**[ 8 ] 클래스의 기초적인 코딩방법**

(cf) 캡슐화(Encapsulation) : 객체가 포함한 속성과 메서드는 객체간의 관계에 있어서 감추거나 권한에 따라 접근이 가능하게 처리하는 것을 말한다. 여기에 사용되는 keyword로 접근제어자(access modifier)가 있다

1. 클래스 제작

package com.ch.ex;

public class ExClass {

private 자료형 인스턴스변수( = 속성 = 필드)명;

public ExClass(){ }

public method(){

. . .

}

}

1. 패키지명
2. 클래스명
3. **데이터(인스턴스 변수=멤버변수, 필드) :** 이 데이터는생성자나 setter를 이용해서 초기화하지 않으면 객체는 null, 숫자는 0, boolean은 false로 초기화되어 들어간다
4. 생성자함수 : 클래스명과 똑같이 리턴타입이 없는 메소드를 생성자라 하며 처음 클래스형 객체를 만들때 호출된다. 모든 클래스는 반드시 하나 이상의 생성자가 있어야 한다. 만약 하나도 없으면 JVM이 디폴트 생성자를 만들어 준다(new 연산자로 호출되는 메서드)
5. **메소드**
6. **Getter & setter**

**package** com.tj.square;

**public** **class** Square {

**private** **int** side;

**public** Square() {System.***out***.println("매개변수가 없는 생성자 호출했음");}

**public** Square(**int** side) {**this**.side = side;

System.***out***.println("매개변수가 있는 생성자 호출했음");

}

**public** **int** area() {**return** side\*side;}

**public** **int** getSide() { **return** side; }

**public** **void** setSide(**int** side) {

**this**.side = side;

}

}**package** com.tj.square;

**public** **class** SquareMain {

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

Square s1 = **new** Square(5);

Square s2 = **new** Square(25);

//s1.setSide(5);

//s2.setSide(10);

System.***out***.println("s1의 넓이는 "+ s1.area());

System.***out***.println("s2의 넓이는 "+ s2.area());

}

}

2. 생성자의 이해 : 생성자는 매개변수 있는 생성자와 매개변수 없는 생성자 등 여러 종류의 생성자를 가질 수 있다. 생성자가 없을 때는 디폴트 생성자가 컴파일러 단계에서 자동 생성한다. 한 개 이상의 생성자가 있으면 디폴트 생성자는 자동 생성되지 않는다.

**public** **class** Man {

**private** **int** age;

**private** **int** height;

**private** **int** weight;

**private** String phoneNum;

// 파라미터값을 갖는 생성자

**public** Man(**int** age, **int** height, **int** weight, String phoneNum) {

**this**.age = age;

**this**.height = height;

**this**.weight = weight;

**this**.phoneNum = phoneNum;

}

**public** Man(**int** height, **int** weight) {

**this**.height = height;

**this**.weight = weight;

System.***out***.println("실행한 파라미터가 있는 생성자");

}

//파라미터가 없는 생성자

**public** Man(){

System.***out***.println("실행한 파라미터가 없는 생성자");

}

//파라미터가 달랑 하나인 생성자

**public** Man(**int** height){

**this**.height = height;

}

// 파라미터가 실수형인 생성자

**public** Man(**double** weight){

**this**.weight = (**int**)weight;

}

**double** calculateBMI(){

**double** result = weight / ((height/100.0)\*(height/100.0));

**return** result;

}

**public** **int** getAge() {**return** age; }

**public** **void** setAge(**int** age) {**this**.age = age;}

**public** **int** getHeight() {**return** height;}

**public** **void** setHeight(**int** height) {**this**.height = height;}

**public** **int** getWeight() {**return** weight;}

**public** **void** setWeight(**int** weight) {**this**.weight = weight;}

**public** String getPhoneNum() {**return** phoneNum;}

**public** **void** setPhoneNum(String phoneNum) {**this**.phoneNum = phoneNum;}

}

**public** **class** Woman {

**public** Woman(){

System.***out***.println("Woman 클래스의 생성자 함수 왔어요");

}

}

**public** **class** MainClass {

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

Woman hee1 = **new** Woman();

------------ --------------

①레퍼런스변수선언 ②인스턴스생성

Woman hee2 = **new** Woman();

System.***out***.println("hee1과 hee2는 같은가"+hee1.equals(hee2));

Man kang = **new** Man(185,75);

Man kim = **new** Man();

Man kang1;

kang1 = kang;

kim.setHeight(185);

kim.setWeight(75);

System.***out***.println("Kim과 Kang이 같은지 " +kim.equals(kang));

System.***out***.println("kang과 kang1이 같은지 " +kang.equals(kang1));

**double** biman = kang.calculateBMI();

**if**(biman>24)

System.***out***.println("Kang님 비만이니 다이어트하세요");

**else**

System.***out***.println("Kang님 비만 아니니 다이어트하지 마세요");

biman = kim.calculateBMI();

**if**(biman>24)

System.***out***.println("Kim님 비만이니 다이어트 하세요");

**else**

System.***out***.println("Kim님 비만아니니 다이어트하지 마세요");

}

}

클래스 객체

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

설계도 제품

객체 ≒ 인스턴스

3. this키워드의 이해

this란 객체자신

~~~~~~~~~~~~~~~~~~~~~~~~~~~~

객체지향 프로그래밍의 특징

(1) 캡슐화와 데이터은닉

(2) 다형성 : 같은 모습이지만 다른 기능 cf.오버로딩

(3) 상속

**(실습예제1)**

은행계좌(Account) 클래스 설계

데이터(속성) : 계좌번호(accountNo:String),

예금주(ownerName:String),

잔액(balance:int)

기능(메소드) : 예금하다(void deposit(int)).

인출하다(int withdraw(int)),

잔액조회(int getBalance())

<Account.java>

/\* 은행계좌(Account) 클래스 설계

데이터(속성) : 계좌번호, 예금주, 잔액

기능(메소드) : 예금하다. 인출하다 \*/

<Main.java>

package oop.acc;

public class Main {

public static void main(String[] args){

Account hong = new Account("111-111","홍길동",2100000000);

Account hong1 = new Account(20000);

Account hong2 = new Account();

System.out.println(hong.getBalance()+"홍");

System.out.println(hong1.getBalance()+"홍1");

System.out.println(hong2.getBalance()+"홍2");

long money = hong.withdraw(1000000);

if(money>300000)

System.out.println("뭐할겨요?");

}

}

**(실습예제2)** 직육면체의 가로, 세로, 높이, 부피나 직사각형의 가로, 세로, 넓이를 속성으로 갖는 클래스를 구현하라. 부피나 넓이를 return하는 메소드 getVolume(부피 혹은 넓이)와 출력하는 메소드 vPrint()v도 구현한다.

데이터 가로 세로 높이, volume(부피 넓이)

메소드 : getVolume

**public** **class** MainClass {

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

BoxOrRect box = **new** BoxOrRect(5,6,10);

BoxOrRect rect = **new** BoxOrRect(10,5);

box.calV();

rect.calV();

System.***out***.println("박스 부피 : "+box.getVolume());

System.***out***.println("rect 넓이 : "+rect.getVolume());

}//main

}

**(실습예제3)** name, age, gender를 데이터로 갖고, 데이터 정보를 출력하는 print() 메소드를 갖는 PersonInfo 클래스를 구현하고 main함수를 이용하여 test 구현하시오(print() 메소드 실행 결과 : 이름=홍길동, 나이=20, 성별=m, main 메소드 안에는 PersonInfo형 객체 인스턴스를 배열로 구현해 봅니다)

**(실습예제4) Member 클래스를 구현하고 테스트하시고**

데이터(속성) : id, pw, name, email, address, birth, gender

메소드 : “아이디 = aaa

이름 = 홍길동

이메일 = hong@company.com

주소 = 서울 강남구

생일 = 2000-01-01

성별 = 남 ”(객체 정보)을 return infoString() 메소드

main()함수 내용 : Member member = new Member(“aaa”, “xxx”, “홍길동”, “hong@company.com”,

“서울 강남구”, “2000-01-01”, ‘M’);

System.out.println(member.infoString() );