HANYANG UNIVERSITY

창의적 소프트웨어 프로그래밍
실습 교재_07

Overview

- 목표
  - C++ struct & class
  - Inheritance

C++ struct and class

```
#include <string>
#include <vector>

#include <iostream>

struct StudentInfo
{
   StudentInfo(int _id, std::string _name, int _score){
      id = _id;
      name = _name;
      score = _score;
   }

   int ShowInfo() const {
      std::cout << id << " | " << name << " | " << score << std::endl;

      return 0;
   }

   int id;
   std::string name;
   int score;
};
```

C++ struct and class

```
class StudentVector{
public:
    StudentVector(){}
    int AddStudent(const std::string& _name, const int _id, const int _score);
    bool CheckDuplicateId(int _id) const;
    int RemoveStudent(const std::string& str);
    int ShowStudents() const;
    int GetStudentByIndex(int num) const;

private:
    std::vector<StudentInfo> student_vec;
};
```

C++ struct and class

```
bool StudentVector::CheckDuplicateId(int _id) const {
   std::vector<StudentInfo>::const_iterator pos;

   for(pos = student_vec.begin() ; pos != student_vec.end() ; ++pos){
      if(pos->id == _id){
         return false;
      }
   }

   return true;
}


int StudentVector::AddStudent(const std::string& _name, const int _id, const int _score){
   if(CheckDuplicateId(_id) == false){
      std::cout << "[Error] Dulicated ID: " << _id << ", Name: " << _name << std::endl;
      return -1;
   }

   StudentInfo si(_id, _name, _score);
   student_vec.push_back(si);

   return 0;
}
```

C++ struct and class

```
int StudentVector::ShowStudents() const {
   std::vector<StudentInfo>::const_iterator pos;

   std::cout << "\nStudent List: " << std::endl;

   for(pos = student_vec.begin() ; pos != student_vec.end() ; ++pos){
      pos->ShowInfo();
   }

   return 0;
}

int main(){
   StudentVector sv;

   sv.AddStudent("Yuna Kim", 1, 100);
   sv.AddStudent("JS Han", 2, 99);
   sv.AddStudent("IW Ro", 2, 77);

   sv.ShowStudents();

   return 0;
}
```

Inheritance

- Inheritance? 상속?
  - 재산 상속 보다는 집합을 떠올리면 좋다
    - 상속의 반대는 피상속이 아니라 파생(Derived)이다
  - 5 원칙
    - SRP: 하나의 클래스는 하나의 책임을 갖는다
    - OCP: 클래스는 변경에는 닫혀 있고, 확장에 대해서는 열려 있다
    - LSP:A를 상속하는 B가 있다면, A는 B로 항상 대체 가능하다
    - ISP: 자신과 상관 없는 내용에 대해서는 상속 받지 않아야 한다
    - DIP: 상위 계층과 하위 계층 사이에 의존성이 없어야 한다

Inheritance

```
struct StudentInfo
{
public:
   StudentInfo(int _id, std::string _name, int _score){
      id = _id;
      name = _name;
      score = _score;
   }

   int ShowInfo() const {
      std::cout << id << " | " << name << " | " << score << std::endl;
      return 0;
   }

   int id;
   std::string name;

private:
   int score;
};
```

Inheritance

```
class StudentInfoEx : public StudentInfo
{
   std::string GetName();
   int GetScore();
};

std::string StudentInfoEx::GetName(){
   return name;
}

int StudentInfoEx::GetScroe(){
   return score;
}
```

```
example.cpp:25:9: error: 'int StudentInfo::score' is private
    int score;
        ^
example.cpp:39:12: error: within this context
    return score;
           ^
```