**Class Declaration**

1. **클래스 선언시 자동으로 선언되는 일시적 멤버 메서드**- 기본 생성자  
   - 복사 생성자  
   - 복사 대입 연산자  
   - 기본 소멸자
2. **얕은 복사, 깊은 복사**

**class** **Person** {

int age;

char \*name;

**public**:

Person(int age, char \*name) { *// 생성자*

**this**->age = age;

**this**->name = **new** char [10]; *// 동적 메모리 할당*

strcpy(**this**->name, name);

}

~Person() { *// 소멸자*

**delete**[] name; *// 동적 할당된 메모리 해제*

}

};

int main() {

Person p1(20, "Mike");

Person p2(p1);

}

*/\**

*Person(Person& p) { // 디폴트 복사 생성자 (얕은 복사)*

*this->age = p.age;*

*this->name = p.name; // 메모리 주소값(포인터) 그대로 복사*

*}*

*Person(Person& p) { // 깊은 복사 생성자 명시적 작성*

*this->age = age;*

*this->name = new char [10]; // 새로운 메모리를 동적으로 할당하고*

*strcpy(this->name, p.name); // p.name의 값을 복사*

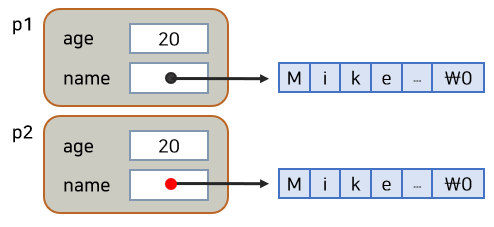
*}*

*\*/*

- 얕은 복사로 모든 멤버를 복사하면 p1.name의 메모리 주소가 p2.name에 그대로 복사  
- 같은 메모리 공간을 공유하게 되는데, 소멸자가 실행될 때 문제가 발생한다.  
- p2.name이 가리키는 주소에 할당된 메모리가 먼저 해제된다.  
- p1.name이 가리키는 주소에 할당된 메모리가 이미 해제된 상태 메모리 접근 에러가 발생한다.

텍스트, 장치, 측정기이(가) 표시된 사진

자동 생성된 설명  
- 깊은 복사는 객체가 복사될 때 새로운 메모리 주소를 할당하여 값만 복사



1. **복사 생성자, 대입연산자 재정의 이유**- 디폴트 복사 생성자, 대입 연산자는 얕은 복사를 기본적으로 사용한다.  
   - 하지만, 위의 문제처럼 멤버 변수에 포인터가 존재하는 경우  
   - 하나의 메모리를 여러 객체가 공유하는 상황이 발생하여 문제가 될 수 있으므로  
   - 명시적으로 복사 생성자, 대입연산자를 작성하는 것이 사전에 오류를 방지할 수 있다.