컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 50강 - C++의 캡슐화 기법

Fast campus

학습 목표

C++의 캡슐화 기법

1) C++의 캡슐화 기법의 원리에 대해서 이해하고 이를 활용할 수 있습니다.

C++의 캡슐화 기법

프렌드

C++에서는 기본적으로 멤버 변수에 접근하기 위해서 public 멤버 함수를 이용해야 합니다. 다만 특정한 객체의 멤버 함수가 아닌 경우에도 private 멤버에 접근해야 할 때가 있습니다. 이 때 프렌드(friend) 키워드를 이용하면 특정한 객체의 모든 멤버에 접근할 수 있습니다.

C++의 캡슐화 기법

프렌드 함수

프렌드(Friend) 함수는 기존의 함수 앞에 friend 키워드를 붙인 형태로 사용할 수 있습니다.

Fast campus

C++의 캡슐화 기법

프렌드 함수의 활용 사례

```cpp
#include <iostream>
#include <string>

using namespace std;

class Student {
private:
  int studentId;
  string name;
public:
  Student(int studentId, string name): studentId(studentId), name(name) { }
  friend Student operator +(const Student &student, const Student &other) {
    return Student(student.studentId, student.name + " & " + other.name); // friend 키워드를 이용해 이름에 접근
  }
  void showName() { cout << "이름: " << name << '\n'; }
};

int main(void) {
  Student student(1, "나동빈");
  Student result = student + student;
  result.showName();
  system("pause");
}
```

C++의 캡슐화 기법

프렌드 클래스

프렌드는 멤버 함수 이외에도 프렌드 클래스(Friend Class) 형태로 사용할 수 있습니다. 두 클래스가 서로 밀접한 연관성이 있으며 상대방의 private 멤버에 접근해야 한다면 클래스 자체를 프렌드로 선언할 수 있습니다.

프렌드 클래스에서는 모든 멤버 함수가 특정 클래스의 프렌드입니다.

C++의 캡슐화 기법

프렌드 클래스: 시간 클래스 정의하기

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string>
#include <ctime>

using namespace std;

class Time {
  friend class Date; // Date 클래스에서 Time 클래스를 이용할 수 있음.
private:
  int hour, min, sec;
public:
  void setCurrentTime() {
    time_t currentTime = time(NULL);
    struct tm *p = localtime(&currentTime);
    hour = p->tm_hour;
    min = p->tm_min;
    sec = p->tm_sec;
  }
};
```

C++의 캡슐화 기법

프렌드 클래스: 날짜 클래스 정의 및 사용해보기

```
class Date {
private:
  int year, month, day;
public:
  Date(int year, int month, int day): year(year), month(month), day(day) { }
  void show(const Time &t) {
    cout << "지정된 날짜: " << year << "년 " << month << "월 " << day << "일" << '\n';
    cout << "현재 시간: " << t.hour << ":" << t.min << ":" << t.sec << '\n';
  }
};

int main(void) {
  Time time;
  time.setCurrentTime();
  Date date = Date(2019, 12, 22);
  date.show(time);
  system("pause");
}
```

C++의 캡슐화 기법

정적 멤버

정적 멤버(Static Member)란 클래스에는 포함 되어 있는 멤버이지만 모든 객체가 공유하는 멤버입니다. 정적으로 선언된 멤버는 메모리 상에 오직 하나만 할당되어 관리됩니다.

정적 멤버를 public으로 선언하면 외부의 어떠한 클래스에서 접근이 가능하며, 오직 하나만 관리됩니다. 정적 멤버는 일반적으로 싱글톤 패턴(Singleton Pattern) 등의 다양한 기능을 위해 사용됩니다.

Fast campus

C++의 캡슐화 기법

정적 멤버

```
#include <iostream>
#include <string>

using namespace std;

class Person {
private:
  string name;
public:
  static int count;
  Person(string name) : name(name) {
    count++;
  }
};

int Person::count = 0;

int main(void) {
  Person p1("나동빈");
  Person p2("홍길동");
  Person p3("이순신");
  cout << "사람의 수: " << Person::count << '\n';
  system("pause");
}
```

C++의 캡슐화 기법

상수 멤버

상수 멤버(Constant Member)는 호출된 객체의 데이터를 변경할 수 없는 멤버를 의미합니다. 일반적으로 클래스에서 사용되는 중요한 상수는 상수 멤버 변수로 정의해서 사용하는 관행이 있습니다.

C++의 캡슐화 기법

상수 멤버

```
class Person {
private:
  const int id;
  string name;
public:
  static int count;
  Person(int id, string name) : id(id), name(name) {
    count++;
  }
  /*
  void setId(int id) {
    this->id = id; // 오류 발생 [수정 불가능]
  }
  */
};

int Person::count = 0;
```

배운 내용 정리하기

C++의 캡슐화 기법

1) C++에서 캡슐화를 위해 사용되는 프렌드, 정적 멤버 등의 개념을 바르게 숙지해야 객체 지향 프로그래밍 기법을 보다 자유롭게 활용할 수 있습니다.