된 객체의 번지 저장 (참조 타입 변수)
- 힙 영역의 객체를 사용하기 위해 사용

```
클래스 변수;
변수 = new 클래스();

클래스 변수 = new 클래스();
```

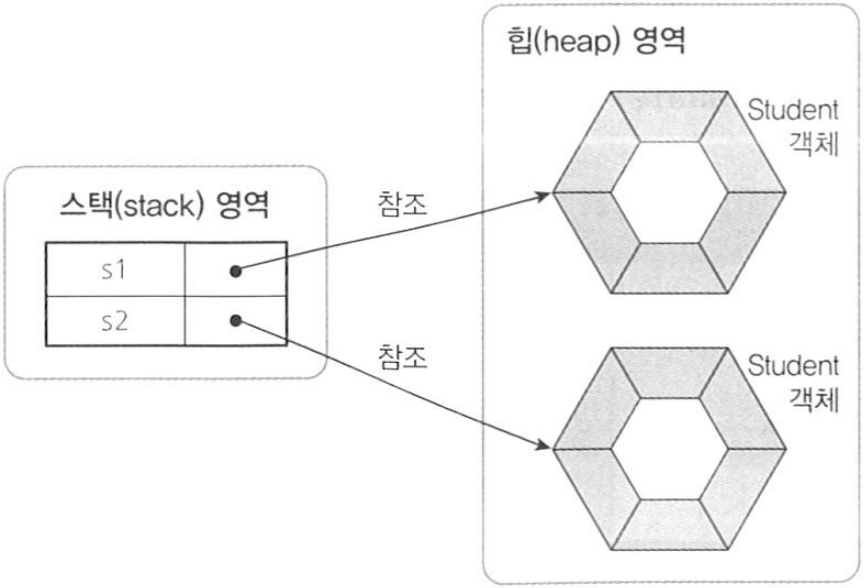


객체 생성과 클래스 변수

❖ 클래스 선언: Student.java

```
public class Student {

}
```



❖ 클래스로부터 객체 생성: StudentExample.java

```
public class StudentExample {
    public static void main(String[] args) {
        Student s1 = new Student();
        System.out.println("s1 변수가 Student 객체를 참조합니다.");

        Student s2 = new Student();
        System.out.println("s2 변수가 또 다른 Student 객체를 참조합니다.");
    }
}
```

❖ 클래스의 용도

- 라이브러리(API: Application Program Interface) 용
  - 자체적으로 실행되지 않음
  - 다른 클래스에서 이용할 목적으로 만든 클래스

- 실행용
  - main() 메소드를 가지고 있는 클래스로 실행할 목적으로 만든 클래스

1개의 애플리케이션 = (1개의 실행클래스) + (n개의 라이브러리 클래스)

❖ 클래스의 구성 멤버

- 필드(Field)
- 생성자(Constructor)
- 메소드(Method)

- **필드(Field)**
  객체의 데이터가 저장되는 곳
- **생성자(Constructor)**
  객체 생성시 초기화 역할 담당
- **메소드(Method)**
  객체의 동작에 해당하는 실행 블록

```
public class ClassName {

    //필드
    int fieldName;

    //생성자
    ClassName() { ... }

    //메소드
    void methodName() { ... }

}
```