**2020-09-07**

**20190269**

**박재우**

**Week05**

<https://github.com/jwoo9928/-Object-oriented-design/tree/master/week5/WEEK05_CODE>

과제 1

1. 코드 및 실행화면

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 과제 수행 과정

기존 복사 방법은 얕은 복사 이므로 복사될 이름의 크기로 새로운 이름의 크기를 할당해준뒤에 changename의 함수로 새로운 주소의

이름을 할당한다.

1. 결과분석

과제에서 원하는대로 A는 Brwon Age: 22, B는 Jenny Age: 10으로 잘

출력된것을 확인할 수 있다.

1. 새로 알게 된 점

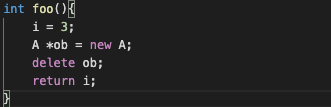
복사생성자에 대해서 새로 알게 되었다.

또한 복사생성자를 작성시 얕은 복사인지 깊은 복사인지를 잘

판단해서 구현해야된다는 점을 알게 되었다.

과제 2

1. 코드 및 실행화면



텍스트이(가) 표시된 사진

자동 생성된 설명

1. 과제 수행 과정

소멸자를 실행하기 위해서 Class A로 선언된 객체 ob를 new A로 동적 할당을 해준뒤에 delete ob를 선언해서 소멸자를 실행해준다.

1. 결과분석

동적할당 뒤에 소멸자를 실행하므로서 I = 10이 되도록 했으며,

출력결과에서 보듯이 아무문제없이 과제에서 원하는 바를 실행했다.

1. 새로 알게 된 점

소멸자에 대해서 새로 알게 되었으며 소멸자는 객체가 동적할당 되지 않는 이상 실행할 수 없다는 것을 알게 되었다.

과제 3

1. 코드 및 실행화면

Example01.cpp & Example02.cpp



텍스트이(가) 표시된 사진

자동 생성된 설명

1. 이동생성자에 대해 이해한 것

이동생성자는 성능향상을 위한 목적으로 사용한다.

이동생성자의 파라미터는 오른쪽값 참조이다.

코드 실행시 임시객체가 컨테이너로 옮겨지면서 이동생성자가 호출된다. 그후 임시객체가 제거 되면서 소멸자를 호출하며 객체 정보를 출력한다. 컨테이너에 있던 객체가 소멸하면서 소멸자를 호출한다.

이동생성자를 정의하지 않으면 이동생성자가 호출되지 않고

복사생성자가 호출된다. 이 경우 벡터가 복사될 때 마다

copy constructor가 호출되어 성능저하를 일으킨다.