|  |
| --- |
| 객체 지향 문법 정리  2015년 4월 29일 수업 내용 |
| 통합ICTO팀 이영주 사원 |

**객체 지향(OOP)이란 무엇인가?**

객체(Object)란 물리적으로 존재하거나 추상적으로 생각할 수 있는 것 중에서 자신의 속성을 가지고 있고 다른 것과 식별 가능한 것이다. 객체는 속성과 동작으로 구성되는데 자바는 이 속성들을 필드(field), 동작들을 메소드(method)라고 부른다.

**객체 지향의 장점**

프로그램을 **유연하고 변경이 용이**하게 만들기 때문에 대규모 소프트웨어 개발에 많이 사용. 프로그래밍을 더 배우기 쉽게 하고 소프트웨어 **개발과 보수를 간편**하게 하며, 보다 **직관적인 코드 분석**을 가능하게 한다.

**클래스를 사용하지 않는 비(非)객체 지향적인 코드**

비슷한 내용의(같은 종류) 데이터를 담는 변수들을 일일이 선언하고 사용하여서 변수 사용이 잦기 때문에 비경제적이다.

**객체 지향 프로그래밍의 특징**

1. **캡슐화(Encapsulation)**

* 객체의 필드, 메소드(method)를 하나로 묶고, 실제 구현 내용을 감추는 것. 외부 객체는 객체 내부의 구조를 알지 못하기 때문에 객체가 노출해서 제공하는 필드와 메소드만 이용할 수 있다.
* 캡슐화를 하는 이유 : 외부의 잘못된 사용으로 인해 **객체가 손상되지 않도록** 한다. 자바 언어는 캡슐화된 멤버의 노출 여부를 결정하는데 ‘**접근 제한자**(Access Modifier)’를 사용한다.

1. **상속(Inheritance)**
   * 상위(부모) 객체가 자기가 가지고 있는 필드, 메소드를 하위(자식) 객체에게 물려주어 하위 객체가 사용할 수 있도록 해주는 것.
   * 상속의 장점 : 상위 객체의 재사용으로 하위 객체를 쉽고 빨리 설계할 수 있도록 하고 반복된 코드의 중복을 줄여준다. 유지 보수 시간의 최소화.
2. **다형성(Polymorphism)**
   * 같은 타입이지만 실행 결과가 다양한 객체를 이용할 수 있는 성질.
   * 오버로딩과 오버라이딩이 다형성에 대한 대표적인 기능이다.

**클래스의 구성 멤버**

1. **필드**
   * 객체 고유 데이터, 부품 객체, 상태 정보를 저장. 선언 형태는 변수와 비슷하지만 필드를 변수라고 부르지 않는다. 변수는 생성자와 메소드 내에서만 사용되고 생성자와 메소드가 실행 종료되면 자동 소멸된다. 초기값이 지정되지 않은 필드들은 객체 생성 시 자동으로 기본 초기값으로 설정된다. 필드 타입 별로 기본 초기값은 다르다.
2. **생성자**
   * New 연산자로 호출되는 특별한 중괄호 { } 블록. 객체 생성 시 초기화를 담당. 메소드와 비슷하게 생겼지만, 클래스 이름으로 되어 있고 리턴 타입이 없다.
3. **메소드**
   * 객체의 동작에 해당하는 중괄호 { } 블록. 이름을 가진다. 메소드는 객체 간의 데이터 전달의 수단으로 사용된다. 외부로부터 매개 값을 받을 수도 있고, 실행 후 어떤 값을 리턴할 수도 있다.

**객체 지향 문법 핵심 Key Point 정리**

1. 내가 만든 클래스도 변수명(변수타입)이다. 이렇게 만들면 다양한 타입의 변수를 하나의 이름으로 관리할 수 있다.
2. equals를 override하는 이유 : 객체들의 내용 비교 가능
3. factory : 말 그대로 클래스 공장. 어떤 클래스를 사용할 지 정해서 그 클래스를 생성, 리턴하는 목적으로 사용한다.
4. 사용할 객체는 factory로부터 받아와서(DI) 사용한다. 의존형 주입(DI)
5. 오버라이딩(Overriding) 된 자원에 한해서만 다운캐스팅 하지 않아도 부모타입에서 자식타입의 함수를 호출할 수 있다. 이식성이 좋아진다는 장점이 있다.