10-3, 4. 값 타입과 불변 객체, 값 타입 비교

임베디드 타입 같은 값 타입을 여러 엔티티에서 공유하면 위험하다.

```
Address address = new Address("city", "street", "10000");

Member member1 = new Member();
member1.setName("member1");
member1.setAddress(address);
em.persist(member1);

Member member2 = new Member();
member2.setName("member2");
member2.setAddress(address);
em.persist(member1);

member1.getAddress().setCity("newCity");
```

멤버1만 바꿀라했는데 2도 바뀐다..

이런 것을 의도하려면 Address도 엔티티로 만들어서 쓰자!

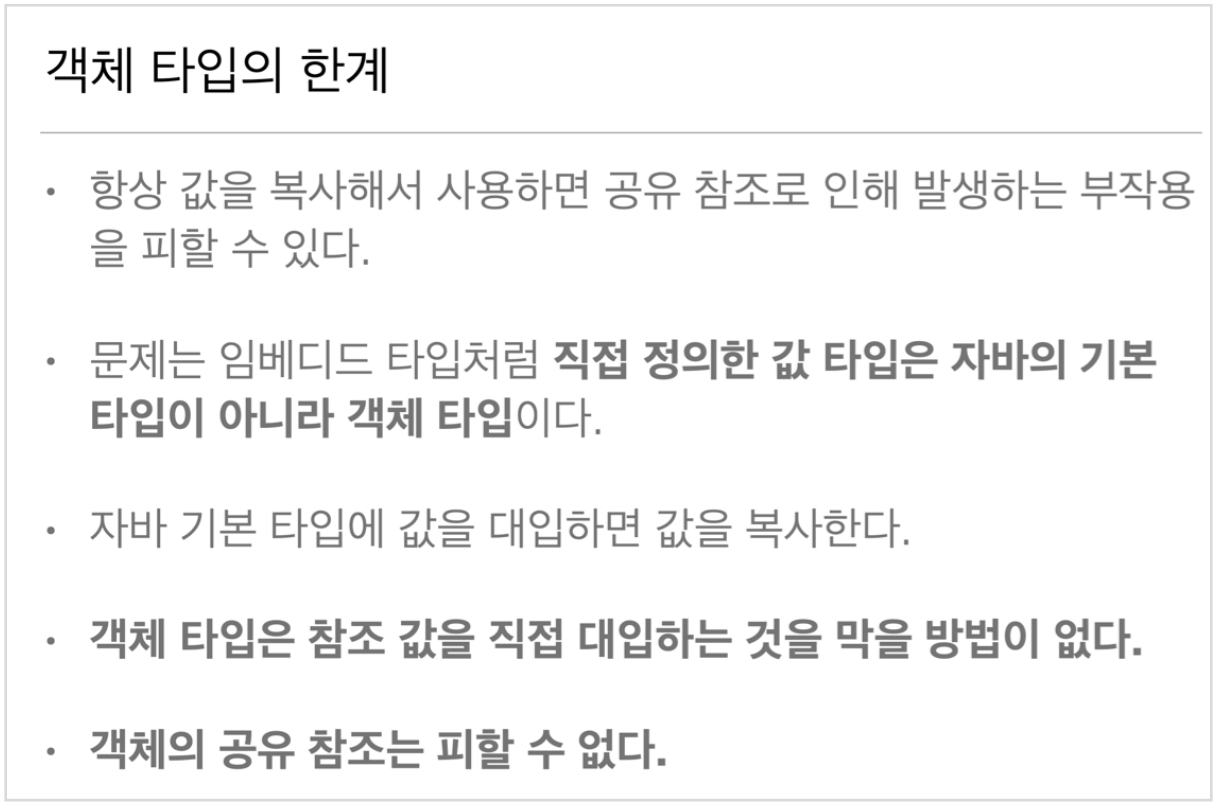
값 타입 복사

위처럼 값 타입을 쓰고싶다면 복사해서 쓰자.

```
Address address = new Address("city", "street", "10000");

Member member1 = new Member();
member1.setName("member1");
member1.setAddress(address);
em.persist(member1);

Address copyAddress = new Address(address.getCity(), address.getStreet(),
```

```
address.getZipcode());

Member member2 = new Member();
member2.setName("member2");
member2.setAddress(copyAddress);
em.persist(member2);

member1.getAddress().setCity("newCity");
```

객체의 공유 참조를 피할 수 있나?



기본 타입은 = 연산자로 복사가 되지만 객체는 레퍼런스 자체가 복사되기 때문에 공유 참조는 피할 수 없다!

불변 객체

생성 시점 이후에는 절대로 값을 변경하지 못하게 만든 객체.
생성자로만 값을 설정하고 setter를 만들지 않으면 가능함.

그럼에도 불구하고 값을 바꾸고 싶다면?
아예 새로운 주소를 만들어서 엔티티의 setter로 바꿔주면 됨.

값 타입 비교

임베디드 타입 Address로 두 개를 만들었을 때 ==비교하면 당연히 틀리다고 나온다.
왜냐면 객체== 비교에서는 참조 값을 비교하기 때문.

- 동일성 비교(identity)
- 동등성 비교(equivalence) ➜ 이게 정석인 방법
  - equals()로 비교해야 하는데, 이걸 오버라이딩 해야한다.
  - 오버라이딩은 cmd+n을 이용하여 자동으로 만들어주는걸 쓰는게 낫다!

#JPA