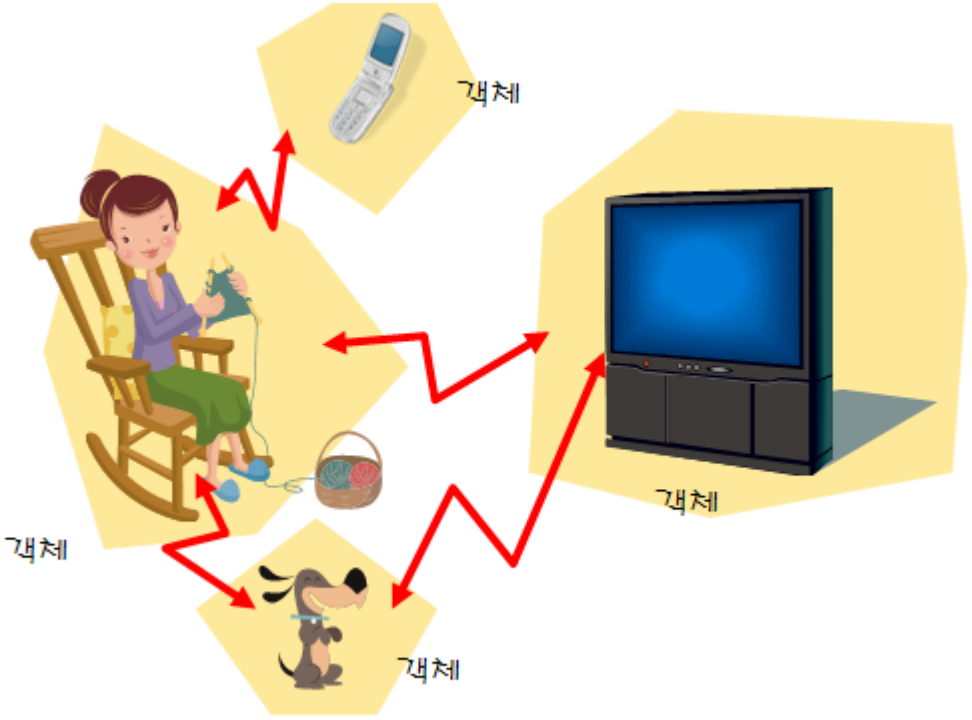
C++ Espresso

# 제4장 객체 지향 소개

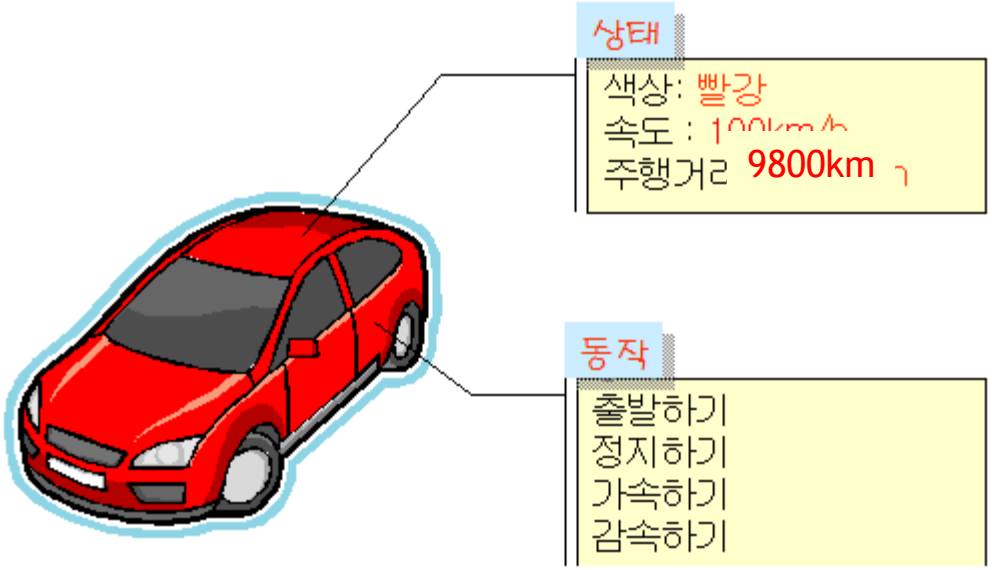


# 객체 지향의 과정

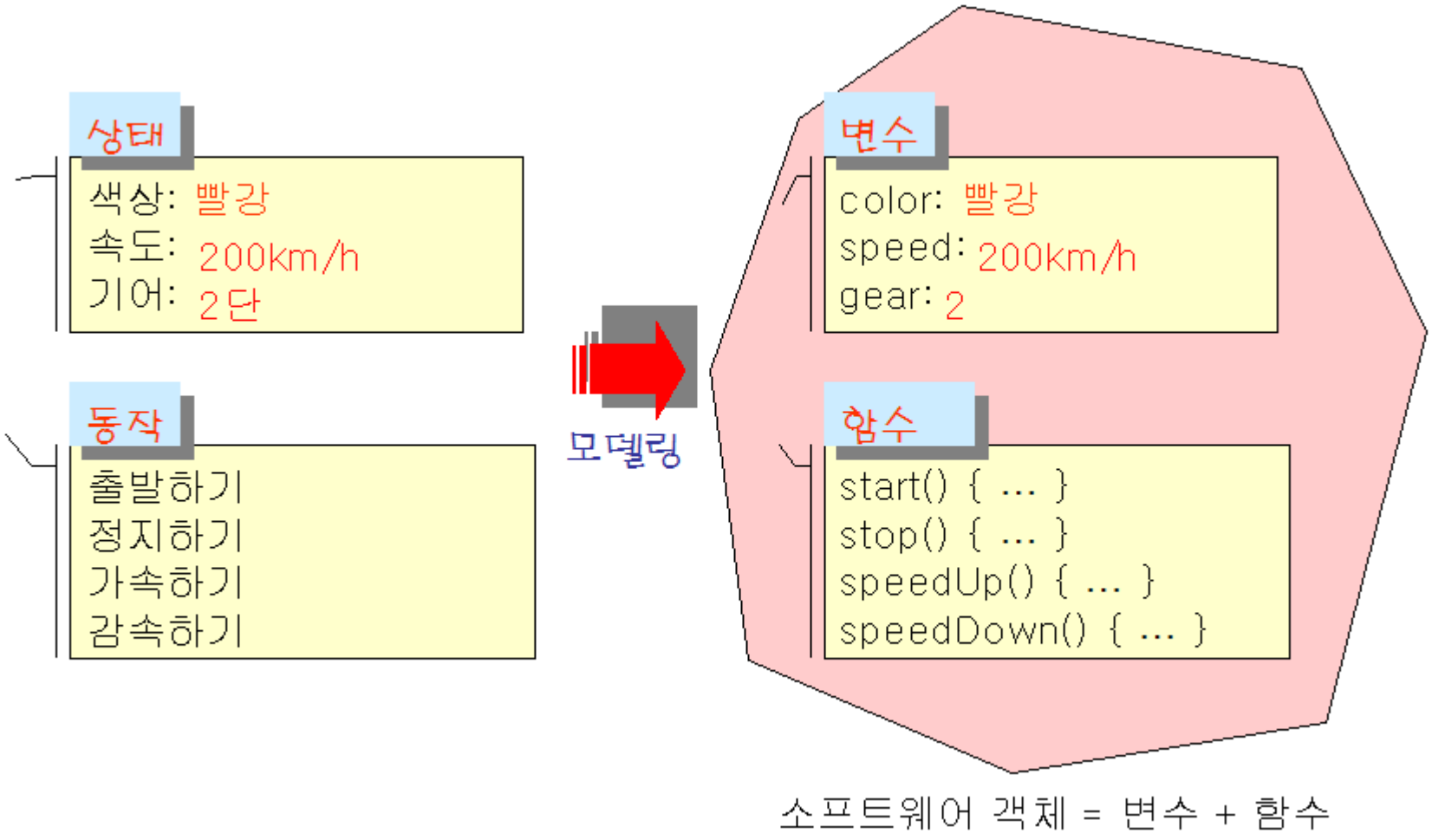
- 실제 세계를 모델링하여 소프트웨어를 개발하는 방법

# 객체란?

- 객체(object)는 상태와 동작을 가지고 있다.
- 객체의 상태(state)는 객체의 특징값(속성)이다.
- 객체의 동작(behavior) 또는 행동은 객체가 취할 수 있는 동작



# 멤버 변수와 멤버 함수




소프트웨어 객체 = 변수 + 함수

# 중간 점검 문제

1. 다음과 같은 실제 세계의 객체에서 가능한 상태와 동작을 정리하여 보자.

| 객체 | 상태 | 동작 |
|---|---|---|
| 라디오 | | |
| 강아지 | | |

# 메시지

- 소프트웨어 객체는 메시지(message)를 통해 다른 소프트웨어 객체와 통신하고 서로 상호 작용한다.

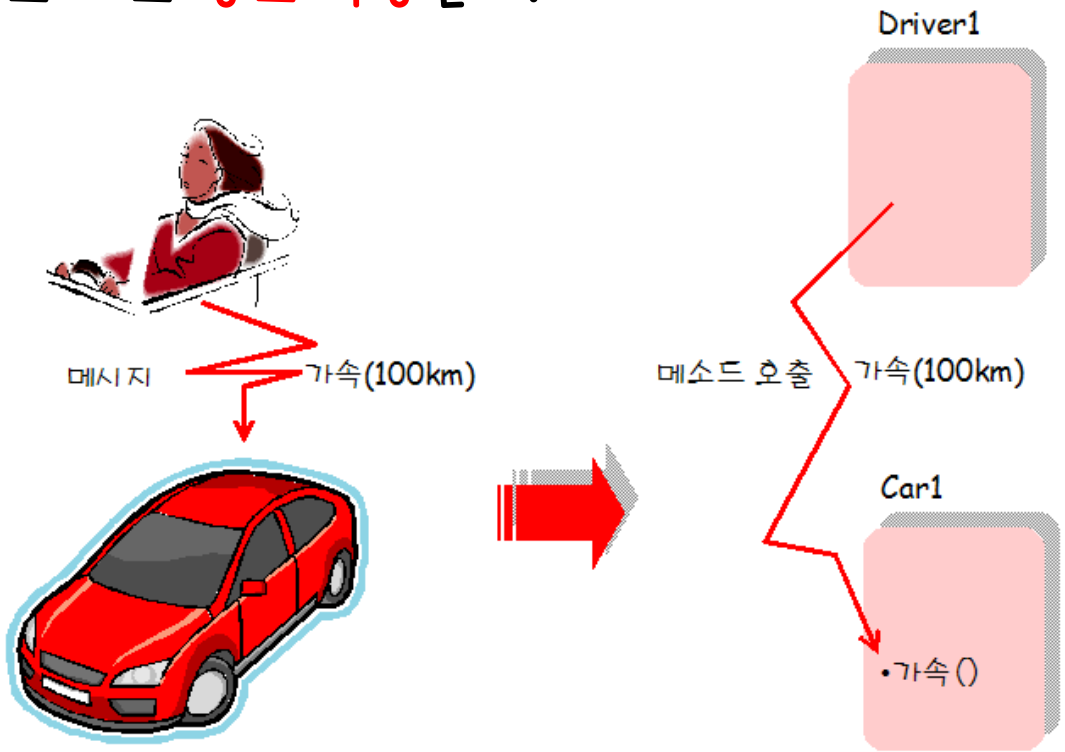



그림 7.5 메시지 전달

# 중간 점검 문제

1. 객체들은 _____전달을 통해서 서로 간에 상호 작용을 한다.

2. 인터넷 객체에서 생각할 수 있는 메시지와 매개 변수에 대하여 나열하여 보라.

   메시지(매개변수)

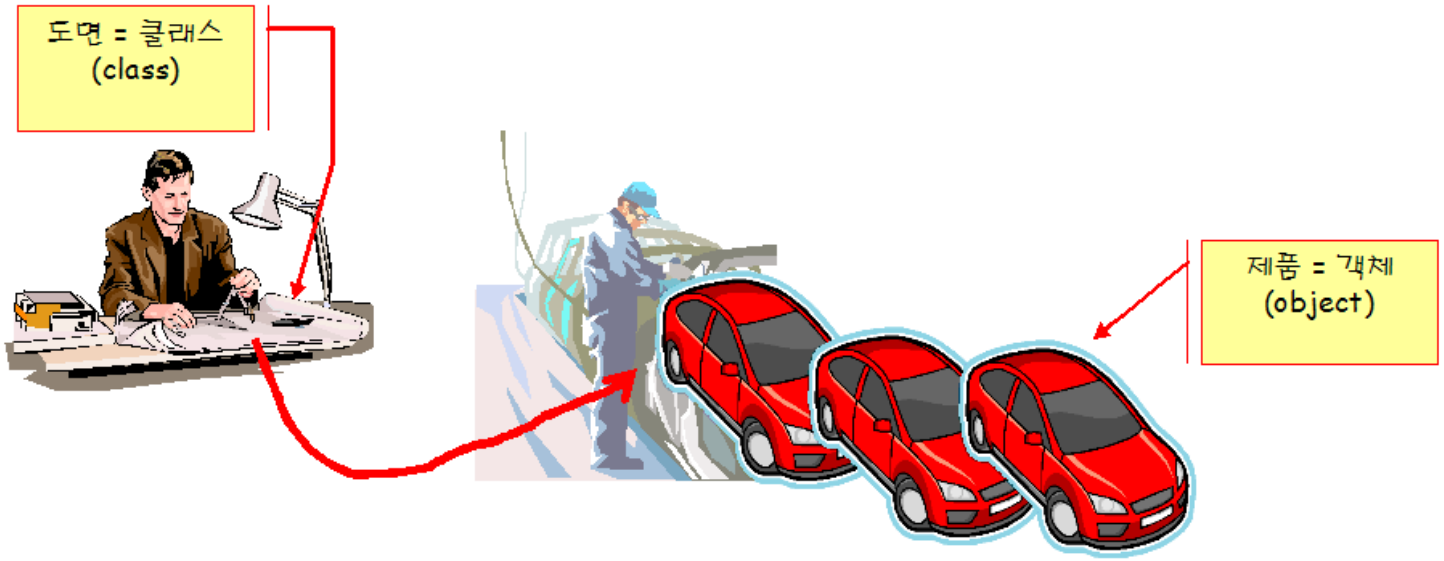
   : 검색하기(　　)

   : 다운받기(　　)

   : 동영상보기(　　)

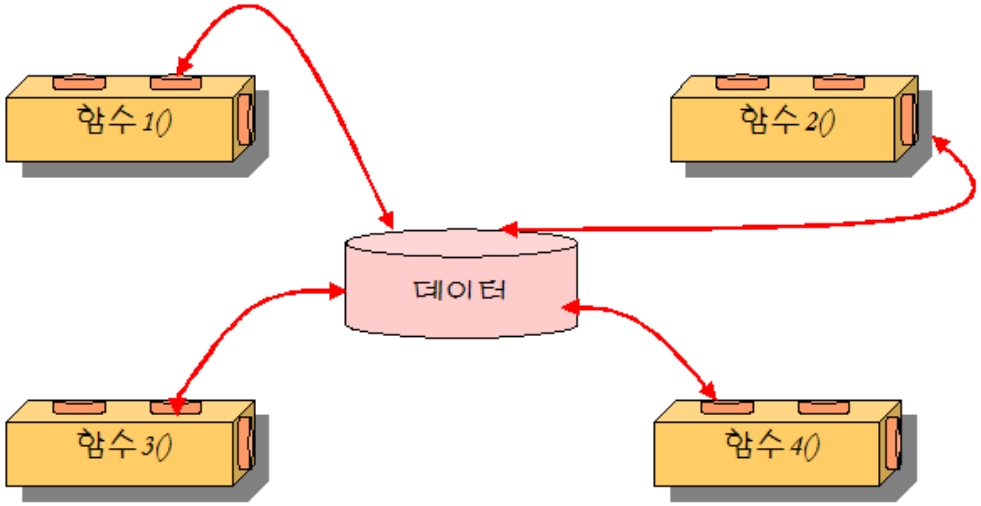
# 클래스

- 클래스(class): 객체를 만드는 설계도
- 클래스로부터 만들어지는 각각의 객체를 특별히 그 클래스의 **인스턴스(instance)**라고도 한다.



# 절차 지향과 객체 지향

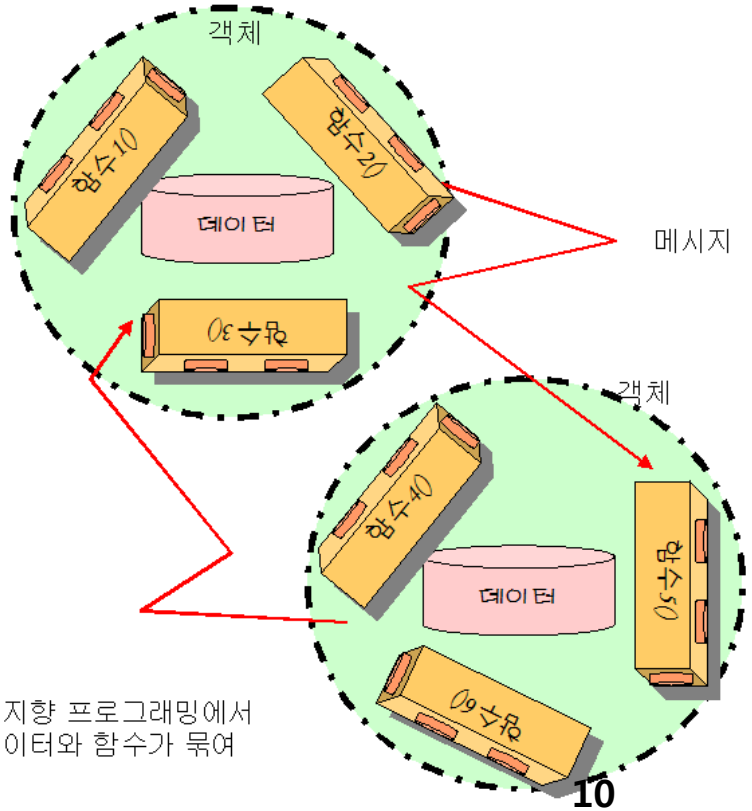
- 절차 지향 프로그래밍(Procedural Programming)
  - 문제를 해결하는 절차를 중요하게 생각하는 소프트웨어 개발 방법. 이들 절차는 모두 함수라는 단위로 묶이게 된다.




절차 지향 프로그래밍에서는 데이터와 함수가 묶여 있지 않다.

# 절차 지향과 객체 지향

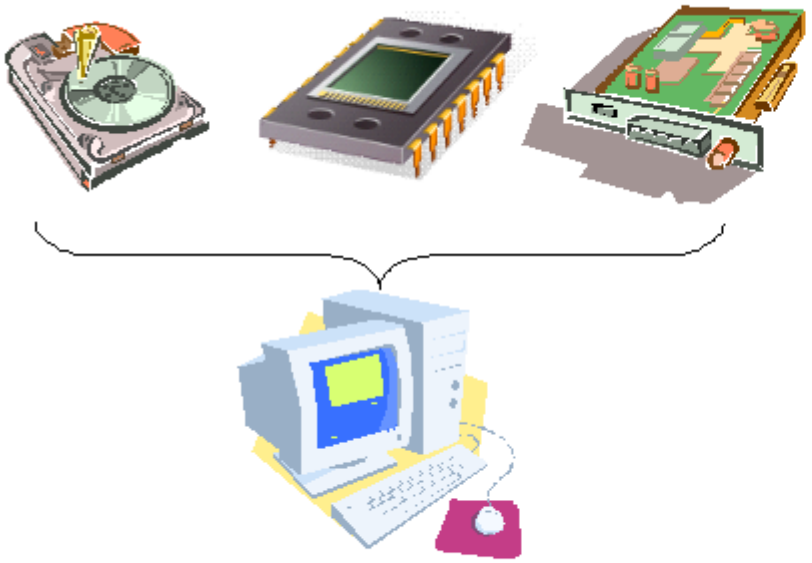
- 객체 지향 프로그래밍(Object-Oriented Programming)
  - 데이터와 함수를 하나의 덩어리로 묶어서 생각하는 방법이다. 데이터와 함수를 객체로 묶는 것을 캡슐화(encapsulation)라고 부른다.

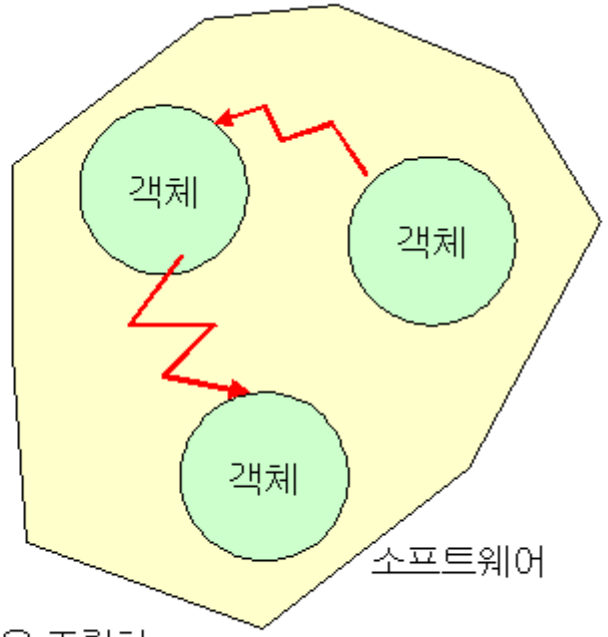



객체 지향 프로그래밍에서는 데이터와 함수가 묶여 있다.

# 객체 지향의 장점

- 객체들을 조립하여서 빠르게 소프트웨어를 만들 수 있다.






부품을 조립하여 제품을 만들듯이 객체들을 조립하여 소프트웨어를 만든다.

# 자동차 경주 게임의 예

절차 지향

```
struct Car {

    int speed;
    int gear;
    char *pcolor;
};

void init(Car& c, char *color);
void start(Car& c);
void stop(Car& c);
int get_speed(Car& c);
void set_speed(Car& c, int speed);


int main()
{
    Car car;
    init(car, "red");
    start(car);
    set_speed(car, 60);
    stop(car);
    return 0;
}
```

객체 지향

```
class Car {
    int speed;
    int gear;
    char *pcolor;

public:
    void init(char *color);
    void start();
    void stop();
    int get_speed();
    void set_speed(int speed);
};

int main()
{
    Car car;
    car.init("red");
    car.start();
    car.set_speed(60);
    car.stop(car);
    return 0;
}
```

# 중간 점검 문제

1. 객체 지향 프로그래밍은 _____ 와 _____를 조합해서 프로그램을 작성하는 기법이다.

2. 클래스로부터 만들어지는 각각의 객체를 특별히 그 클래스의 _____라고도 한다.

# 객체 지향의 개념들

- 캡슐화(encapsulation)
- 정보 은닉(information-hiding)
- 상속(inheritance)
- 다형성(polymorphism)

# 캡슐화

- 캡슐화(encapsulation)란

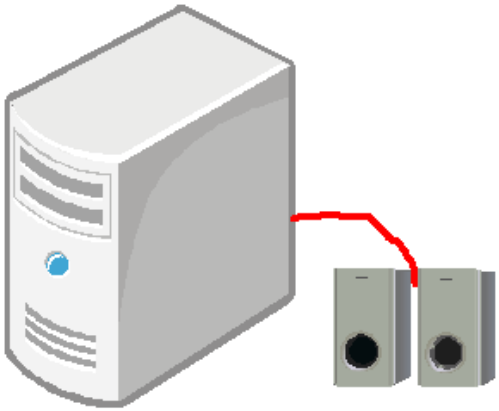
  데이터와 연산들을 객체 안에 넣어서 묶는다는 의미이다.

  따라서 정보를 보호할 수 있다.

# 정보 은닉

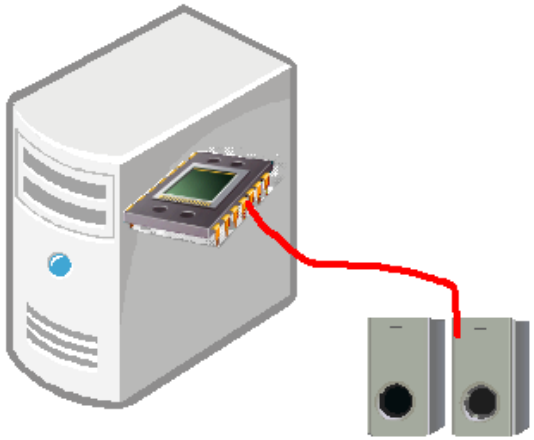
- 객체 내부의 데이터와 구현의 세부 사항을 외부 세계에게 감추는 것.
- 외부 세계에 영향을 끼치지 않으면서 쉽게 객체 내부를 업그레이드할 수 있다.

O



만약 외부의 표준 오디오 단자를 이용하였으면 내부의 사운드 카드를 변경할 수 있다.

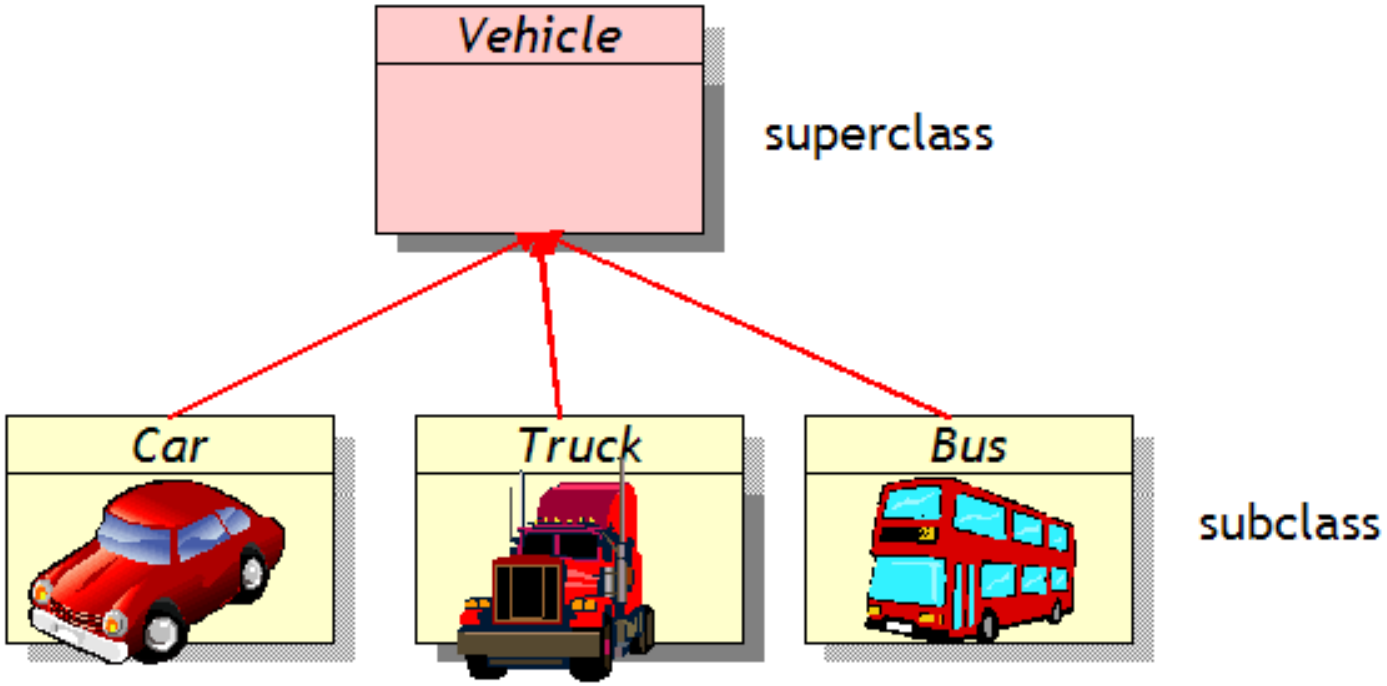
X



만약 내부의 오디오 제어칩의 단자에 연결하였으면 내부의 사운드 카드를 변경할 수 없다.

# 상속

- 상속은 기존의 코드를 재활용하기 위한 기법으로 이미 작성된 클래스(부모 클래스)를 이어받아서 새로운 클래스(자식 클래스)를 생성하는 기법이다.



# 다형성

- 다형성이란 객체가 취하는 동작이 상황에 따라서 달라지는 것을 의미한다. -> 함수 이름의 재사용

# 객체 지향의 장점

- 신뢰성있는 소프트웨어를 쉽게 작성할 수 있다.
- 코드를 재사용하기 쉽다.
- 업그레이드가 쉽다.
- 디버깅이 쉽다.

# 중간 점검 문제

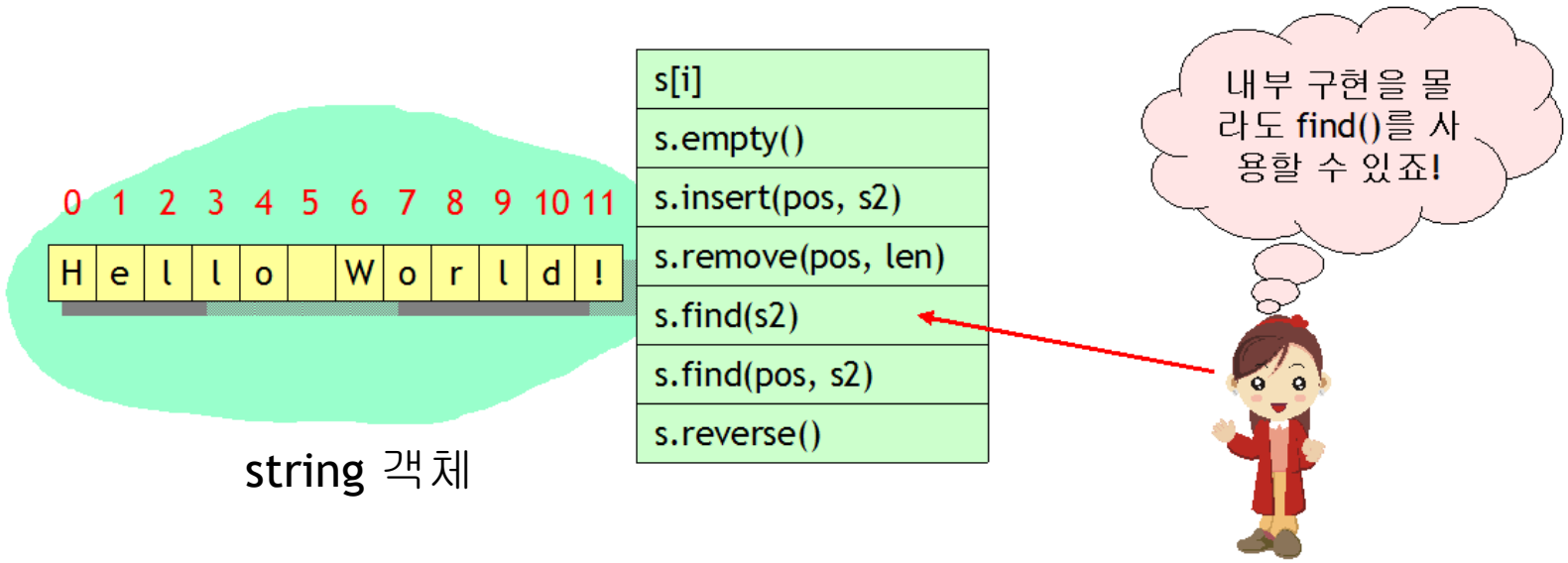
1. 자바에서 코드 재사용이 쉬운 이유는 관련된 _____와 _____가 하나의 덩어리로 묶여 있기 때문이다.

2. 정보 은닉이란 _____을 외부로부터 보호하는 것이다.

3. 정보를 은닉하면 발생하는 장점은 무엇인가?

# string 클래스

- C++에서는 문자열을 나타내는 클래스 string을 제공한다.



# 클래스에서 객체를 생성하는 방법

- string s1;
- string s2 = "Hello World";

# # 실습

# 객체 생성의 예

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string s1 = "This is a test.";    // string 객체를 생성하고 초기화한다.
        string s2;                                  // 비어있는 string 객체를 생성한다.

        cout << s1 << endl;
        cout << "문자열을 입력하시오: " << endl;
        cin >> s2;
        cout << s2 << endl;
        return0;
}
```

```
This is a test.
문자열을 입력하시오: This
This
```

# 실습

# 멤버 함수 호출

- string s1 = "This is a test.";
- int size = s.size();// size는 15가 된다.

# string 클래스의 멤버 함수

| 멤버 함수 | 설명 |
|---|---|
| s[i] | i번째 원소 |
| s.empty() | s가 비어있으면 true 반환 |
| s.insert(pos, s2) | s의 pos 위치에 s2를 삽입 |
| s.remove(pos, len) | s의 pos 위치에 len만큼을 삭제 |
| s.find(s2) | s에서 문자열 s2가 발견되는 첫번째 인덱스를 반환 |
| s.find(pos, s2) | s의 pos 위치부터 문자열 s2가 발견되는 첫번째 인덱스를 반환 |

# 실습

# 멤버 함수 호출의 예

```
#include <iostream>
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string s1 = "This is a test."; // string 객체를 생성하고 초기화한다.

    s1.insert(4, "Hello");
    cout << s1 << endl;
    int index = s1.find("test");
    cout << index << endl;
    s1.append("World");
    cout << s1 << endl;
    return 0;
}
```

```
ThisHello is a test.
15
ThisHello is a test.World
```

# 실습

# 멤버 함수 호출의 예

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string s1("Slow"), s2("steady");
    string s3 = "the race.";
    string s4;

    s4 = s1 + " and " + s2 + " wins " + s3;
    cout << s4 << endl;
    return 0;
}
```

```
Slow and steady wins the race.
계속하려면 아무 키나 누르십시오 . . .
```

# 문자열의 비교

```
string s1("Hello"), s2("World");
if( s1 == s2 )
    cout << "동일한 문자열입니다 " << endl;
else
    cout << "동일한 문자열이 아닙니다 " << endl;
```

== 연산자를 사용하여서 문자열을 비교합니다.

# 중간 점검 문제

1. 객체를 생성하는 방법은 무엇인가?
2. 문자열은 클래스 _____의 객체로 저장할 수 있다.
3. 문자열의 길이를 반환하는 멤버 함수는 _____이다.