60200307 우연정

목차

1. Object Instantiation(객체 추상화)

Student student; <- 오브젝트의 이름 만들기(주소를 가리키고 있다.)
student = new Student(); <- 오브젝트의 바디 만들기(메모리에 실제로 만들어진 객체를 할당함, Binding)

객체를 추상화하려면 객체를 쓸 수 있게 이름과 바디가 만들어져야 한다. 처음에 오브젝트의 이름을 만들고(Object Name Declaration), new를 해서 바디를 만들면(Object Body Allocation) 실제로 메모리에 객체를 할당한다. 그 후로 그 이름으로 객체를 쓸 수 있다. (Object Name and Body Binding)

2. Strongly-Typed and Weakly-Typed Language

Strongly-Typed Language

Rectangle shape;
shape = new Rectangle();
shape = new Oval(); // error

Weakly-Typed Language
shape = new Rectangle(); shape = new Oval();

Strongly-Typed Language는 이름이 미리 타입을 한정시킨다. Weakly-Typed Language는 어떤 타입으로 쓰는지에 대한 정의가 없다(이름이 없다).
강한 타입을 가진 언어들은 이름을 한정하고 쓰기 때문에 에러가 잘 나지 않는 장점이 있지만 번거롭다는 단점이 있다. 또한, 다형성을 가진 객체를 만들지 못한다.
Weakly-Typed Language는 다형성을 기본적으로 지원한다. (여러 가지 형태로 쓸 수 있다.) Weakly-Typed Language에서는 컴파일러가 코드를 만들 때부터 이미 객체가 binding 된다.

Strongly-Typed Language와 Weakly-Typed Language의 차이는 이름을 미리 정의하느냐(타입을 미리 확정해서 제약해서 쓰는 것)의 차이이다. 이름이 여러 가지 클래스들과 binding 될 수 있게 만드느냐의 차이이다.

3. Polymorphism(다형성)
다형성(Polymorphism)은 객체 지향 프로그래밍의 핵심 원칙 중 하나로, 하나의 인터페이스나 부모 클래스를 가진 여러 객체들이 해당 인터페이스나 부모 클래스의 메

서드를 다르게 동작하게 하는 기능을 말한다.

일반화된 클래스(상위 클래스)에 정의된 객체의 이름이 특수화된 클래스(하위 클래스)의 인스턴스와 바인딩시켜도 에러가 나지 않는다. 하위 클래스는 부모 클래스의 메서드를 오버라이드(재정의)하여 그 동작을 대체할 수 있다. 오버라이딩하기 위해서는 시그니처(함수 이름)가 같아야 한다.