
CODINGON
X
posco
K-Digital Training
스마트 팩토리 4기

업캐스팅 & 다운캐스팅

캐스팅 ??

- 타입을 변환하는 것 (a.k.a 형변환)

📌 업캐스팅

- 자식 클래스의 객체가 부모 클래스 타입으로 형변환 되는 것
- 부모 클래스의 포인터로 자식 클래스 객체를 가리키는 것

📌 다운캐스팅

- 업캐스팅된 것을 다시 원상태로 돌리는 것

업캐스팅 & 다운캐스팅

📌 업캐스팅

```
Person *p = new Student(); // 업 캐스팅
```

📌 다운캐스팅

```
Student *stu = (Student *)p; // 다운 캐스팅
```

실습1

(1) 앞 실습 포인터 변수로 만들어서 처리하기

(2) 1번 끝나면 업캐스팅 했던 것들 다운캐스팅도 해보기

가상 함수

가상 함수??

- 가상함수는 부모 클래스에서 상속받을 클래스에서 재정의할 것으로 기대하고 정의해 놓은 함수

- 실행 중(런 타임)에 어떤 함수를 호출할 것인지 결정함.

  ➡ 동적 바인딩 (지연 바인딩)

- 단, 포인터나 참조를 통하여 호출될 때만 동적 바인딩을 함.

가상함수

```
class Person {
public:
    virtual void intro() {
    cout << "사람입니다~" << endl;
    }
};

class Student: public Person {
public:
    void intro() {
    cout << "학생입니다~" << endl;
    }
}
```

```
Student stu;
Person *p1= &stu;
p1->intro(); // 동적 바인딩
```

```
Person *p2= new Student();
p2->intro(); // 동적 바인딩
delete p2;
```

실습2 가상함수 실습

```cpp
class Person {
public:
    virtual void intro() {
        cout << "사람입니다~" << endl;
    }
};

class Student : public Person {
    string name;

public:
    Student(string name) {
        this->name = name;
    }
    void intro() {
        cout << name << "학생입니다." << endl;
    }

    void learn() {
        cout << "배웁니다." << endl;
    }
};
```

```cpp
class Teacher : public Person {
    string name;

public:
    Teacher(string name) {
        this->name = name;
    }
    void intro() {
        cout << name << "선생입니다." << endl;
    }

    void teach() {
        cout << "가르칩니다." << endl;
    }
};
```

실습2 가상함수 실습

```
int main() {
    Person* pList[3];
    string names[3];

    cout << "3명의 이름을 입력해주세요.(선생님, 학생, 학생)" << endl;
    cin >> names[0] >> names[1] >> names[2];

    /* names[] 배열 이용하여 각 class 생성 */
    // write your code

    /* pList에 할당하는 코드 추가 */
    // write your code

    for (auto p : pList) {
        p->intro();
    }

    /* 각 class의 고유 함수 실행 (teach(), learn(), learn()) */
    // write your code
}
```

static 멤버

static(정적) 멤버 ??

- 클래스에는 속하지만, 객체 별로 할당되지 않고 클래스의 모든 객체가 공유하는 멤버

[특징]

1. 객체와 독립적이다 => 객체를 생성하지 않아도 접근 가능!
2. 정적 메소드 안에서는 일반 멤버에 접근할 수 없음! (static 멤버에만 접근 가능)

static 멤버

```
class Person {
    static int count; // static 필드
    string name; // 일반 필드
public:
    static int get_count() { // static 메소드
        return count;
    }
}

int Person::count = 0; // 정적 멤버 변수의 정의 및 초기화
cout << Person::get_count() << endl;
```

실습3 static 멤버

(1) 간식 바구니 프로그램을 활용하여 아래 사진과 같은 프로그램 만들기.