**객체 지향 프로그래밍(OOP, Objective-Oriented Programming)**

* 객체 지향 프로그래밍이란 프로그램을 어떻게 설계해야 하는지에 대한 일종의 방법론
* 기존 절차 지향 프로그래밍에서는 데이터와 처리 방법으로 나누고 프로그램의 흐름에 따라 설계를 하는 반면, 객체 지향 프로그래밍에서는 여러 객체라는 단위로 데이터와 처리 방법을 하나의 덩어리로 묶어 각 객체들이 유기적으로 상호작용 하도록 설계한다.
* 절차 지향이 모든 요리 재료를 순서에 맞게 조리하여 나오는 일품요리라면, 객체 지향은 이미 조리된 각 재료들을 필요에 따라 조합하여 먹는 구절판 같은 것.
* 작은 문제점들을 해결하는 객체들을 만들고, 이를 조합해서 큰 문제를 해결하는 상향식(Bottom-up)방법을 택함.
* 이 후에, 필요한 객체만을 재사용함으로써 개발 기간과 비용을 줄이는 데도 이점이 있음.
* 현재 C++, C#, JAVA, Objective C 등이 객체 지향 언어로 사용 중.

**객체 지향 프로그래밍의 세가지 특징**

1. **캡슐화(Encapsulation)**

* 객체 지향 방식의 기본 개념
* 계층적으로 기능(메서드)와 특성(데이터, 변수)을 분류한 뒤 클래스(Class)라는 캡슐(Capsule)에 분류된 집단 별로 집어 넣는다.
* 이러한 클래스를 실체화(Instance)하면 객체(Object)를 만들 수 있다.

1. **상속(Inheritance)**

* 이전 언어들에서 사용하던 라이브러리보다 더 논리적이고 체계적으로 기능과 데이터를 계층적으로 분류해서 사용의 편의성을 도모
* 상속을 통해 자식 클래스는 부모 클래스의 특성과 기능을 그대로 이어받고 필요한 기능 일부분만을 변경하여 재정의가 가능하다.

**==[자바에서의 상속]===================================================**

1. 자바에서 상속을 받게 하려면 새로운 클래스 이름 뒤에 extends와 상속받고자 하는 클래스를 입력한다.

class A{}

class B extends A{}

class C extends B{}

1. 다중 상속이 불가능 하며 부모의 생성자는 상속 되지 않는다. 또한 부모 클래스가 가진 멤버변수와 메서드를 모두 상속 받으며, private 접근 제한자를 사용할 경우 접근 불가능 하다.
2. Static 메서드 또는 변수도 상속이 가능하다.
3. 부모 클래스의 생성자를 호출하기 위해서 super()를 사용한다.

class ParentClass{

private int x;

public ParentClass(int x){

this.x = x;

}

}

class ChildClass extends ParentClass{

private int x;

public ChildClass(int x){

super(x\*2);

this.x = x;

}

1. **다형성(Polymorphism)**

* 여러 가지 형태를 가질 수 있는 능력을 의미
* 자바에서는 한 타입의 참조변수로 여러 타입의 객체를 참조할 수 있도록 함으로써 다형성을 프로그램적으로 구현.

**[상속에 의한 다형성]**

Animal creature = new Horse();

* 이 경우 변수는 Animal이라는 클래스 타입으로 변수를 선언하여 Animal의 어떤 객체라도 참조 가능하다.
* 따라서 Animal을 상속한 Horse()객체를 참조할 수 있다.

**[인터페이스에 의한 다형성]**

Public interface Speaker{

Public void speak();

}

class A implements Speaker{

public void speak(){…}

}

* 기능과 변수가 선언되지 않은 interface를 선언하고 다른 클래스에서 implements를 통해 기능을 구현할 수 있다.
* 이를 통해 메서드가 전달되는 매개변수의 타입을 제어하는 것을 허용하고, 메서드에게 다양한 타입의 인자들을 받아들일 수 있는 유연성을 제공한다.

**==[오버로드(Overload)와 오버라이드(Overide)]==============================**

오버로드 : 메서드의 이름은 같고 파라미터의 개수나 타입이 다른 함수를 정의하는 것.

class A{

void test(){…}

void test(int a, int b){…}

void test(double d){…}

}

오버라이드 : 상위 클래스의 메서드를 재정의 하는 것, 메서드의 이름과 파라미터의 개수나 타입도 동일해야 함.

class B{

void print(){

System.out.println(“B”);

}

}

class C{

void print(){

System.out.println(“C”);

}

}  
==========================**[출처]**===============================   
http://goo.gl/BfgOx3 객체지향-vs-절차지향-프로그래밍

http://goo.gl/VHDH45  객체 지향 프로그래밍

http://blog.eairship.kr/116 자바 상속

http://arabiannight.tistory.com/314 자바의 다형성이란?

http://hyeonstorage.tistory.com/266 상속, 인터페이스에 의한 다형성

http://javahwan.tistory.com/45 오버로드와 오버라이드의 차이점