

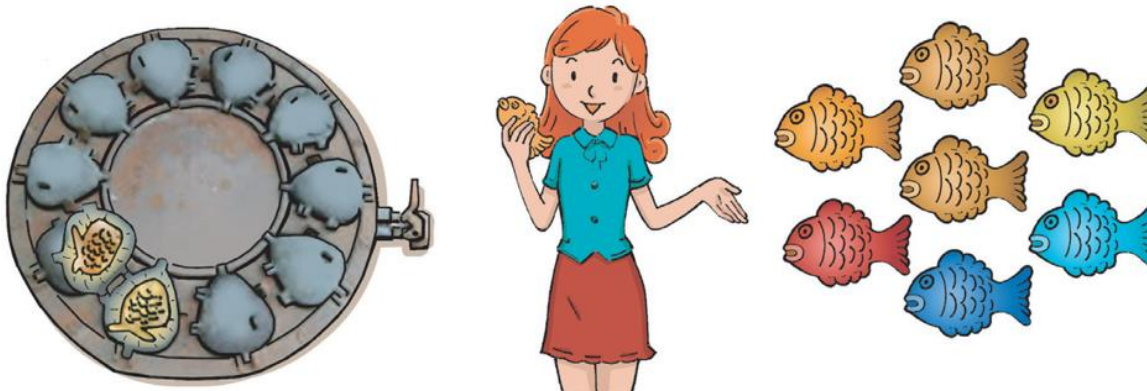
객체 생성과 클래스 변수

▣ 객체와 클래스

- 현실 세계에서 자동차를 생성하려면 자동차의 설계도가 필요하듯이, 객체 지향 프로그래밍에서도 객체를 생성하려면 설계도에 해당하는 클래스(class)가 필요하다.



붕어빵 틀은 클래스이며, 이 틀의 형태로 구워진 붕어빵은 바로 객체입니다. 붕어빵은 틀의 모양대로 만들어지지만 서로 조금씩 다릅니다. 치즈붕어빵, 크림붕어빵, 앙코붕어빵 등이 있습니다. 그래도 이들은 모두 붕어빵입니다.




▣ 클래스 선언

- 클래스 선언은 객체 생성을 위한 설계도를 작성하는 작업이다.
- 어떻게 객체를 생성(생성자)하고, 객체가 가져야 할 데이터(필드)가 무엇이고, 객체의 동작(메소드)은 무엇인지를 정의하는 내용이 포함된다.

[클래스명.java]

```
//클래스 선언  
public class 클래스명 {  
}
```

```
public class SportsCar {
```



클래스 선언

```
}
```

※ 공개 클래스 (public class)

- 하나의 소스 파일에 복수 개의 클래스를 선언할 때 주의할 점은,
소스 파일명과 동일한 클래스만 공개 클래스로 선언할 수 있다는 것이다.
- 복수 개의 클래스 선언이 포함된 소스 파일을 컴파일하면 바이트코드 파일(.class)은 클래스 선언 수만큼 생긴다.

SportsCar.java

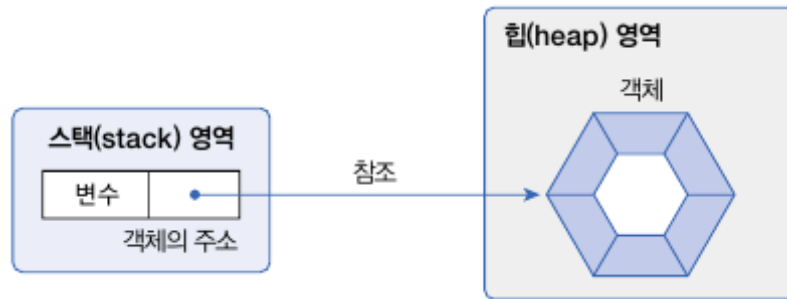
```
public class SportsCar {  
}  
  
class Tire {  
}
```

 SportsCar.class

 Tire.class

▣ 객체 생성과 클래스 변수

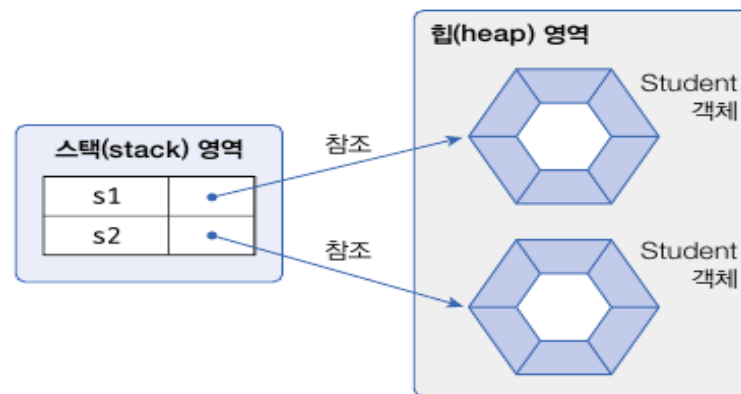
- new 연산자는 객체를 생성시킨 후, 객체의 주소를 리턴하기 때문에 클래스 변수에 대입할 수 있다.



- 객체의 멤버 접근 : 객체 레퍼런스 ● 멤버

```
public class Student {  
}
```

```
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.");  
    }  
}
```



- 라이브러리(library) 클래스 : 실행할 수 없으며, 다른 클래스에서 이용하는 클래스
- 실행 클래스 : main() 메소드를 가지고 있는 실행 가능한 클래스