

클래스와 객체

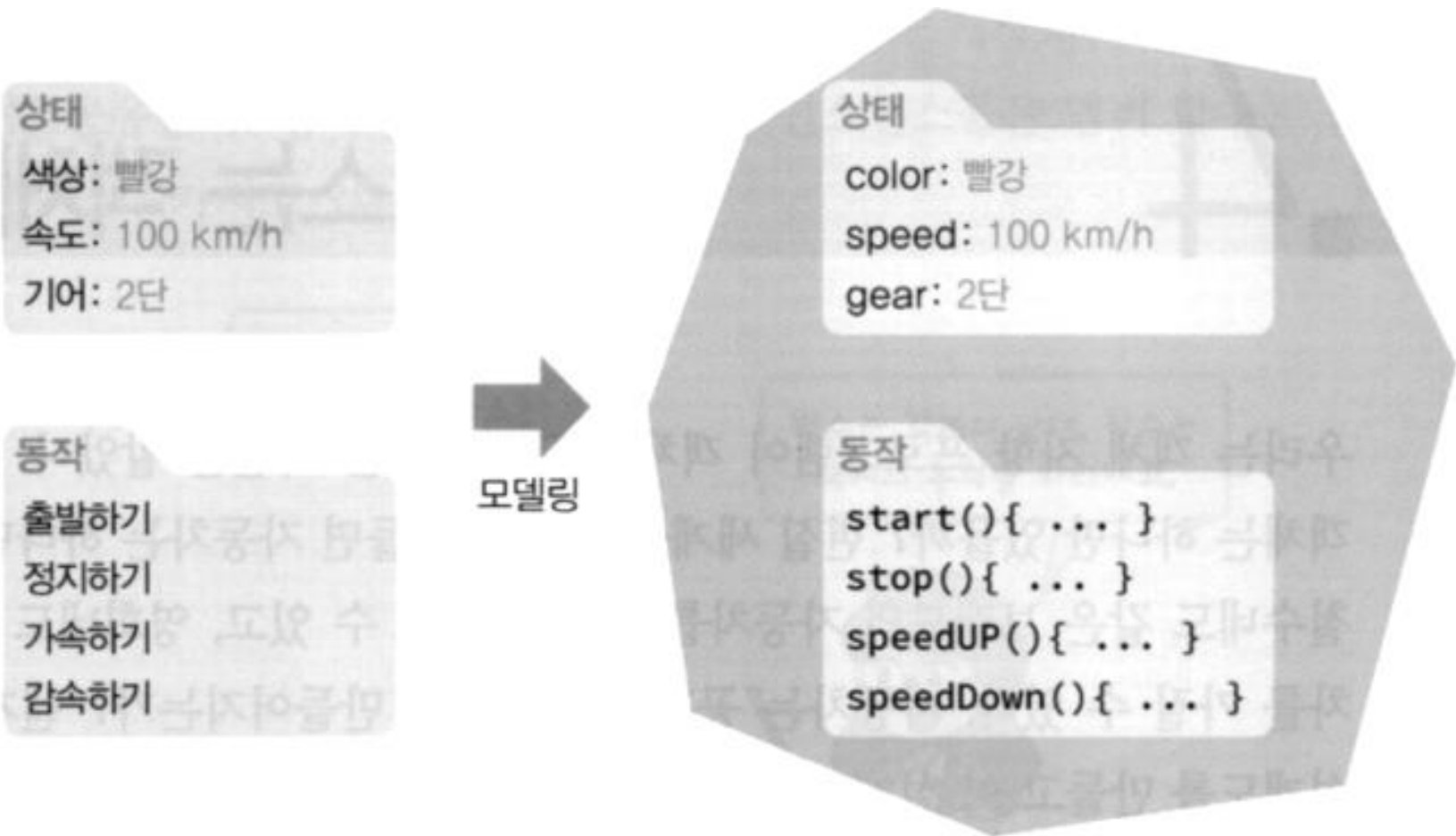
클래스와 객체

■ 객체의 구성요소

- 멤버 변수
 - 객체의 상태를 저장하는 변수
- 멤버 함수(메서드)
 - 객체 안에 정의된 함수
 - 멤버 변수 접근에 자유로움

객체

■ 객체의 구성요소



소프트웨어 객체 = 변수 + 함수

객체

■ 클래스와 객체(인스턴스)

- 클래스
 - 객체의 형태를 정의하는 설계도
- 객체(인스턴스)
 - 클래스의 형태를 취하는 실체

객체

■ 클래스 정의하기

```
class 클래스이름 {
```

```
자료형 멤버변수1;
```

```
자료형 멤버변수2;
```

멤버 변수

```
반환형 멤버함수1();
```

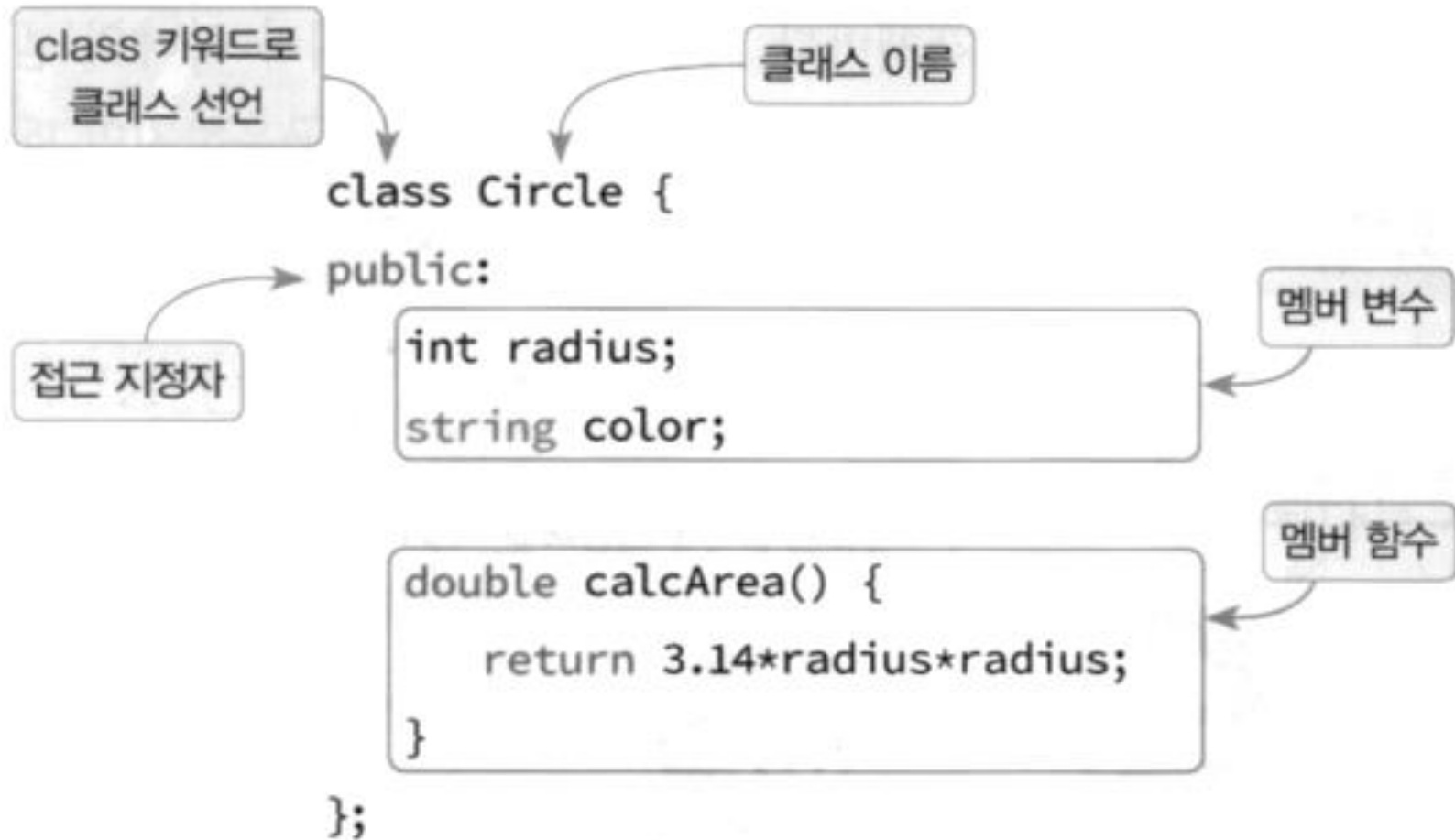
```
반환형 멤버함수2();
```

멤버 함수 선언부

```
};
```

객체

■ 클래스 정의하기



객체

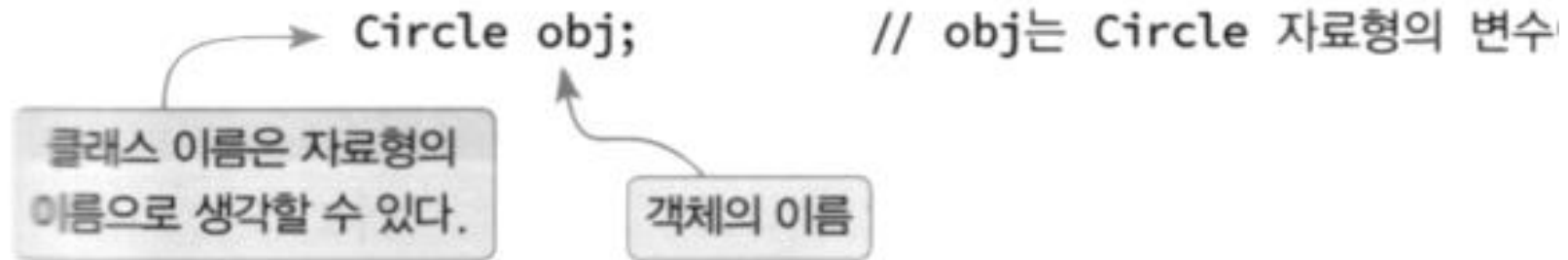
■ 접근 지정자

- `private`
 - 클래스 안에서만 접근(사용)할 수 있음
- `protected`
 - 클래스 안과 상속된 클래스에서 접근 가능
- `public`
 - 어디서나 접근이 가능

객체

■ 객체 생성

- 클래스 타입으로 변수를 선언하면 해당 객체(인스턴스)가 생성됨



객체

■ 객체 멤버 접근

- . 연산자로 접근

`obj.radius = 3;`

// obj의 멤버 변수인 radius에 3을 저장

obj 객체의

radius 멤버 변수에 접근

`obj.calcArea();`

// obj의 멤버 함수인 calcArea()를 호출

obj 객체의

멤버 함수 calcArea() 호출

객체

■ chapter04/ex01_class.cpp] 객체의 사용

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

class Circle {
public:
    int radius;    // 반지름
    string color;  // 색상

    double calcArea() {
        return 3.14 * radius * radius;
    }
};
```

객체

■ chapter04/ex01_class.cpp] 객체의 사용

```
int main(int argc, char const *argv[])
{
    Circle obj; // 객체 생성
    obj.radius = 100;
    obj.color = "blue";

    cout << "원의 면적 " << obj.calcArea() << endl;
    return 0;
}
```

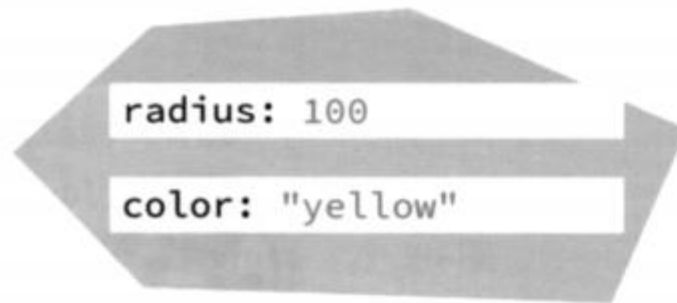
원의 면적 31400

객체

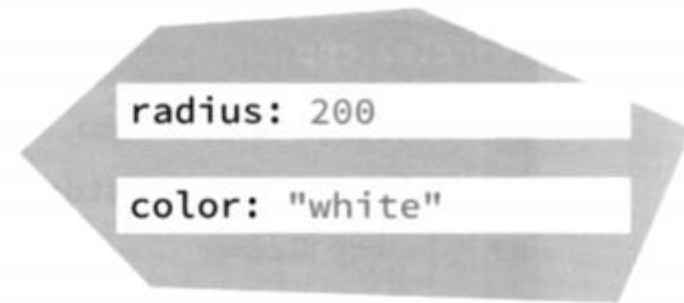
■ chapter04/ex02_class.cpp] 여러 개의 객체 생성

```
int main(int argc, char const *argv[]) {  
    Circle pizza1, pizza2; // 객체 생성  
  
    pizza1.radius = 100;  
    pizza1.color = "yellow";  
    cout << "피자의 면적 " << pizza1.calcArea() << endl;  
  
    pizza1.radius = 200;  
    pizza1.color = "white";  
    cout << "피자의 면적 " << pizza2.calcArea() << endl;  
  
    return 0;  
}
```

피자의 면적 31400
피자의 면적 12.56



pizza 1



pizza 2

객체

■ chapter04/ex03_car.cpp] Car 클래스

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

class Car {
public:
    int speed;    // 속도
    int gear;     // 기어
    string color; // 색상

    void speedUp() {
        speed += 10;
    }

    void speedDown() {
        speed -= 10;
    }
};
```

객체

■ chapter04/ex03_car.cpp] Car 클래스

```
int main(int argc, char const *argv[])
{
    Car myCar;

    myCar.speed = 100;
    myCar.gear = 3;
    myCar.color = "red";

    myCar.speedUp();
    myCar.speedDown();

    return 0;
}
```

객체

■ chapter04/ex04_overload.cpp] 멤버 함수 오버로드

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

class PrintData {
public:
    void print(int i) { cout << i << endl;}
    void print(double f) { cout << f << endl;}
    void print(string s = "No Data!") { cout << s << endl;}
};
```

객체

■ chapter04/ex04_overload.cpp] 멤버 함수 오버로드

```
int main(int argc, char const *argv[]) {  
    PrintData prn;  
  
    prn.print(1);  
    prn.print(3.14);  
    prn.print("C++ is cool.");  
    prn.print();  
  
    return 0;  
}
```

1
3.14
C++ is cool.
No Data!

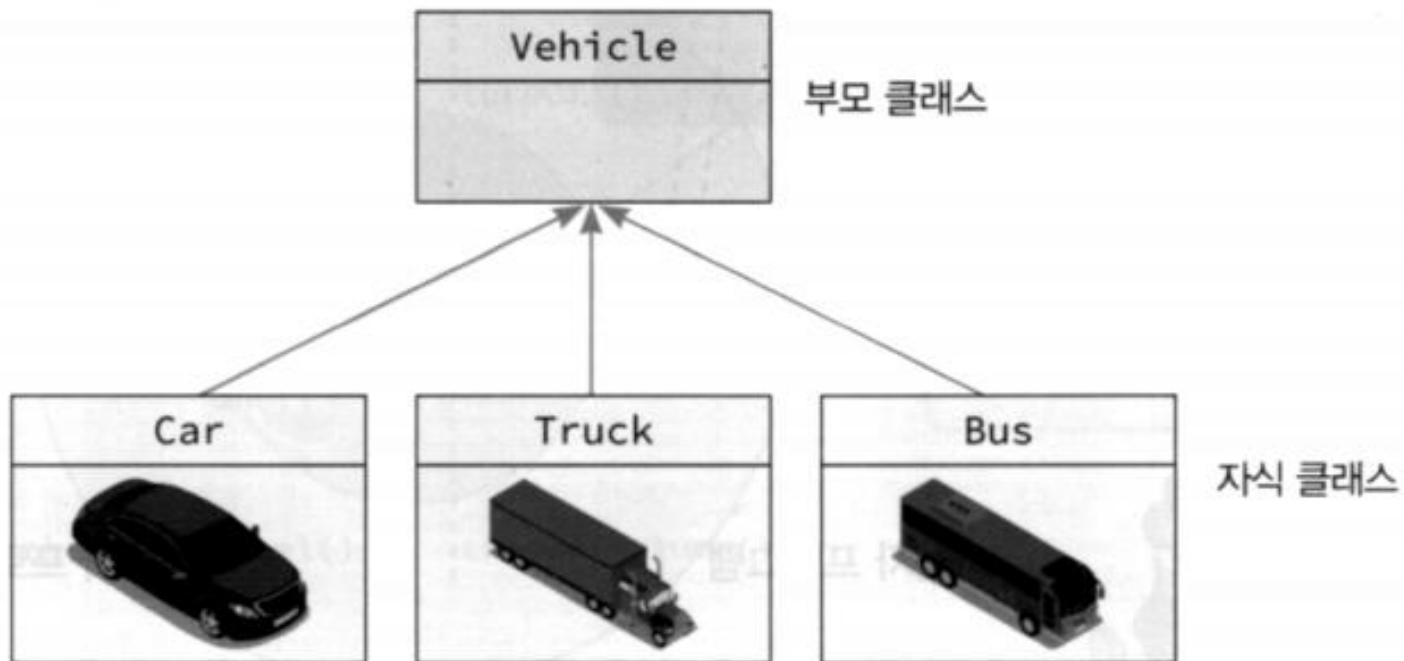
■ 객체 지향의 특징

- 캡슐화
- 정보 은닉
- 상속과 다형성

객체

■ 상속

```
class 자식 클래스명: public 부모 클래스명 {  
    // 자식 클래스 멤버 정의  
}
```



객체

■ chapter04/ex05_inherit.cpp] 상속

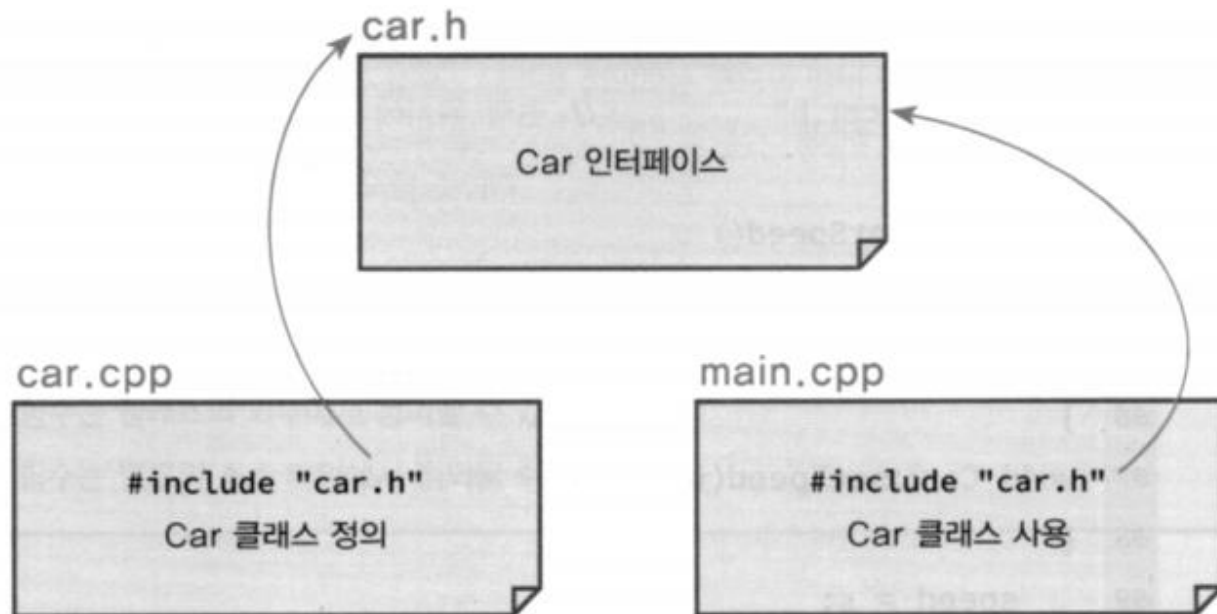
```
class Shape {
protected:
    int x, y;
public:
    void draw() {}
    void move() {}
};

class Rectangle: public Shape {
protected:
    int width;
    int height;
public:
    int calcArea() {
        return width*height;
    }
};
```

객체

■ 클래스 원형과 구현 정의 분리

- 헤더 파일에 클래스 원형
 - 멤버 변수 정의
 - 멤버 함수의 원형 정의
- cpp 파일에 멤버 함수 정의
 - 헤더 파일을 먼저 include
 - 멤버 함수 구현



객체

■ 다중 파일을 위한 프로젝트 관리

- Easy C++ projects 확장 팩 설치



Easy C++ projects

acharluk.easy-cpp-projects

Alejandro Charte Luque

87,021

★★★★☆

저장소

v2.0.0

Create C++ projects easily for GCC, WSL-GCC, Clang and MSVC

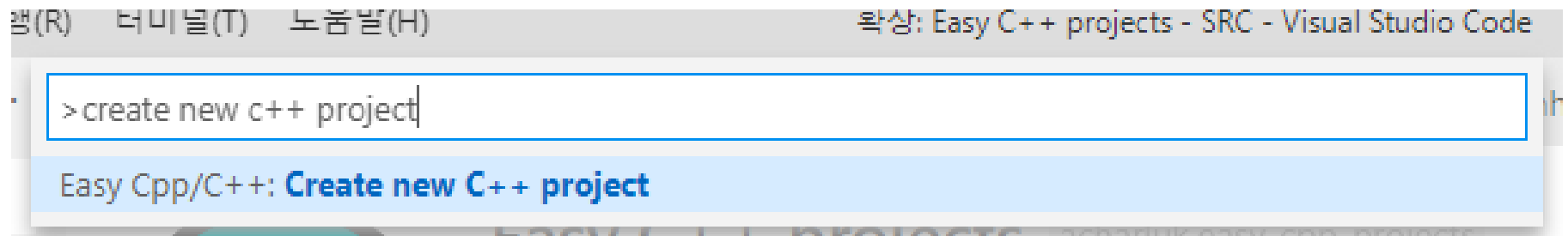
설치

[세부 정보](#) [기능 기여도](#) [변경 로그](#)

객체

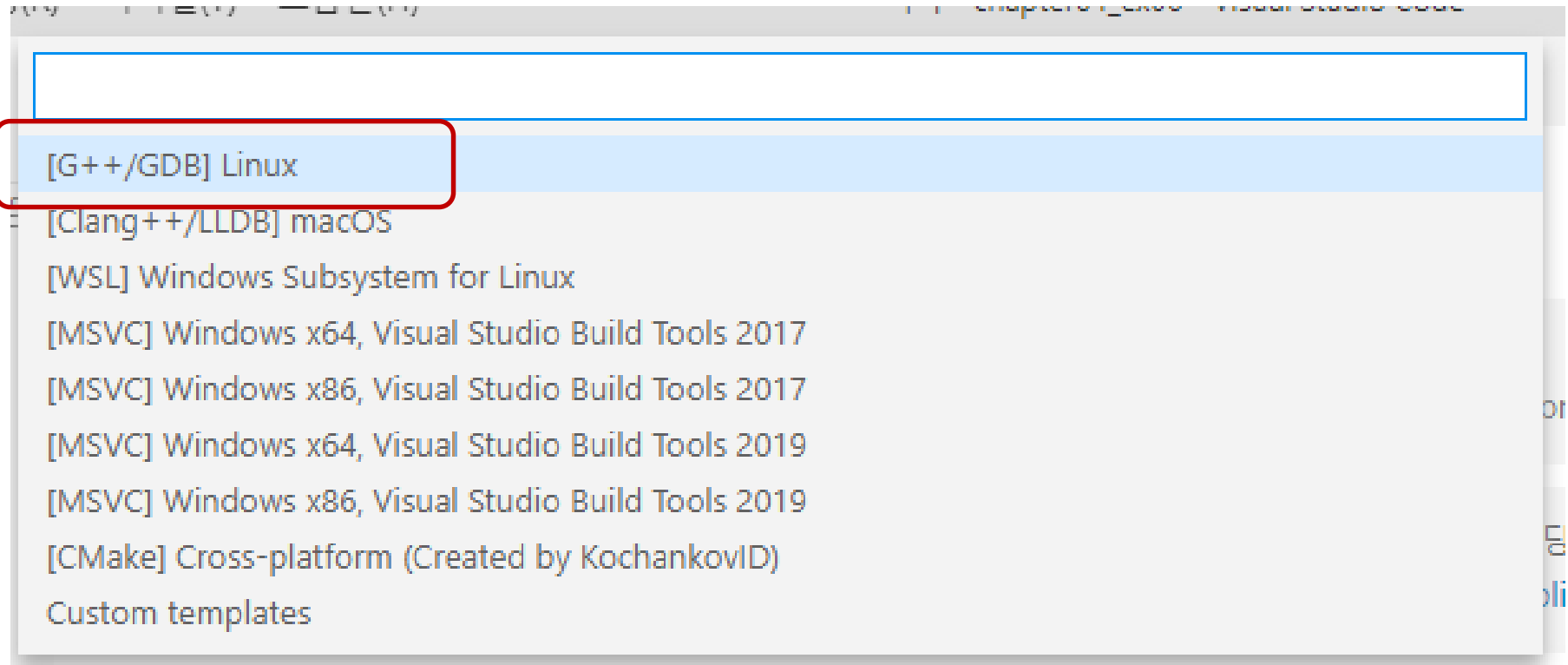
■ 프로젝트 만들기

- 디렉토리 새로 오픈
 - chapter04_ex06 디렉토리
- F1> create new c++ project



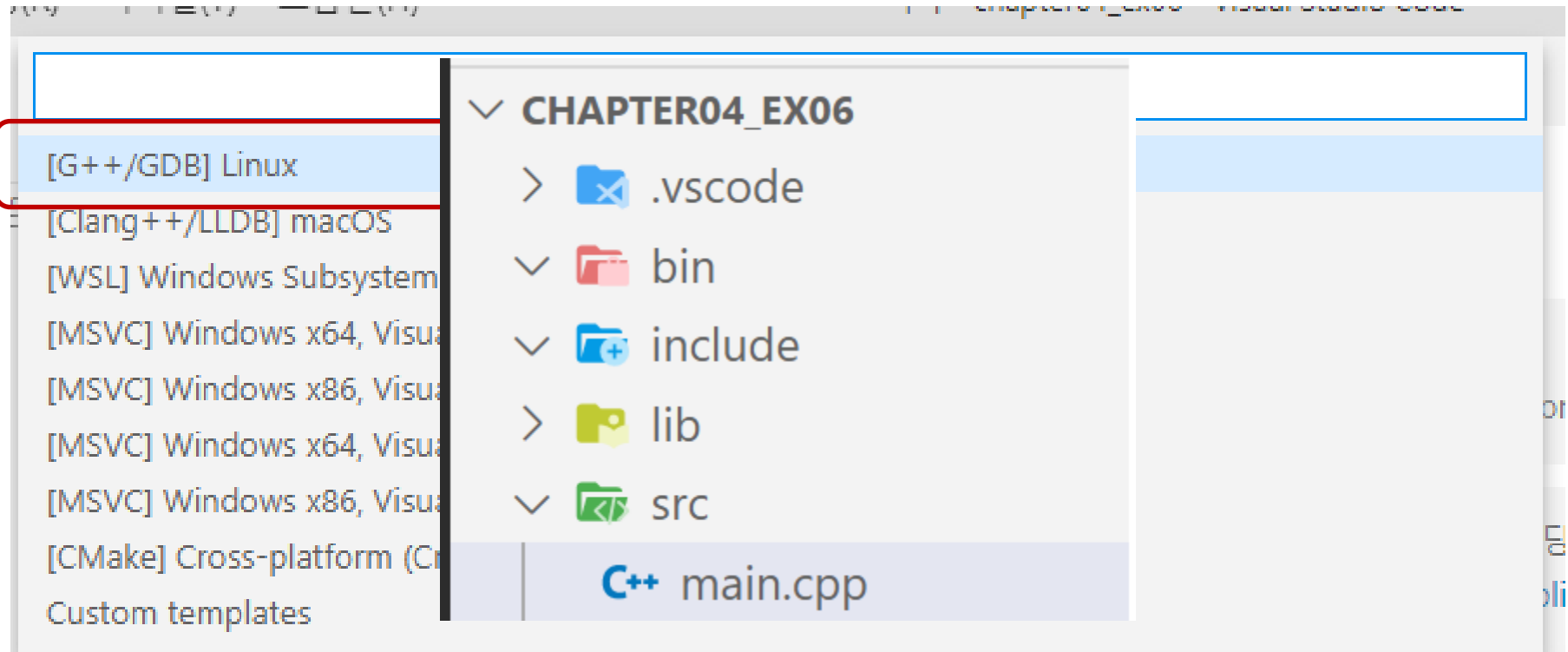
객체

■ 프로젝트 만들기



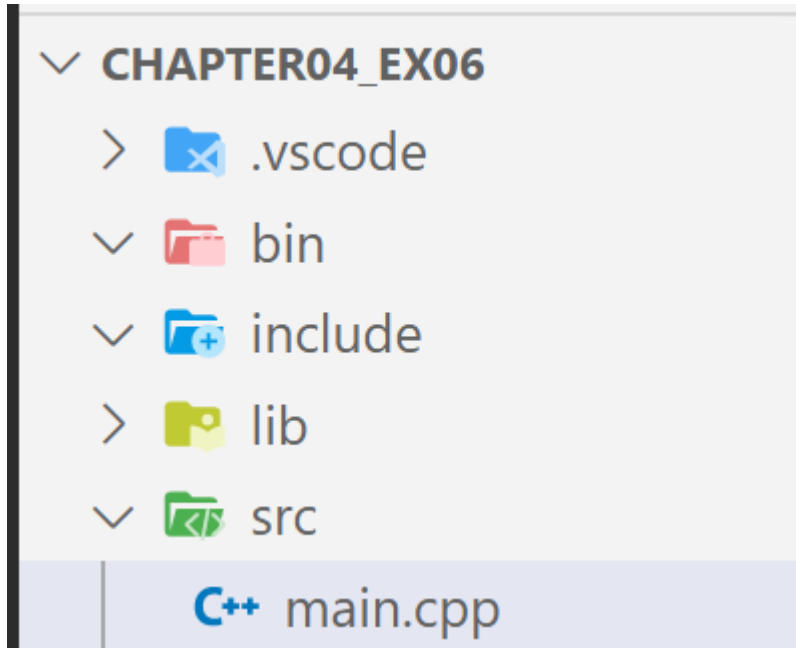
객체

■ 프로젝트 만들기



객체

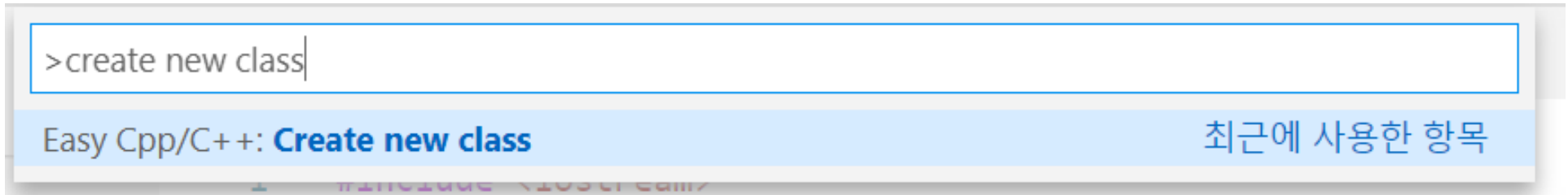
■ 프로젝트 만들기



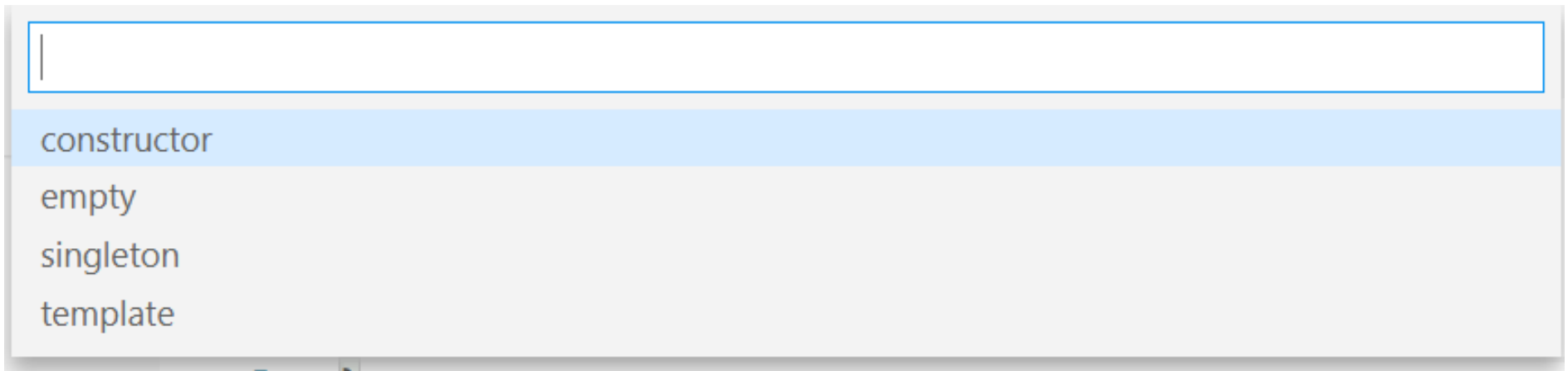
객체

■ 클래스 만들기

- F1> create new class



- 클래스 유형 선택 : constructor



객체

■ 클래스 만들기

- 클래스명 입력

Car|

Enter class name(확인하려면 'Enter' 키를 누르고, 취소하려면 'Escape' 키를 누름)

객체

■ include/Car.hpp] Car 클래스 헤더 파일

```
#pragma once

class Car {

public:
    Car();
    ~Car();
};
```

■ src/Car.cpp] Car 클래스 헤더 파일

```
#include "Car.hpp"

Car::Car() {

}

Car::~~Car() {

}
```

객체

■ include/Car.hpp] Car 클래스 헤더 파일

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

class Car {
    int speed;    // 속도
    int gear;     // 기어
    string color; // 색상

public:
    int getSpeed();
    void setSpeed(int s);
};
```

객체

■ src/Car.cpp] Car 클래스 정의 파일

```
#include<iostream>
#include "Car.h"

void Car::setSpeed(int s) {
    speed = s;
}

int Car::getSpeed() {
    return speed;
}
```

객체

■ src/main.cpp]

```
#include<iostream>
#include<string>
#include "Car.hpp"
using namespace std;

int main(int argc, char const *argv[])
{
    Car myCar;

    myCar.setSpeed(100);

    cout << "속도 : " << myCar.getSpeed() << endl;

    return 0;
}
```

속도 : 100
