

# C++\_클래스와 객체

*class, object*

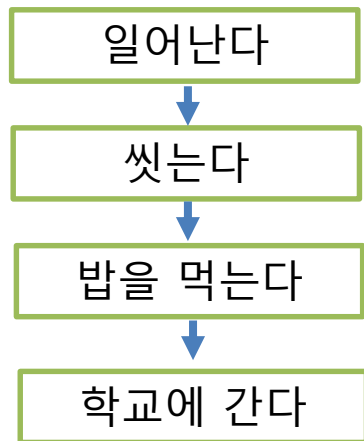
# 객체 지향 프로그래밍

## ■ 객체(Object)란?

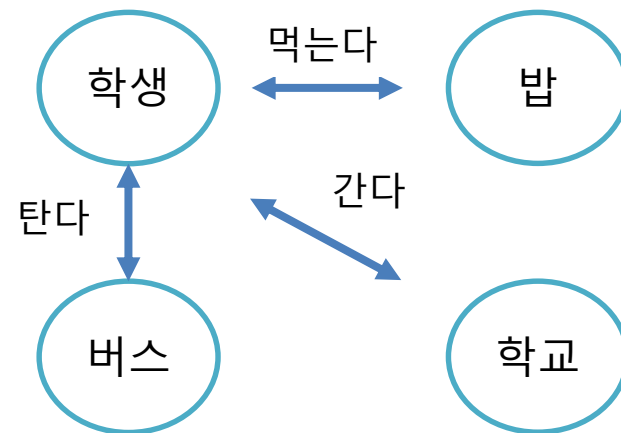
- 의사나 행위가 미치는 대상 -> 사전적 의미
- 구체적, 추상적 데이터 단위 (구체적-책상, 추상적-회사)

## ■ 객체지향 프로그래밍(Object Oriented Programming, OOP)

- 객체를 기반으로 하는 프로그래밍
- 먼저 객체를 만들고, 객체 사이에 일어나는 일을 구현함.



<절차지향 -C언어>



<객체지향 -C++,Java>

# 객체지향 프로그래밍이란?

## 절차지향 프로그래밍

```
작업 1();  
작업 2();  
작업 3();  
.  
.  
작업 99();  
작업 100();
```

작업(함수) 100개가 동  
등한 위치에서 나열되  
어 있다.

## 객체지향 프로그래밍

```
작업 1();  
작업 2();  
작업 3();  
.  
.  
작업 10();
```

```
작업 1();  
작업 2();  
작업 3();  
.  
.  
작업 10();
```

```
작업 1();  
작업 2();  
작업 3();  
.  
.  
작업 10();
```

연관있는 작업을 객체  
로 묶어서 처리하기때  
문에 보다 효율적으로  
관리할 수 있다.

# C++의 객체 지향 특성

- 객체와 캡슐화(Encapsulation)

캡슐화는 데이터를 캡슐로 싸서 외부의 접근으로부터 데이터를 보호하는 객체 지향 특성이다.

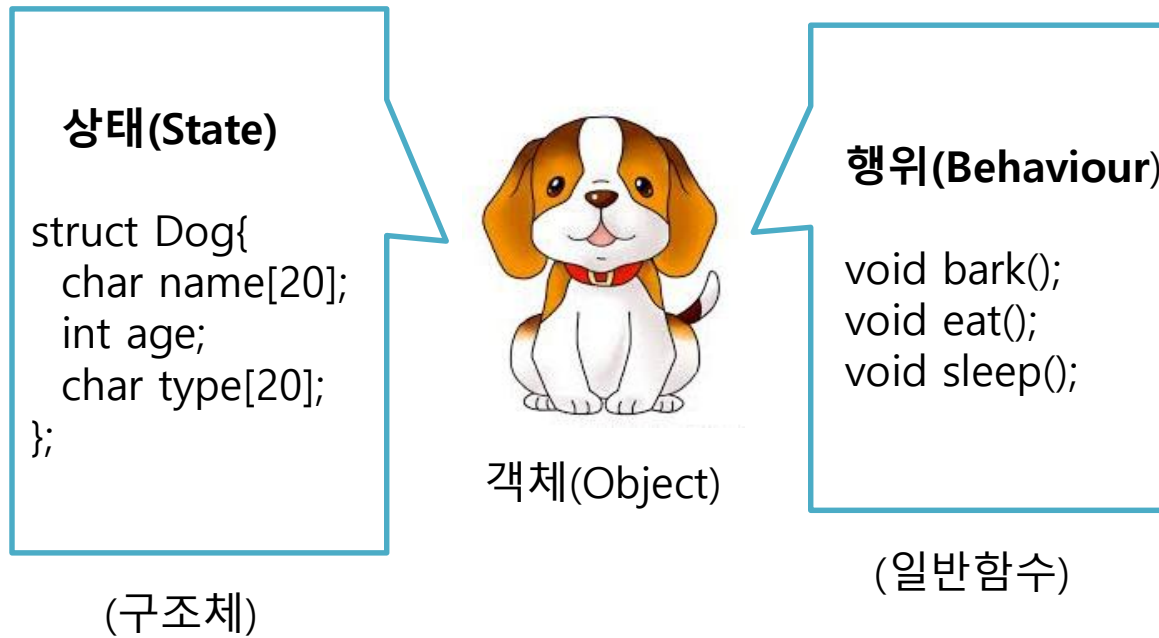
C++에서는 캡슐의 역할을 하는 것이 클래스이며 **class 키워드**를 이용하여 작성한다. 객체는 클래스라는 틀에서 생겨난 실체(instance)-인스턴스-이다.

C++ 클래스는 멤버변수들과 멤버 함수들로 이루어지며, 멤버들은 캡슐 외부에 공개하거나(public), 보이지 않게(private) 선언할 수 있다.

- 상속성

자식 클래스의 객체가 생성될때 부모 클래스의 멤버나 함수를 사용할수 있다. 그래서 클래스를 재사용 할 수 있고, 유지 보수하기에 좋다.

# 구조체의 진화 -> 클래스



★ 절차지향적 접근 방법

# 구조체의 진화 -> 클래스



객체(Object)

```
class Dog{
```

```
    string name;  
    int age;  
    string type;
```

```
    void bark();  
    void eat();  
    void sleep();
```

```
};
```

클래스

★ 클래스는 구조체에 함수를 포함시킨 틀이다.

# 클래스 정의 및 사용

## ● 클래스(class) 정의

- 클래스란 객체(사물)를 추상화한 자료형이다.
- 클래스란 객체를 정의하는 틀 혹은 설계도이다.
- 클래스에 멤버 변수와 멤버 함수를 선언한다.
- 클래스 이름은 대문자로 시작한다.
- 접근 제어자 private – 접근 불허, public – 접근 허용

```
class 클래스 이름{  
private:  
    멤버 변수;  
public:  
    멤버 함수;  
}
```

## ● 인스턴스(instance)

- 클래스를 사용하기 위해 생성된 객체를 인스턴스(instance)라 한다.
- 인스턴스로 클래스의 멤버 변수에 점(.) 연산자로 접근하여 값을 지정한다.

클래스 이름 인스턴스(객체)  
인스턴스.멤버변수

# Dog 클래스 만들기

- 클래스의 구성 요소(속성, 기능)

```
//Dog 클래스 정의
class Dog {
public: //접근 제어자 - 멤버변수 및 함수에 접근 허용
    string type; //종류
    int age;      //나이 ← 멤버 변수

    void dogInfo()
    {
        cout << "강아지 종류 : " << type << endl;
        cout << "강아지 나이 : " << age << "세" << endl;
    }

    void bark()
    {
        cout << "멍~ 멍~\n";
    }
};
```

← 멤버 함수



# Dog 클래스 만들기

## ■ 클래스의 인스턴스 생성

```
int main()
{
    Dog dog1; //객체(인스턴스) 생성

    dog1.type = "푸들"; //멤버 변수 초기화
    dog1.age = 2;
    dog1.dogInfo();
    dog1.bark();

    Dog dog2; //객체(인스턴스) 생성

    dog2.type = "진돗개";
    dog2.age = 3;
    dog2.dogInfo();
    dog2.bark();

    return 0;
}
```

```
강아지 종류 : 푸들
강아지 나이 : 2세
멍~ 멍~
강아지 종류 : 진돗개
강아지 나이 : 3세
멍~ 멍~
```

# 생성자(Constructor)

## ■ 클래스의 구성 요소(생성자)

- ✓ 생성자는 객체가 만들어질때 자동으로 호출된다.
- ✓ 생성자 이름은 클래스와 동일하다.
- ✓ 생성자가 정의되어 있지 않으면 컴파일러가 자동으로 기본생성자(default constructor)를 제공한다.

```
class Dog {  
public: //접근 제어(공개)  
    string type; //종류  
    int age;      //나이  
  
    Dog() {} //기본 생성자(생략 가능)  
  
    void dogInfo() {  
        cout << "강아지 종류: " << type << endl;  
        cout << "강아지 나이: " << age << endl;  
    }  
  
    void bark() {cout << "왈~ 왈~\n";}   
};
```

# 생성자(Constructor)

- 클래스의 구성 요소(생성자) – 인자가 있는 생성자

```
class Dog {  
public: //접근 제어(공개)  
    string type; //종류  
    int age;      //나이  
  
    Dog() {} //기본 생성자(생략 가능)  
  
    Dog(string t, int a) { //인자를 가진 생성자  
        type = t;  
        age = a;  
    }  
  
    void dogInfo() {  
        cout << "강아지 종류: " << type << endl;  
        cout << "강아지 나이: " << age << endl;  
    }  
  
    void bark() { cout << "왈~ 왈~\n"; }  
};
```

\* 생성자 오버로딩(중복)

이름이 같고 매개변수가 다를 것을 뜻함

# 생성자(Constructor)

- 클래스의 구성 요소(생성자) – 인자가 있는 생성자

```
int main()
{
    //기본 생성자로 인스턴스 dog1 생성
    Dog dog1;
    dog1.type = "푸들";
    dog1.age = 2;

    dog1.dogInfo();
    dog1.bark();

    //인자가 있는 생성자로 인스턴스 dog2 생성
    Dog dog2("진돗개", 3);
    dog2.bark();
    dog2.bark();

    return 0;
}
```

```
강아지 종류 : 푸들
강아지 나이 : 2
왈~ 왈~
강아지 종류 : 진돗개
강아지 나이 : 3
왈~ 왈~
```

# 소멸자(Destructor)

## ▪ 클래스의 구성 요소(소멸자)

- ✓ 소멸자는 객체가 생성된 후 자동으로 호출된다.
- ✓ 소멸자 이름은 클래스와 동일하고, 이름 앞에 '~'을 붙인다.
- ✓ 소멸자가 정의되어 있지 않으면 컴파일러가 자동으로 기본 소멸자(default constructor)를 제공한다.

```
class Dog {  
public: //접근 제어(공개)  
    string type; //종류  
    int age;      //나이  
  
    Dog() {} //기본 생성자(생략 가능)  
  
    Dog(string t, int a) { //인자를 가진 생성자  
        type = t;  
        age = a;  
    }  
  
    ~Dog() { cout << "객체가 소멸합니다.\n"; } //소멸자
```

```
강아지 종류 : 푸들  
강아지 나이 : 2  
왈~왈~  
강아지 종류 : 진돗개  
강아지 나이 : 3  
왈~왈~  
객체가 소멸합니다.  
객체가 소멸합니다.
```

# 클래스 선언부와 구현부 분리

- 클래스의 선언부와 구현부 분리 이유
  - 클래스를 사용하는 다른 C++ 파일에서는 컴파일 시 클래스 선언부(헤더 파일)만 필요하기 때문임 – 재사용성 향상

```
class Dog {  
public:  
    string type;  
    int age;  
  
    Dog(); //기본 생성자  
    Dog(string t, int a); //매개변수가 있는 생성자  
    ~Dog(); //소멸자  
  
    void dogInfo();  
    void bark();  
};
```

# 클래스 선언부와 구현부 분리

- 클래스의 선언부와 구현부 분리 이유

```
Dog::Dog() {  
    type = "강아지";  
    age = 1;  
}  
  
Dog::Dog(string t, int a) {  
    type = t;  
    age = a;  
}  
  
Dog::~~Dog() {  
    cout << "소멸자 입니다.\n";  
}  
  
void Dog::dogInfo() {  
    cout << "강아지 종류: " << type << endl;  
    cout << "강아지 나이: " << age << endl;  
}
```

# 클래스 선언부와 구현부 분리

- 클래스의 선언부와 구현부 분리

```
void Dog::bark() {  
    cout << "왈~ 왈~\n";  
}  
  
int main()  
{  
    Dog dog1;  
    dog1.dogInfo();  
    dog1.bark();  
  
    Dog dog2("진돗개", 3);  
    dog2.dogInfo();  
    dog2.bark();  
  
    return 0;  
}
```



# 캡슐화 - 정보은닉(Information Hiding)

- 정보 은닉

- 접근 제어자 : 접근 권한 지정

| 접근 지정자  | 설 명                |
|---------|--------------------|
| public  | 외부 클래스 어디에서나 접근 가능 |
| private | 외부 클래스에서 접근 불가     |

- 클래스 멤버에 접근 방법 - **get(), set()** 함수를 만들어 사용
    - 설정자(setter) : set + 멤버변수이름(), 예) setType()
    - 접근자(getter) : get + 멤버변수이름(), 예) getType()

# 정보은닉(Information Hiding)

- 접근 제어자 사용

```
class Dog {  
private: //클래스 외부에서 접근 불가  
    string type;  
    int age;  
  
public: //클래스 외부에서 접근 가능  
    Dog() {} //기본 생성자  
  
    //setter(설정자)  
    void setType(string _type) {type = _type;}  
    void setAge(int _age) {age = _age;}  
  
    //getter(접근자)  
    string getType() {return type;}  
    int getAge() {return age;}  
};
```

# 정보은닉(Information Hiding)

- 접근 제어자 사용

```
int main()
{
    Dog dog1;
    //dog1.type = ""; //접근 불가

    dog1.setType("말티즈");
    dog1.setAge(2);

    cout << "강아지 종류: " << dog1.getType() << endl;
    cout << "강아지 나이: " << dog1.getAge() << endl;

    return 0;
}
```

# 헤더 파일(.h), cpp 파일로 분할하기

## ❖ 분할 컴파일

- Dog.h – 헤더 파일(클래스 포함)
- Dog.cpp – 함수 포함(생성자 및 멤버 함수 구현)
- Main.cpp – 실행 파일(객체 생성)

# 헤더 파일(.h), cpp 파일로 분할하기

## ▪ Dog.h

```
#ifndef DOG_H //조건부 컴파일 블록 시작
#define DOG_H //헤더파일 중복되지 않도록 매크로 이름 정의
#include <string>
using namespace std;

class Dog {
private:
    string type;
    int age;

public:
    Dog(string _type, int _age);
    ~Dog(){}

    string getType();
    int getAge();
    void bark();
};
#endif //조건부 컴파일 블록 종료
```

# 헤더 파일(.h), cpp 파일로 분할하기

## ▪ Dog.cpp

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

//생성자 초기화
Dog::Dog(string _type, int _age) {
    type = _type;
    age = _age;
}

string Dog::getType() {
    return type;
}

int Dog::getAge() {
    return age;
}

void Dog::bark() {
    cout << "왈~ 왈~\n";
}
```

# 헤더 파일(.h), cpp 파일로 분할하기

## ▪ DogMain.cpp

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

int main()
{
    Dog dog1("진돗개", 5);

    cout << "***** 강아지 정보 *****\n";
    cout << "강아지 종류: " << dog1.getType() << endl;
    cout << "강아지 나이: " << dog1.getAge() << endl;

    dog1.bark();

    return 0;
}
```

# 헤더 파일(.h), cpp 파일로 분할하기

## ▪ DogMain.cpp

```
const int SIZE = 3;

// Dog 객체 배열 생성
Dog dogs[SIZE] = {
    Dog("진돗개", 3),
    Dog("말티즈", 2),
    Dog("시베리안 허스키", 4)
};

cout << "***** 강아지 목록 *****\n";

for (int i = 0; i < SIZE; i++) {
    cout << i + 1 << "번 강아지 정보\n";
    cout << "종류: " << dogs[i].getType() << endl;
    cout << "나이: " << dogs[i].getAge() << "살\n";
    dogs[i].bark();
    cout << endl;
}
```

```
***** 강아지 목록 *****
1번 강아지 정보
종류: 진돗개
나이: 3살
왈~ 왈~

2번 강아지 정보
종류: 말티즈
나이: 2살
왈~ 왈~

3번 강아지 정보
종류: 시베리안 허스키
나이: 4살
왈~ 왈~
```



# 실습 예제

- Book 클래스

| Book   |       |
|--|-------|
| number   | //책번호 |
| title  | //책제목 |
| author   | //저자  |
| setNumber()<br>getNumber()<br>setTitle()<br>getTitle()<br>setAuthor()<br>getAuthor() |       |

# 실습 예제

- Book 클래스

```
//Book 클래스 정의 - 정보 은닉
class Book {
private:
    int number;    //책 번호
    string title;  //책 제목
    string author; //저자

public:
    //Book();    //기본 생성자(생략)

    //get(), set() 함수로 private 멤버에 접근
    void setNumber(int n);
    int getNumber();
    void setTitle(string t);
    string getTitle();
    void setAuthor(string a);
    string getAuthor();
};
```

# 실습 예제

- Book 클래스

```
void Book::setNumber(int n){
    number = n;
}

int Book::getNumber(){
    return number;
}

void Book::setTitle(string t) {
    title = t;
}

string Book::getTitle() {
    return title;
}

void Book::setAuthor(string a) {
    author = a;
}

string Book::getAuthor() {
    return author;
}
```

# 실습 예제

- Book 클래스

```
int main()
{
    Book book1;

    book1.setNumber(100);
    book1.setTitle("채식주의자");
    book1.setAuthor("한강");

    cout << "***** 책의 정보 *****" << endl;
    cout << "책 번호 : " << book1.getNumber() << endl;
    cout << "책 제목 : " << book1.getTitle() << endl;
    cout << "책 저자 : " << book1.getAuthor() << endl;

    return 0;
}
```

# 객체 배열

## ▪ Book 클래스 – 객체 배열

```
class Book {  
private:  
    int number;    //책 번호  
    string title;  //책 제목  
    string author; //저자  
public:  
    Book(int n, string t, string a);  
  
    //get() 함수만 사용  
    int getNumber();  
    string getTitle();  
    string getAuthor();  
};
```

```
// 생성자 초기화  
Book::Book(int n, string t, string a) {  
    number = n;  
    title = t;  
    author = a;  
}  
  
int Book::getNumber() {  
    return number;  
}  
  
string Book::getTitle() {  
    return title;  
}  
  
string Book::getAuthor() {  
    return author;  
}
```

# 객체 배열

- Book 클래스

```
int main()
{
    //객체 배열
    Book book[3] = {
        Book(100, "채식주의자", "한강"),
        Book(101, "C++ 완전정복", "조규남"),
        Book(102, "모두의 C언어", "이형우"),
    };

    cout << "***** 책의 정보 *****" << endl;
    for (int i = 0; i < 3; i++)
    {
        cout << "책 번호 : " << book[i].getNumber() << endl;
        cout << "책 제목 : " << book[i].getTitle() << endl;
        cout << "책 저자 : " << book[i].getAuthor() << endl;
    }

    return 0;
}
```

# this 예약어 사용

## ▪ 자신의 메모리를 가리키는 this

- 생성된 인스턴스 스스로를 가리키는 예약어
- 객체 자신의 메모리상의 주소를 나타내는 포인터이다.

```
class Birthday {  
private:  
    int day;  
    int month;  
    int year;  
  
public:  
    void setYear(int year) {  
        this->year = year;  
    }  
  
    void printThis() {  
        cout << this << endl;  
    }  
};
```

# this 예약어 사용

## ■ 자신의 메모리를 가리키는 this

- 생성된 인스턴스 스스로를 가리키는 예약어
- 객체 자신의 메모리상의 주소를 나타내는 포인터이다.

```
int main()
{
    Birthday bDay;
    bDay.setYear(2025);

    cout << &bDay << endl; //객체의 주소

    bDay.printThis(); //this의 주소

    return 0;
}
```

```
00000051901FFD38
00000051901FFD38
```



# this 예약어

## ▪ Car 클래스

| Car                     |
|-------------------------|
| model //모델<br>year //연식 |
| carInfo()<br>drive()    |

# this 예약어

## ▪ Car.h

```
#ifndef CAR_H
#define CAR_H
#include <string>
using namespace std;

class Car {
private:
    string model; //모델명
    int year;     //연식

public:
    Car(string model, int year);

    void carInfo();

    void drive();
};
#endif // !CAR_H
```

# this 예약어

## ▪ Car.cpp

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

Car::Car(string model, int year) { //생성자
    this->model = model;
    this->year = year;
}

void Car::carInfo() { //정보 출력 함수
    cout << "모델명: " << this->model <<
        ", 연식: " << this->year << endl;
}

void Car::drive() {
    cout << this->model << "가 달립니다.\n";
}
```

# this 예약어 사용

## ■ CarMain.cpp

```
Car car1("Ionic6", 2024);
car1.carInfo();
cout << "=====\\n";

//객체 배열
Car carList[3] = {
    Car("소나타", 2020),
    Car("스포티지", 2022),
    Car("EV6", 2025)
};
//cout << size(carList) << endl;  //3

for (int i = 0; i < size(carList); i++) {
    carList[i].carInfo();
    carList[i].drive();
}
```

```
모델명 : Ionic6, 연식 : 2024
=====
모델명 : 소나타, 연식 : 2020
소나타가 달립니다.
모델명 : 스포티지, 연식 : 2022
스포티지가 달립니다.
모델명 : EV6, 연식 : 2025
EV6가 달립니다.
```

# 실습 문제 1 - 클래스

회원(Member) 클래스를 정의하고 배열을 사용하여 객체를 생성하세요.

[파일이름: MemberTest.cpp]

| 데이터 이름 | 필드 이름    | 타입  | 접근 제어   |
|--------|----------|-----|---------|
| 아이디    | id       | 문자열 | private |
| 패스워드   | password | 문자열 | private |

👉 실행 결과

```
***** 회원 현황 *****  
아이디 : flower, 패스워드 : f1234  
아이디 : tree, 패스워드 : t1234  
아이디 : bird, 패스워드 : b1234
```

# 회원 로그인 서비스

- 회원 로그인 서비스

```
//회원 서비스 클래스 정의
class MemberService {
public:
    bool login(string id, string pwd); //로그인
    void logout(); //로그아웃
};

bool MemberService::login(string id, string pwd) {
    if (id.compare("hangang") == 0 && pwd.compare("k1234") == 0) {
        return true;
    }
    return false;
}

void MemberService::logout() {
    cout << "로그아웃 되었습니다.\n";
}
```

# 회원 로그인 서비스

- 회원 로그인 서비스

```
MemberService service;

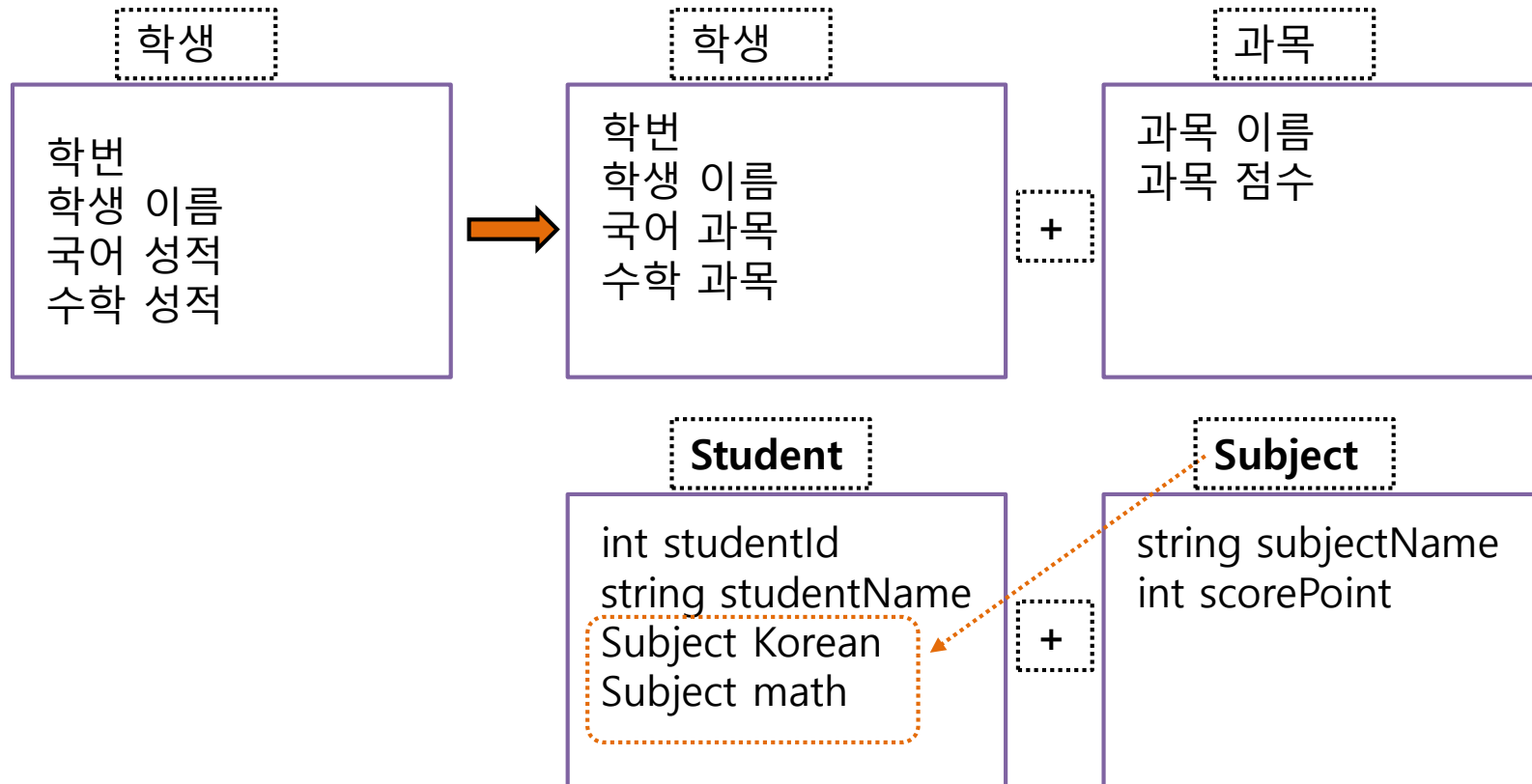
string userId = "hangang";
string password = "k1234";

//로그인
bool result = service.login(userId, password);
if (result) {
    cout << "로그인 되었습니다.\n";
    cout << userId << "님 환영합니다.!\n";
}
else {
    cout << "아이디나 비밀번호가 일치하지 않습니다.\n";
}
cout << "-----\n";

//로그 아웃
service.logout();
```

# 클래스(객체) 참조

## ■ 클래스 간 참조



문제점 : 이 클래스는 학생에 대한 클래스인데 과목 변수가 계속 늘어남

해결책 : 과목이름과 성적을 과목(Subject) 클래스로 분리함.



# 클래스(객체) 참조

## ▪ Subject.h

```
#ifndef SUBJECT_H
#define SUBJECT_H

#include <iostream>
using namespace std;

class Subject {
private:
    string subjectName; //과목명
    int scorePoint;      //점수

public:
    void setSubjectName(string subjectName);
    string getSubjectName();

    void setScorePoint(int scorePoint);
    int getScorePoint();
};
#endif
```

# 클래스(객체) 참조

## ▪ Subject.cpp

```
#include "Subject.h"

void Subject::setSubjectName(string subjectName) {
    this->subjectName = subjectName;
}

string Subject::getSubjectName() {
    return subjectName;
}

void Subject::setScorePoint(int scorePoint) {
    this->scorePoint = scorePoint;
}

int Subject::getScorePoint() {
    return scorePoint;
}
```

# 클래스(객체) 참조

## ▪ Student.h

```
#include "Subject.h"
using namespace std;

class Student {
private:
    int studentId;        //학생 아이디
    string studentName;   //학생 이름
    Subject korean;       //국어
    Subject math;         //수학

public:
    Student(int studentId, string studentName);

    void setKoreanSubject(string name, int score);
    void setMathSubject(string name, int score);

    void showInfo();
};
#endif
```

# 클래스(객체) 참조

## ▪ Student.cpp

```
#include "Student.h"

Student::Student(int studentId, string studentName) {
    this->studentId = studentId;
    this->studentName = studentName;
}

void Student::setKoreanSubject(string name, int score) {
    korean.setSubjectName(name);
    korean.setScorePoint(score);
}
```

# 클래스(객체) 참조

## ▪ Student.cpp

```
void Student::setMathSubject(string name, int score) {  
    math.setSubjectName(name);  
    math.setScorePoint(score);  
}  
  
void Student::showInfo() {  
    cout << "학번: " << studentId  
        << "\n이름: " << studentName  
        << "\n국어점수: " << korean.getScorePoint()  
        << "\n수학점수: " << math.getScorePoint()  
        << "\n-----\n";  
}
```

# 클래스(객체) 참조

## ▪ Main

```
#include "Student.h"

int main()
{
    Student lee(1001, "이정우");
    lee.setKoreanSubject("국어", 85);
    lee.setMathSubject("수학", 80);
    lee.showInfo();

    Student shin(1002, "신유진");
    shin.setKoreanSubject("국어", 90);
    shin.setMathSubject("수학", 85);
    shin.showInfo();

    return 0;
}
```

```
학 번 : 1001
이름 : 이정우
국어 점수 : 85
수학 점수 : 80
```

```
-----
학 번 : 1002
이름 : 신유진
국어 점수 : 90
수학 점수 : 85
-----
```

# 학생의 성적 관리 – 배열로 구현

## ▪ Students.h

```
#ifndef STUDENTS_H
#define STUDENTS_H

#include "Subject.h"

class Students {
private:
    int studentId;           //학생 아이디
    string studentName;      //학생 이름
    Subject subjects[10];    //과목(배열)
    int subjectCount = 0;    //현재 과목수

public:
    Students(int studentId, string studentName);

    void addSubject(string name, int score);
    void showInfo();
};
#endif
```

# 학생의 성적 관리 – 배열로 구현

## ▪ Students.cpp

```
#include "Students.h"

Students::Students(int studentId, string studentName) {
    this->studentId = studentId;
    this->studentName = studentName;
}

void Students::addSubject(string name, int score) {
    if (subjectCount < 10) { //과목수 제한
        subjects[subjectCount].setSubjectName(name);
        subjects[subjectCount].setScorePoint(score);
        subjectCount++; //과목수 1 증가
    }
}
```



# 학생의 성적 관리 – 배열로 구현

## ▪ Students.cpp

```
void Students::showInfo() {  
    cout << "학번: " << studentId  
        << "\n이름: " << studentName << endl;  
    for (int i = 0; i < subjectCount; i++) {  
        cout << subjects[i].getSubjectName()  
            << " 점수: " << subjects[i].getScorePoint() << "점" << endl;  
    }  
    cout << "-----\n";  
}
```

# 성적 관리 – 배열로 구현

## ▪ StudentsMain.cpp

```
#include "Students.h"

int main()
{
    //인스턴스 생성
    Students park(1001, "박상희");

    //과목 설정
    park.addSubject("국어", 90);
    park.addSubject("미술", 75);
    park.addSubject("음악", 85);

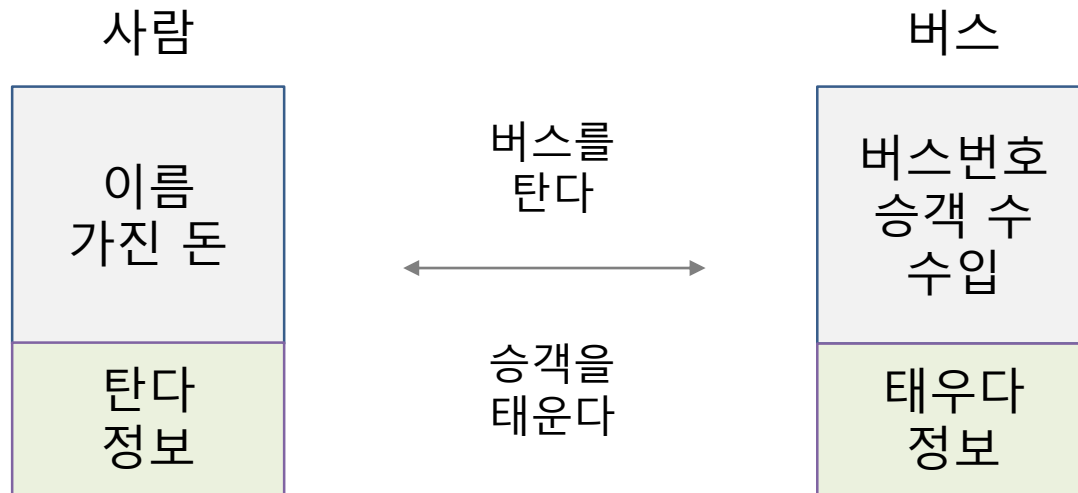
    //정보 출력
    park.showInfo();

    return 0;
}
```

```
학 번 : 1001
이름 : 박 상 희
국 어 점 수 : 90점
미 술 점 수 : 75점
음 악 점 수 : 85점
-----
```

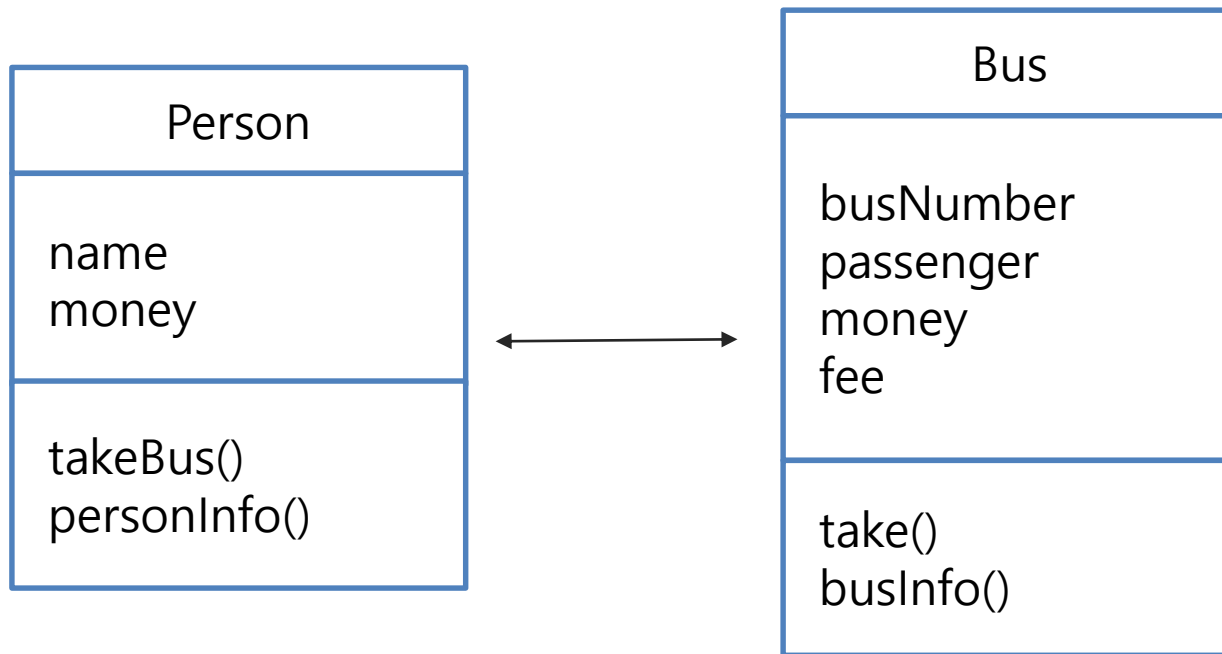
# 클래스(객체)간 협력

## ▪ 사람이 버스를 타는 상황



# 클래스(객체)간 협력

## ▪ 사람, 버스, 택시 클래스 다이어그램



# 클래스(객체)간 협력

## ▪ Bus.h

```
#ifndef BUS_H
#define BUS_H

class Bus {
private:
    int busNumber;    //버스 번호
    int passenger;    //승객수
    int money;        //수입
    const int FEE;    //버스 요금(상수화)
public:
    //기본 매개변수는 선언부에서만 작성함
    Bus(int busNumber, int FEE = 1500);

    void take();      //승객 태우기
    int getFEE();     //요금 가져오기
    void displayInfo(); //버스의 정보 출력
};
#endif
```

# 클래스(객체)간 협력

## ▪ Bus.cpp

```
#include <iostream>
#include "Bus.h"
using namespace std;

//생성자 초기화 목록
Bus::Bus(int busNumber, int FEE) : busNumber(busNumber),
    passenger(0), money(0), FEE(FEE){}

void Bus::take() {
    money += FEE;
    passenger++;
}

int Bus::getFEE() { return FEE; }

void Bus::displayInfo() {
    cout << busNumber << "번 버스: 수입 " << money
        << "원, 승객수: " << passenger << "명\n";
}
```

# 클래스(객체)간 협력

## ▪ Person.h

```
#ifndef PERSON_H
#define PERSON_H

#include <iostream>
#include "Bus.h"

using namespace std;

class Person {
private:
    string name;    //이름
    int money;      //가진 돈
public:
    Person(string name, int money);

    void takeBus(Bus& bus); //매개변수-버스 객체 전달(참조자)
    void displayInfo();

};
#endif
```

# 클래스(객체)간 협력

## ▪ Person.cpp

```
Person::Person(string name, int money) {  
    this->name = name;  
    this->money = money;  
}  
  
void Person::takeBus(Bus& bus) {  
    if (money < bus.getFEE()) {  
        cout << "잔액이 부족합니다.\n";  
    }  
    else {  
        bus.take();  
        money -= bus.getFEE();  
    }  
}  
  
void Person::displayInfo() {  
    cout << name << ": 잔액 " << money << "원\n";  
}
```



# 클래스(객체)간 협력

## ▪ transportMain.cpp

```
//인스턴스 생성
Person woo("우영우", 10000); //사람 인스턴스 생성
Person han("한강", 2000);
Bus bus600(600, 1500); //Bus 인스턴스 생성

//버스 탑승
woo.takeBus(bus600);
han.takeBus(bus600);
han.takeBus(bus600); //잔액 부족!

//정보 출력
woo.displayInfo();
han.displayInfo();
bus600.displayInfo();
```

```
잔액이 부족합니다!!
우영우 : 잔액 8500원
한강 : 잔액 500원
600번 버스 : 수입 3000원, 승객 2명
```