

<상속>

Super()로 호출한 상위클래스의 생성자는 default 생성자임

항상 상위 클래스를 만들고 하위 클래스를 만들어야한다.

하위클래스는 super()를 통해 상위클래스의 default 생성자를 불러온다.

상위클래스에 어떠한 생성자도 없는 경우 default 생성자를 컴파일러가 자동 생성하기에 문제되지 않고, default 생성자를 직접 생성해 주어도 문제되지 않지만, default가 아닌 생성자가 있는 경우, 하위클래스에서는 클래스 생성 후 프리컴파일을 해 줘야한다.

프리컴파일을 하는 방법은 하위 클래스를 만들고 거기 생기는 빨간 선을 클릭하면된다.

<메소드 오버라이딩>

\* 메소드를 덮어쓴다.

\* 상위클래스의 메소드가 하위클래스에서 쓰고싶어하는 메소드랑 다를 경우 하위클래스에서는 상위클래스에서의 메소드와 똑 같은 이름의 메소드를 정의할 수 있다는 것!

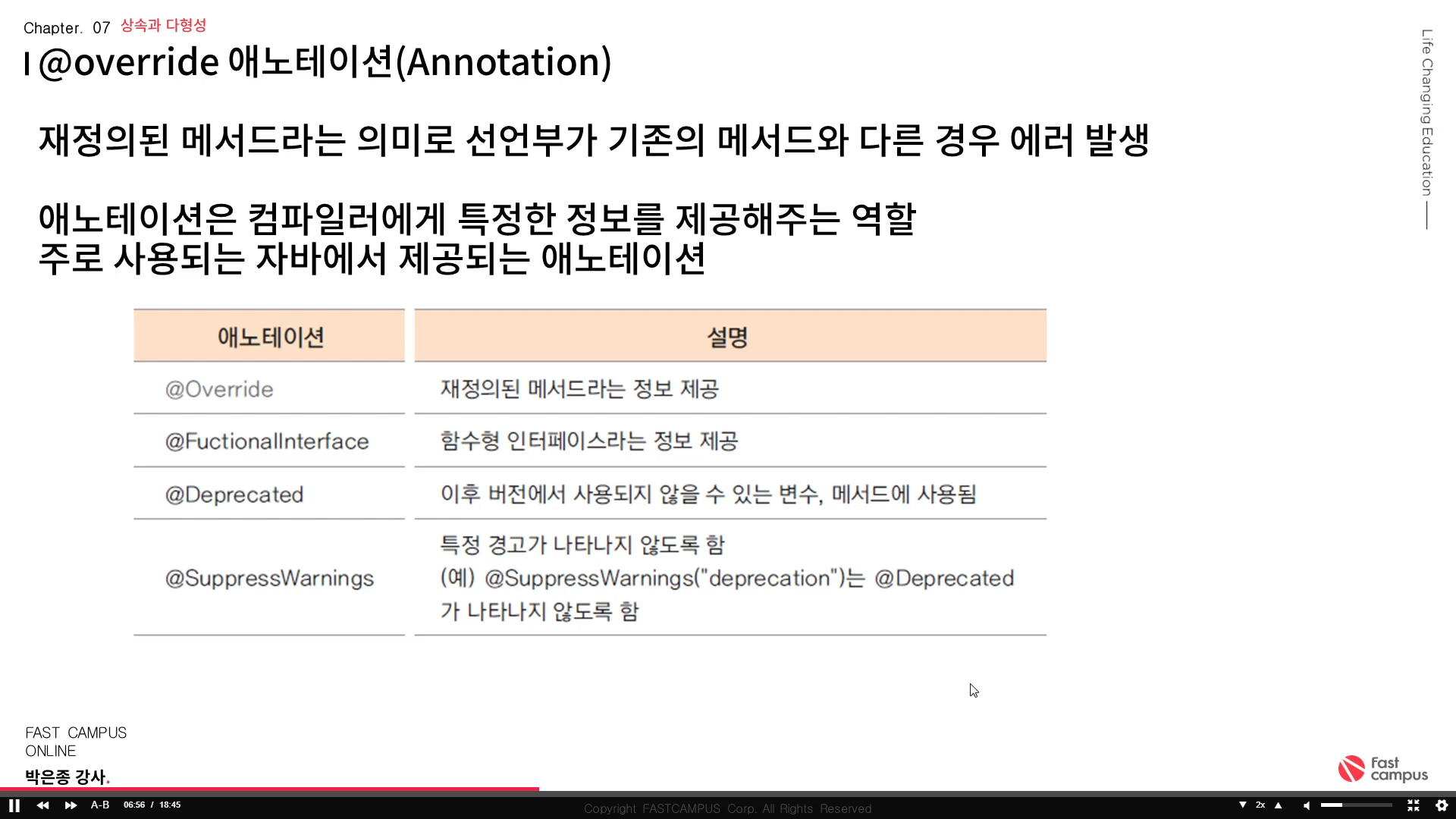
\* 메소드를 오버라이드할 하위의 클래스에서 >마우스 우클릭>Source>Override어쩌구 > 오버라이드할 메소드를 누르고 OK하면 어노테이션을 가진 오버라이드한 메소드가 생성된다.

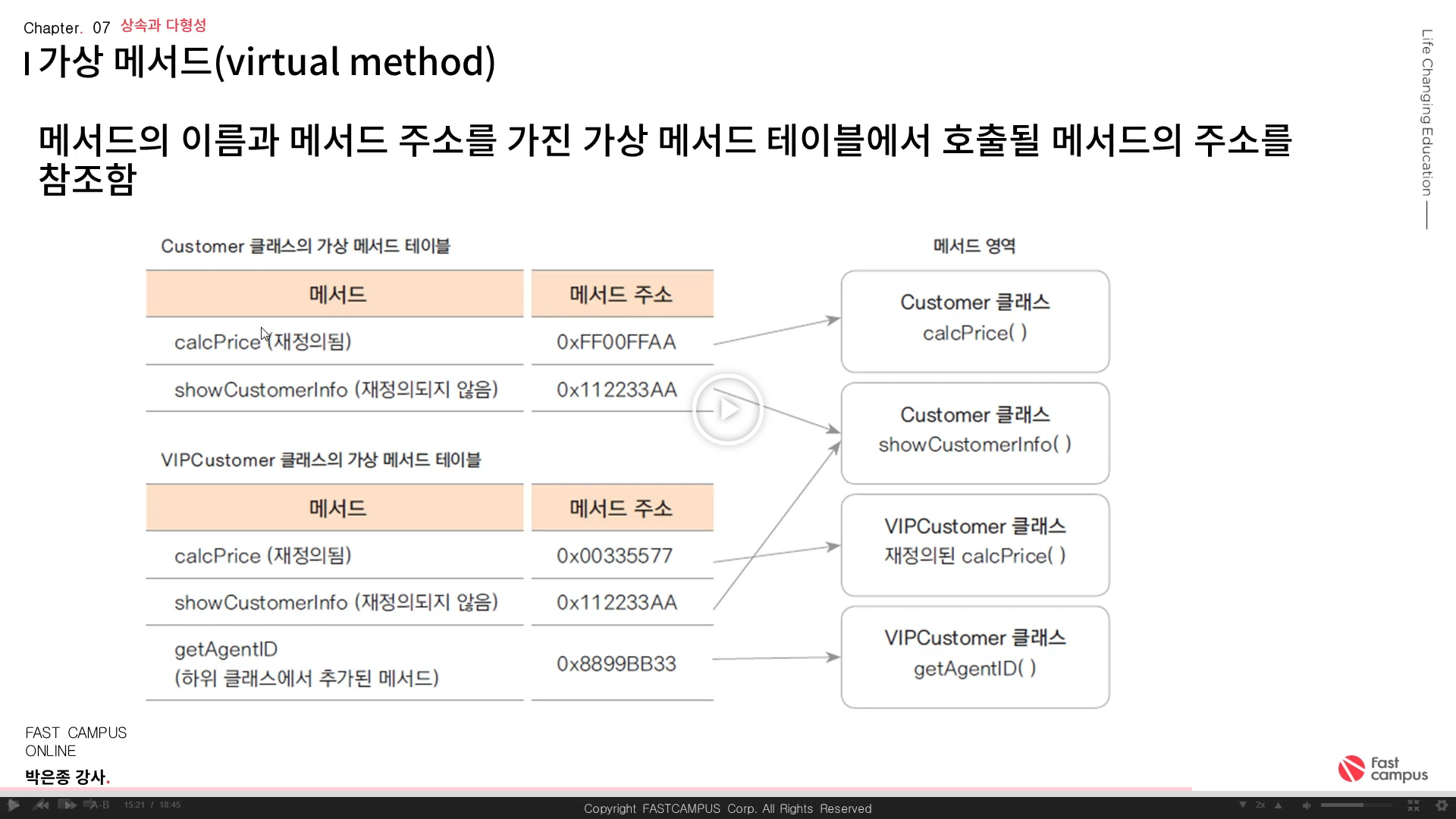
\*오버라이딩한 하위클래스의 함수는 상위클래스의 오버라이드된 함수와 선언부가 다르면, 에러가 난다. 반드시 선언부를 같게 해줘야한다. 선언부를 같게 해줘야한다는 것은

**public** **int** calcPrice(**int** price) {

int price의 자료형을 같게 해주는걸 주의하란거다.

이게 메소드 오버로딩과의 차이인데, 메소드 오버로딩은 함수의 이름은 같은데 받아오는 파라미터가 다른거다.

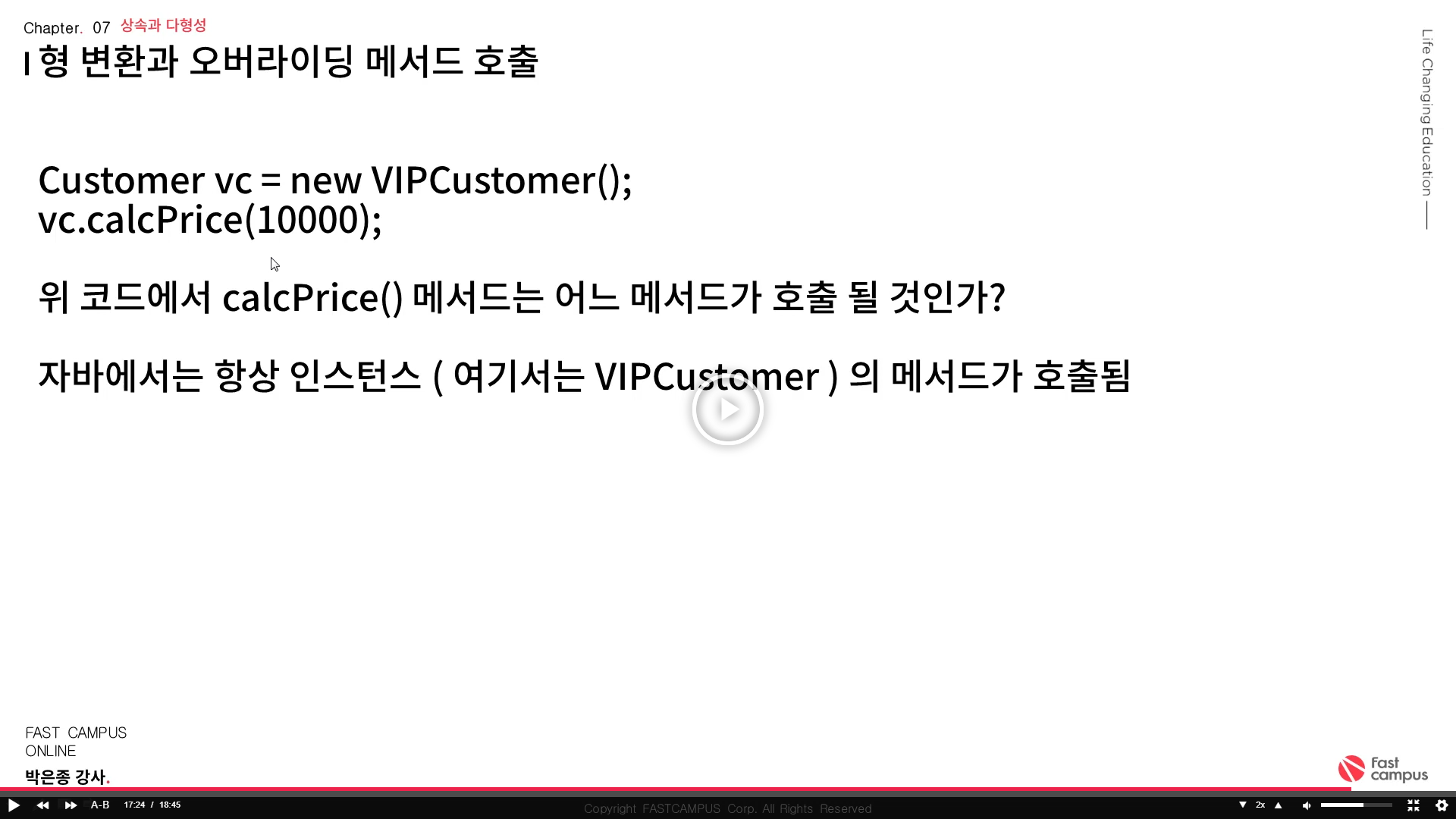




즉 상위클래스에서 재정의된 함수는 상위클래스의 메서드 영역을 차지하는 것이고,

하위클래스에서 재정의된 함수는 하위클래스의 메서드 영역을 차지하는 것이다.

재정의되지 않은 함수는 상위 클래스의 메서드 영역을 차지한다.



따져 생각해야 할 것 :

1. 위 코드와 같은 경우, 항상 하위클래스의 메서드가 호출되지만
2. calcPrice함수와같이 재정의가 된 경우, 하위클래스인 VIPCustomer의 함수가 호출된다.

