

하위클래스로 변환되어야 하위클래스의 메소드를 사용할 수 있는 경우 다운캐스팅을 한다.

**package** polymorphism;

**import** java.util.ArrayList;

**class** Animal {

**public** **void** move() {

System.***out***.println("동물이 움직입니다.");

}}

**class** Human **extends** Animal {

**public** **void** move() {

System.***out***.println("사람이 두 발로 걸어갑니다.");

}

}

**class** Tiger **extends** Animal {

**public** **void** move() {

System.***out***.println("호랑이가 네 발로 걸어갑니다.");

}

**public** **void** hunting() {

System.***out***.println("호랑이가 사냥을 합니다.");

}

}

**class** Eagle **extends** Animal {

**public** **void** move() {

System.***out***.println("독수리가 날아갑니다.");

}

**public** **void** flying() {

System.***out***.println("독수리가 퍼덕퍼덕 날아갑니다.");

}

}

**public** **class** AnimalTest {

**public** **static** **void** main(String[] args) {

//아래에 인스턴스화하는걸 업캐스팅이라한다.

Animal hAnimal = **new** Human();

Animal tAnimal = **new** Tiger();

Animal eAnimal = **new** Eagle();

//2.이렇게 생성될 수 있는 하위 객체의 종류들을 고려했을 때,

//AnimalTest test = new AnimalTest();

//test.moveAnimal(hAnimal);

//test.moveAnimal(tAnimal);

//test.moveAnimal(eAnimal);

/\*다운캐스팅 주석\*/

/\*Human, Tiger, Eagle을 모두 Animal 타입으로 유지하고싶다면 ArrayList의 타입을 Animal로 하면된다.\*/

ArrayList<Animal> animalList = **new** ArrayList<Animal>();

animalList.add(hAnimal);

animalList.add(tAnimal);

animalList.add(eAnimal);

/\*이렇게하면 각Animal의 move가 아닌 각 인스턴스의 move가 호출된다.\*/

**for**(Animal animal : animalList) {

animal.move();

}

}

//1. moveAnimal이라는 AimalTest 클래스의 메소드를 만들었다.

//public void moveAnimal( animal) { 매개변수에 타입으로 들어갈 수 있는 것은

//3.매개변수의 타입은 Animal이 된다.

**public** **void** moveAnimal(Animal animal) {

animal.move();

}

}

Tiger 클래스의 hunting() 메소드, Eagle 클래스의 flying() 메소드는 각 클래스에만 있고 상위 클래스에는 없는 메소드이기 때문에 상위 클래스 타입으로는 사용할 수 없다.

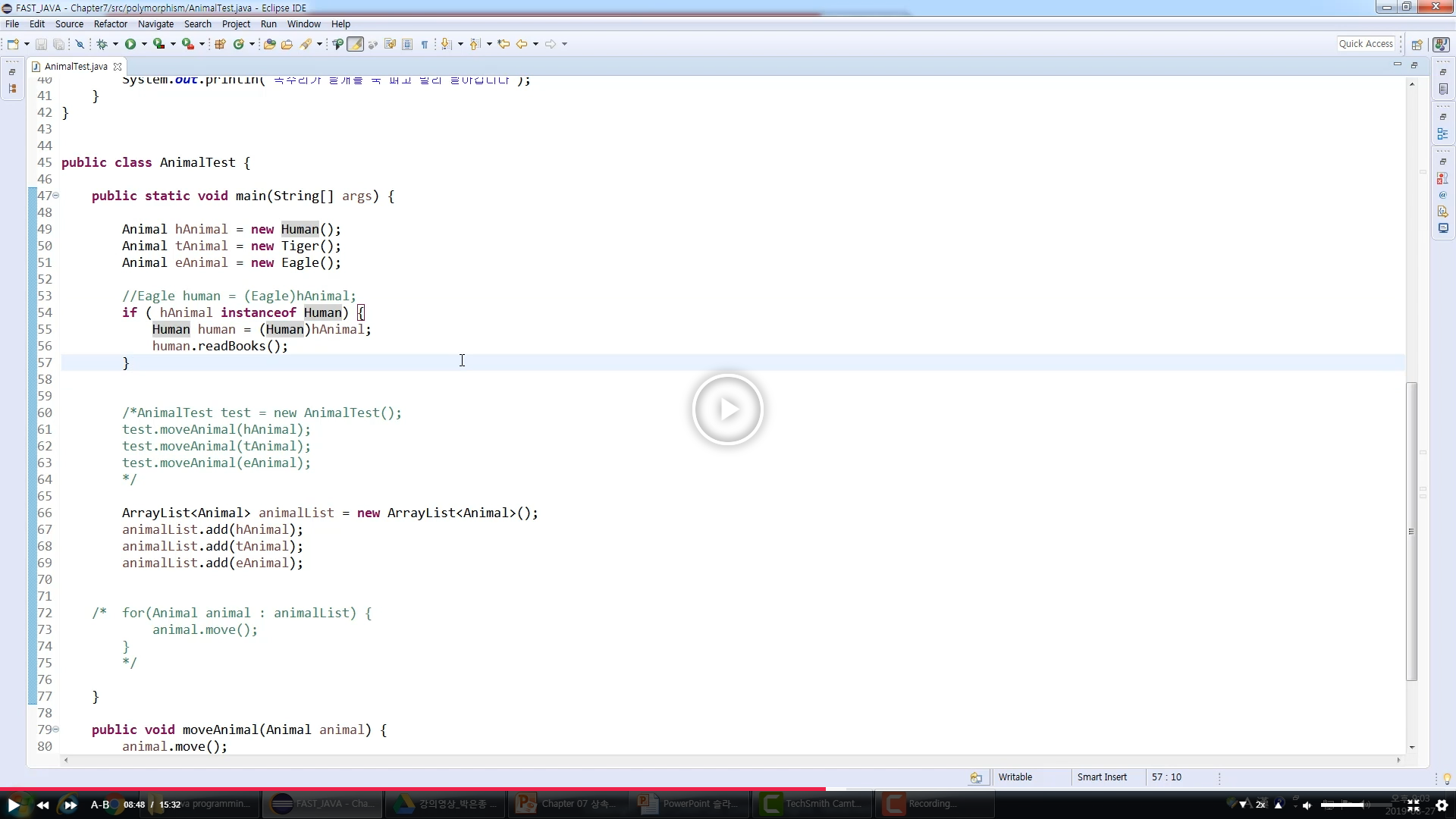
이 경우,

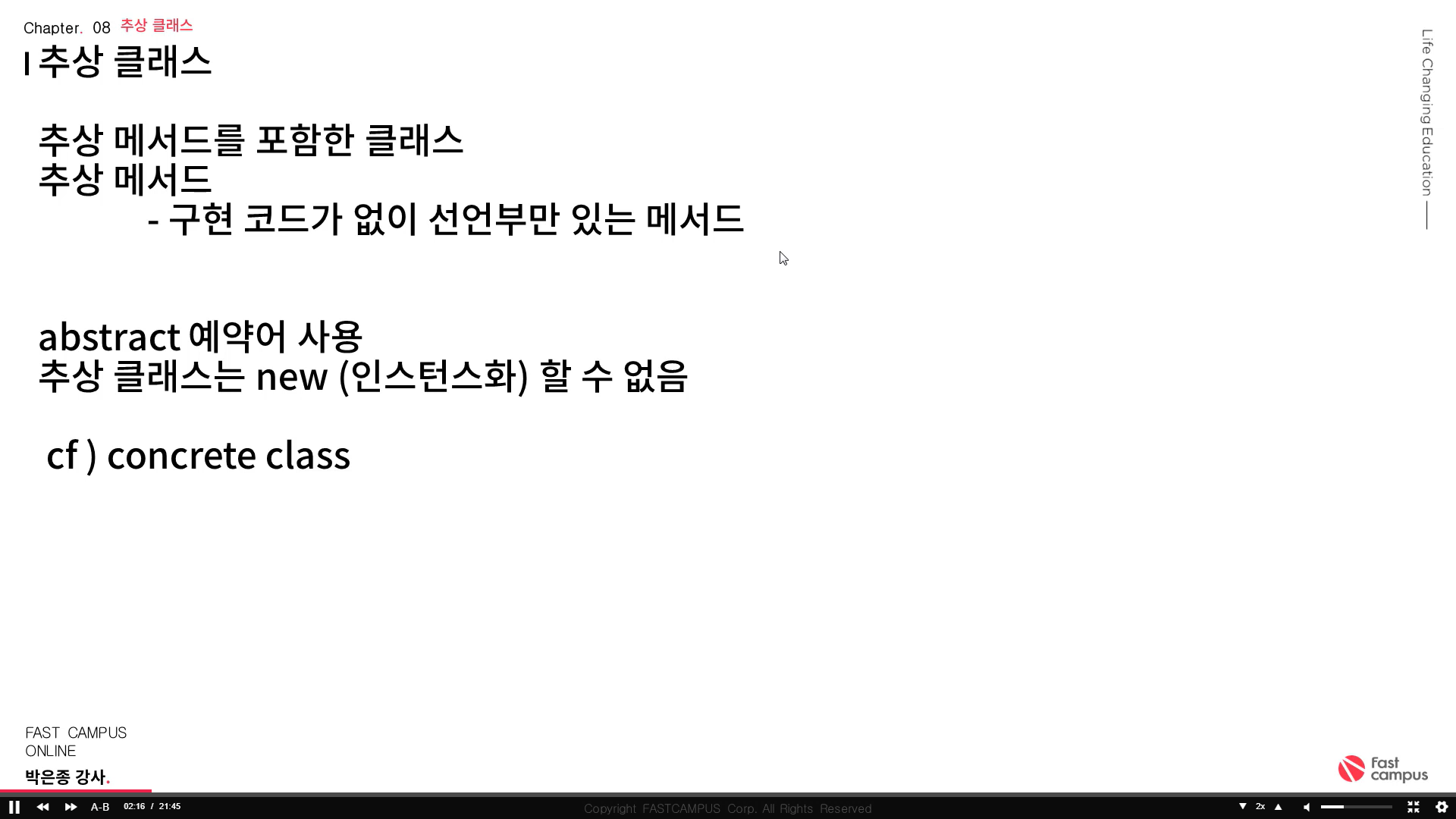
Tiger tiger=(Tiger)tAnimal;

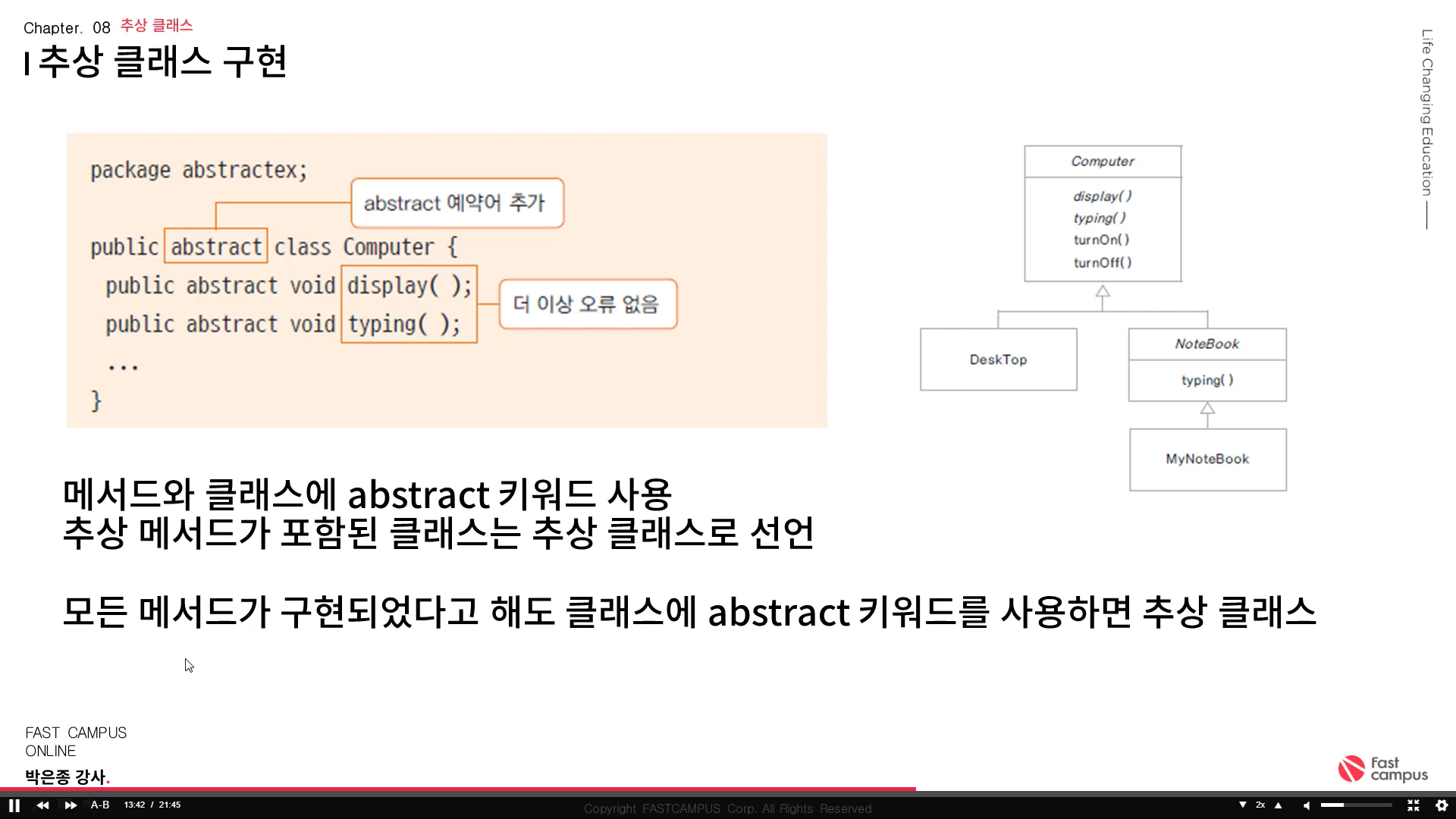
tiger.hunting();

이렇게 다운캐스팅을 해야한다.

안정적인 형변환을 하기위해서는 hAnimal이 Human의 객체인지 알기위해 instanceof를 사용하여 형변환을 한다. 원래 형으로 반환받기위한 방법이다.

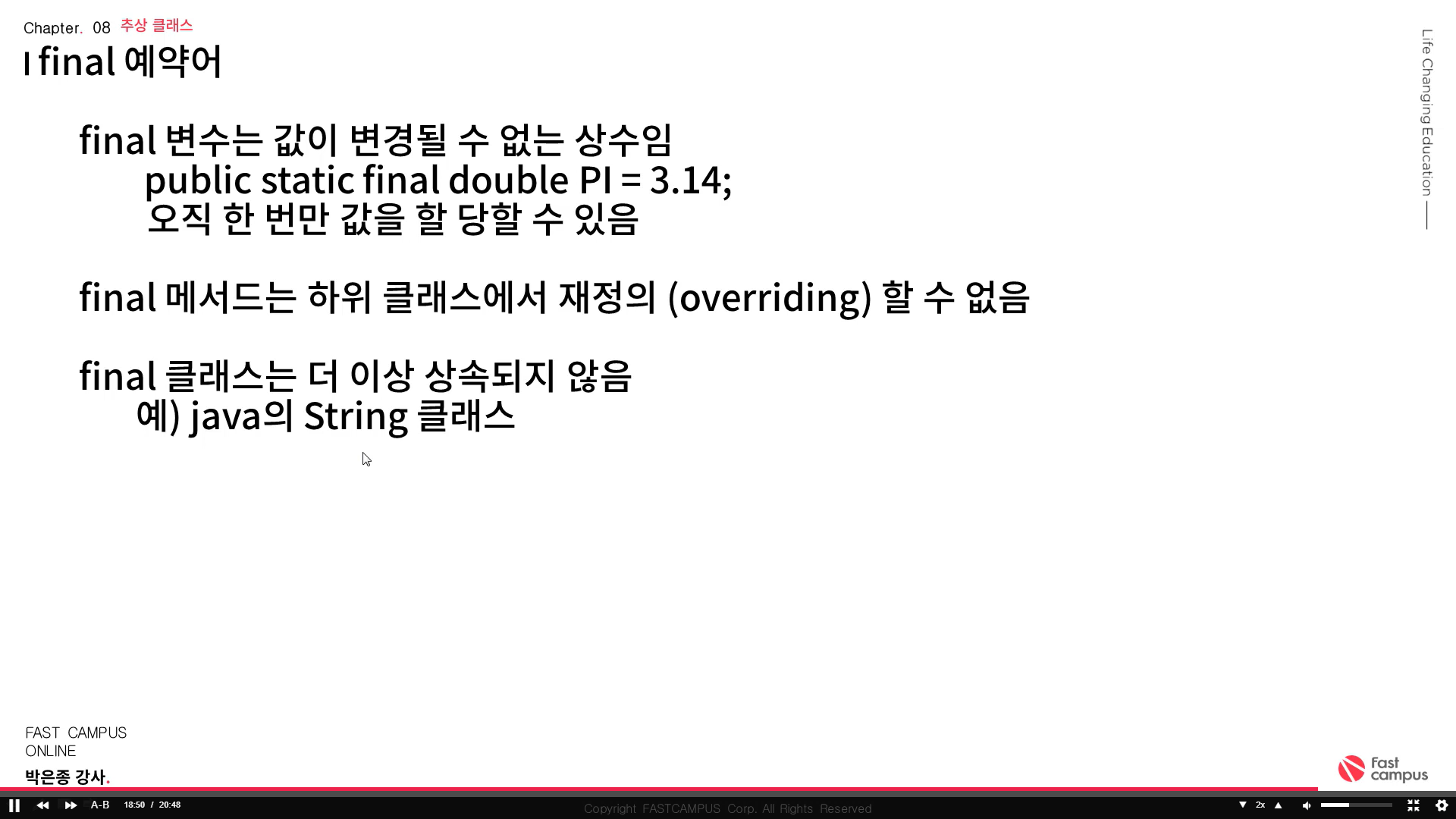


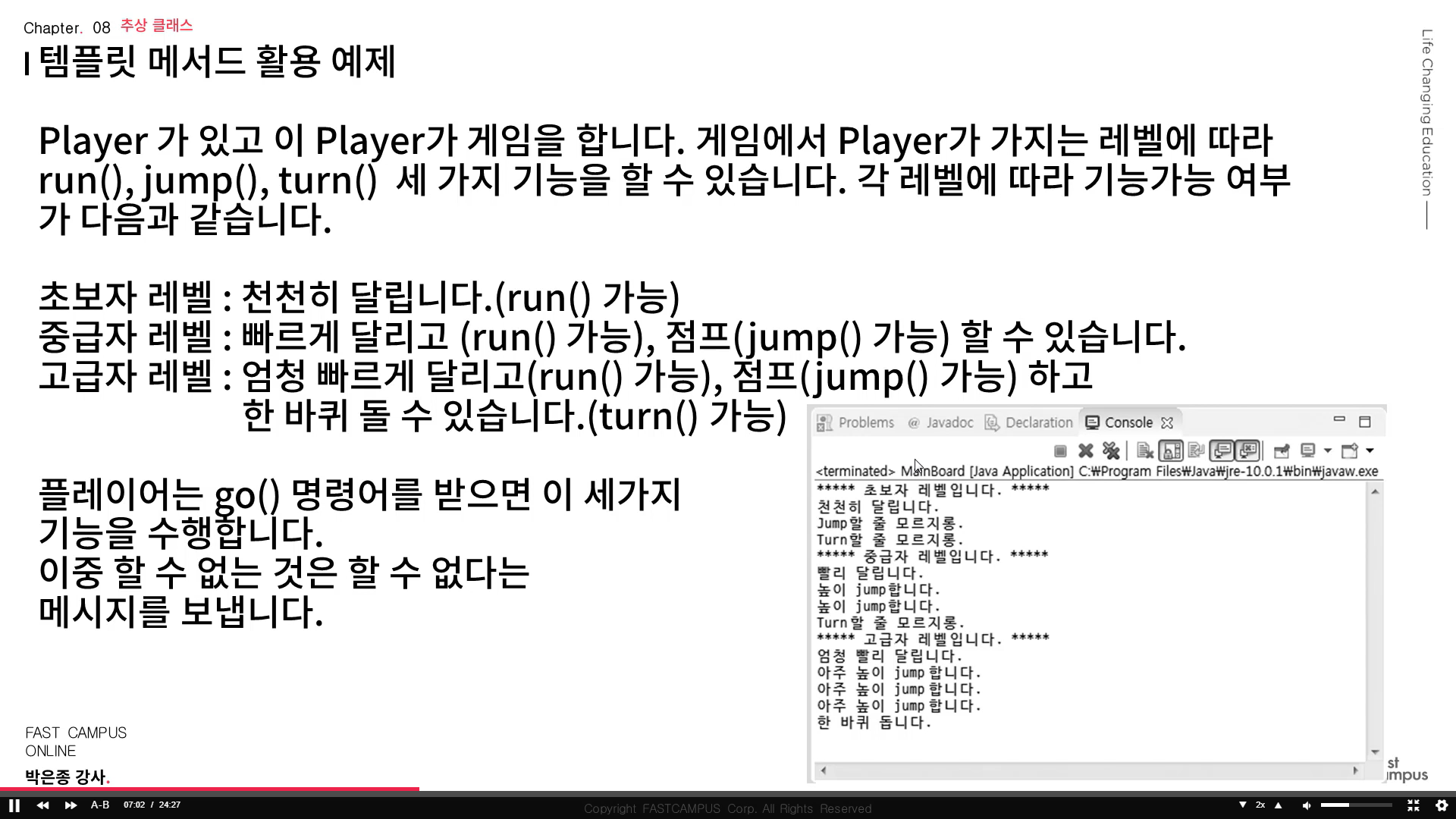




추상 클래스는 new할 수 없다.

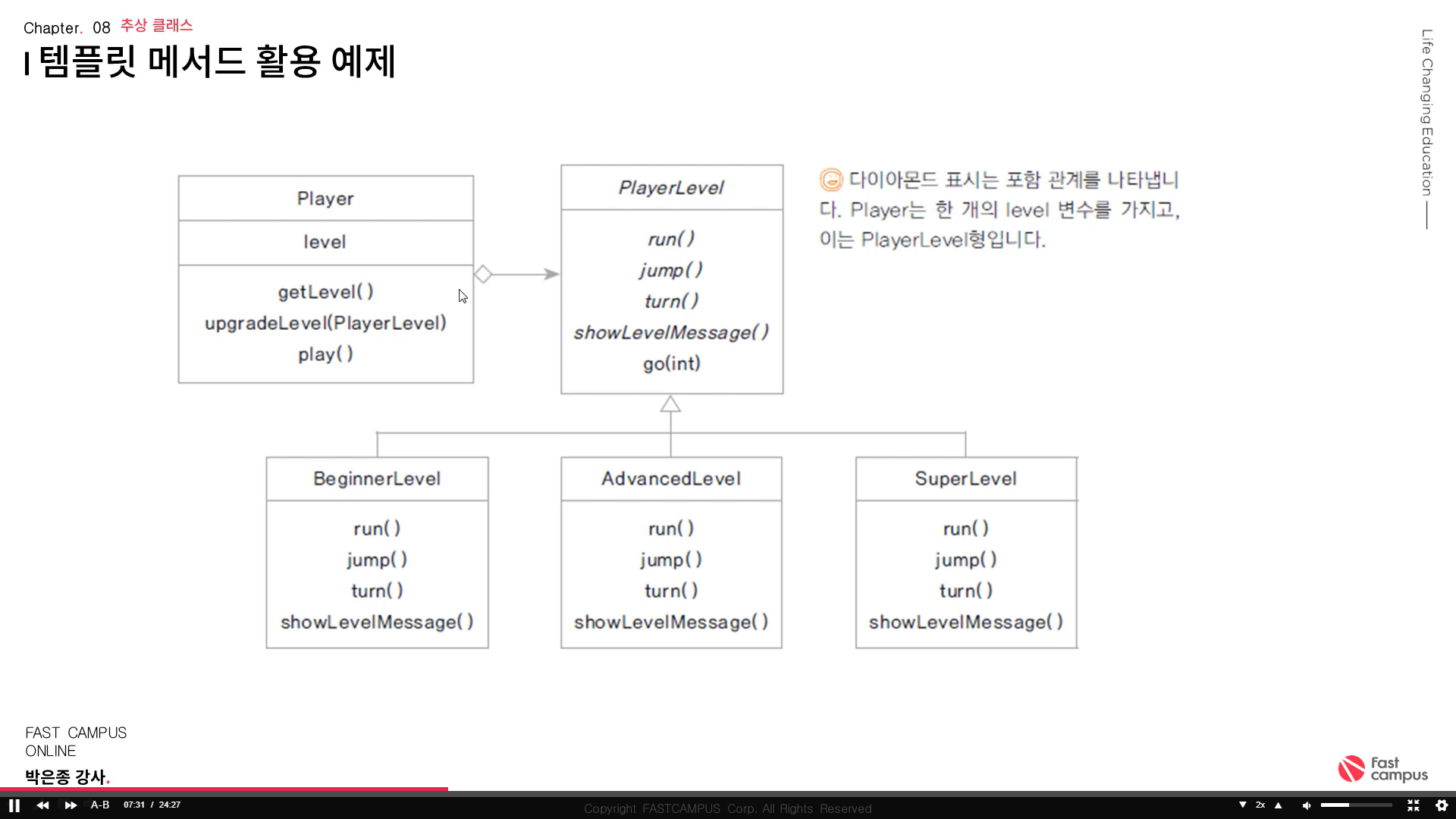
추상메소드는 이탤릭체로 표현한다.





>>레벨이 여러가지니까 ‘상속’을 써야겠구나.

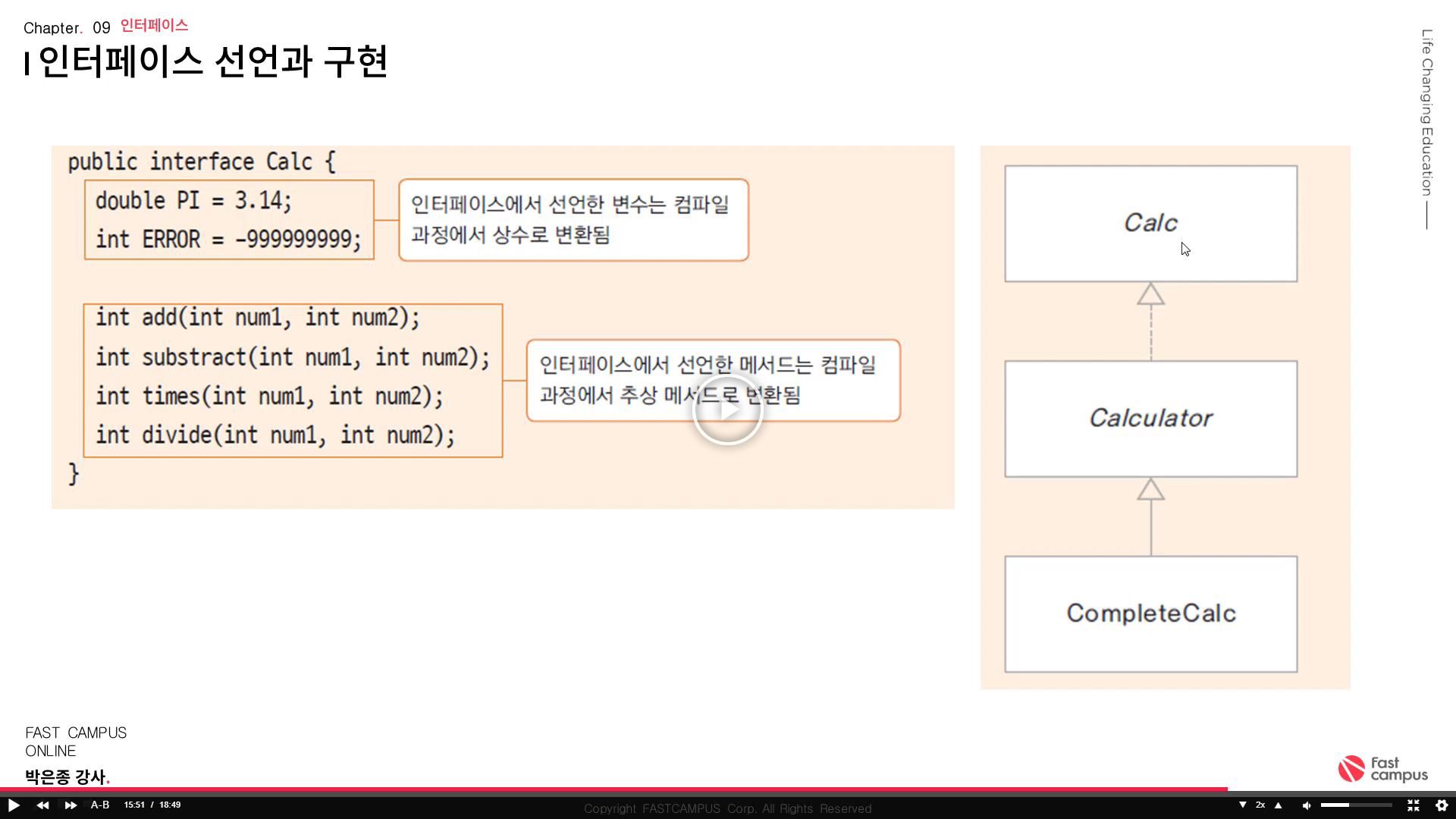
>> Player에는 레벨이 포함되어야겠구나



인터페이스는 추상메소드로만 구성되어있는 클래스이다.

인터페이스는 모든 변수가 상수이다.

어떤 역할을 한 객체라는 설명을 해 주는 역할을 하는게 인터페이스임



CompleteCalc는 Calculator가 상속받고 Calc를 구현되어있다고 한다.

