**[ 9 ] 접근제한과 static**

**오늘의 워밍업1)** 객체 두 개를 생성하여 다음과 같이 출력하는 프로그램을 구현하시오

/\* 결과 : 이름 = 홍길동 나이 = 20 성별 = m

\* 이름 = 홍길순 나이 = 19 성별 = f \*/

**package** com.tj.janmuri;

**class** PersonInfo {

**private** String name;

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

**private** **char** gender;

**public** PersonInfo() { }

**public** PersonInfo(String name, **int** age, **char** gender) {

**this**.name = name;

**this**.age = age;

**this**.gender = gender;

}

**public** **void** print() {

System.***out***.println("이름 = " + name + "\t나이 = " + age + "\t성별 = "

+ gender);

}

**public** **void** setName(String name) { **this**.name = name; }

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

**public** **void** setGender(**char** gender) { **this**.gender = gender; }

**public** String getName() { **return** name; }

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

**public** **char** getGender() { **return** gender;}

}

**public** **class** PersonInfoTestMain {

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

PersonInfo[] person = { **new** PersonInfo("홍길동", 20, 'm'),

**new** PersonInfo("홍길순", 19, 'f'),

**new** PersonInfo()};

person[2].setName("홍길숙");

person[2].setAge(21);

person[2].setGender('m');

**for** (PersonInfo p : person)

p.print();

}

}

**오늘의 워밍업2)** 클래스를 이용하여 다음 값을 입력하고 다음과 같이 출력

Student = 데이터 : name, kor, eng, mat, tot, avg

생성자 : new Student(“정우성”, 90,90,90) 가능하도록

메소드 : print() ; “정우성 90 90 90 270 90.0” 객체값들 출력

Getter들 & setter들

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

성적표

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

이름 국어 영어 수학 총점 평균

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

정우성 90 90 90 270 90.0

김하늘 90 90 91 271 90.3

황정민 80 80 80 240 80.0

강동원 80 80 81 241 80.3

유아인 70 70 70 210 70.0

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

성적표

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

이름 국어 영어 수학 총점 평균

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

정우성 90 90 90 270 90.0

김하늘 90 90 91 271 90.3

황정민 81 80 80 241 80.3

강동원 80 80 80 240 80.0

유아인 70 70 70 210 70.0

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

총점 411 410 411 1232 410

평균 82.2 82.0 82.2 246.4 82.0

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

**오늘의 워밍업3)** 다음과 같은 결과가 나오도록 클래스를 정의하고 객체를 생성하도록 프로그램을 구현하세요

주차장(ParkingSystem) 클래스 설계 // Parking car1 = new Parking(“11라1111”,12);

car1.out(14)

Parking – 데이터: no, inTime, outTime, fee (주차요금은 시간당 2000)

메소드 : out(int outTime)

주차장 들어올 때 결과 : “11라1111”님 어서오세요

입차시간 : 12시

주차장 나갈 때 결과 : “11라1111”님 안녕히 가세요

\* 입차시간 : 12시

\* 출차시간 : 14시

\* 주차요금 : 4000원

* **멤버변수 선언**

[접근제한자] [static/final] 데이터형 변수명;

① [접근제한자] 접근제한이란 클래스의 데이터나 메소드에 대해서 다른 클래스로부터 접근을 제한하는 것.

public : 접근을 제한하지 않는다. 다른 **모든 클래스에서 사용** 가능

protected : 해당 클래스와 **동일한 패키지**에 있거나 **상속받은 클래스**일 경우에만 사용 가능

default : 접근제어자를 명시하지 않은 경우의 디폴드 접근제어자. **같은 패키지**내의 클래스들은

public 권한을 갖고 접근가능

private : 해당 클래스만이 이 멤버를 사용할 수 있다. **외부에서는 절대로 접근할 수 없다.**

**public**

**protected**

**default**

**private**

**package** com.tj.ex3\_accesstest1;

**public** **class** AccessTest {

**private** **int** privateMember;

**int** defaultMember;

**protected** **int** protectedMember;

**public** **int** publicMember;

**public** AccessTest() {System.***out***.println("AccessTest 클래스 생성자");}

**private** **void** privateMethod() {

System.***out***.println("private 접근 제한자");

}

**void** defaultMethod() {

System.***out***.println("디폴트 접근 제한자");

}

**protected** **void** protectedMethod() {

System.***out***.println("protected 접근 제한자");

}

**public** **void** publicMethod() {

System.***out***.println("public 접근 제한자");

}

}

**package** com.tj.ex3\_accesstest1;

**public** **class** TestMain {

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

AccessTest obj = **new** AccessTest();

//System.out.println(obj.privateMember); private은 외부에서 접근 불가

System.***out***.println(obj.defaultMember); // 같은 패키지 내만 접근허용

System.***out***.println(obj.protectedMember);//같은 패키지나 상속받은 클래스만 접근 허용

System.***out***.println(obj.publicMember); // 모두 접근 허용

//obj.privateMethod(); private은 외부에서 접근 불가

obj.defaultMethod();

obj.protectedMethod();

obj.publicMethod();

}

}

**package** com.tj.ex4\_aceestest2;

**import** com.tj.ex3\_accesstest1.AccessTest;

**public** **class** TestMain {

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

AccessTest obj = **new** AccessTest();

//System.out.println(obj.privateMember); private은 외부에서 접근 불가

//System.out.println(obj.defaultMember); 같은 패키지 내만 접근허용

//System.out.println(obj.protectedMember);같은 패키지나 상속받은 클래스만 접근 허용

System.***out***.println(obj.publicMember); // 모두 접근 허용

//obj.privateMethod(); private은 외부에서 접근 불가

//obj.defaultMethod();

//obj.protectedMethod();

obj.publicMethod();

}

}

②멤버변수의 초기화 시기와 순서

※클래스 변수의 초기화 시점 : 클래스가 처음 로딩될 때 단 한번

※인스턴스변수의 초기화 시점 : 인스턴스(객체)가 생성될 때 마다

**public** **class** Test1 {

**private** **int** i=1; // ② 코드 직접 초기화

**public** Test1() {i=1; } // ①생성자를 통한 초기화

**public** **int** getI() {

**return** i;

}

**public** **void** setI(**int** i) {

**this**.i = i;

}

}

③ static : 클래스변수(=static 변수)

; 객체변수(객체속성)는 객체가 생성될 때마다 각 객체 안의 속성 변수들이 생성되지만,

클래스 변수는 클래스로부터 생성된 객체들의 수와 상관없이 하나만 생성된다.

한 클래스로부터 생성된 모든 객체들은 클래스 변수를 공유

클래스 변수를 이용하여 객체들 사이의 통신에 사용하거나 객체들의 공통속성을 나타낼 수 있다.

객체변수와 달리 클래스 변수(static 변수)는 클래스 이름을 통하여 접근한다.

1. static의 이해

; 객체들끼리 공유하고 싶은 정보가 있을 때 사용. 일명 깍두기.

필요에 의해 클래스를 만들고, 사용할 때는 new 연산자를 통해 객체로 만들어서 사용을 합니다.

이러한 방식으로 모든 것을 처리하다 보면, 문제가 생길 수 있는 다음이 예제를 봅시다.

<잘못된 예>

**public** **class** MomPouch {

**int** money = 2000;

}

**public** **class** Child {

MomPouch momPouch= **new** MomPouch();

**private** String name;

**public** Child(String name) {//Child c1 = new Child("첫째 똘만이");

**this**.name = name;

}

**public** **void** takeMoney(**int** money) {

**if**(momPouch.money >= money) {

momPouch.money -= money;

System.***out***.println(name+"가 "+money

+"원 가져가서 엄마 지값에 "+momPouch.money+"남음");

}**else** {

System.***out***.println(name+"가 돈을 못 받음. 엄마가 돈이 없어");

}//if

}//takeMoney

}

**public** **class** PouchMain {

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

Child firstChild = **new** Child("첫째 똘만이");

Child secondChild = **new** Child("둘째 갑돌이");

Child thirdChild = **new** Child("셋째 똘순이");

firstChild.takeMoney(1000);

secondChild.takeMoney(1000);

thirdChild.takeMoney(1000);

System.***out***.println("첫째 엄마 지갑 돈 : "+firstChild.*momPouch*.*money*);

System.***out***.println("둘째 엄마 지갑 돈 : "+secondChild.*momPouch*.*money*);

System.***out***.println("셋째 엄마 지갑 돈 : "+thirdChild.*momPouch*.*money*);

}

}

예를 들어 집안에 부모, 그리고 세 아이가 있습니다.

엄마 지갑에 100원짜리 2개가 있습니다.

이러한 상황에서 아이들이 엄마한테 100원을 달라고 합니다.

당연히 엄마지갑에는 100원 동전 2개가 있으니, 줄 수 있습니다.

첫째 아이가 먼저 엄마 지갑 객체를 생성하여 100원을 받았습니다.

그리고 둘째 아이가 엄마 지갑 객체를 생성하여 100원을 받았습니다.

마지막으로 셋째 아이가 엄마 지갑 객체를 생성하여 100원을 받았습니다.

여기까지 보면, 뭔가 이상 합니다. 분명 집안에 엄마도 한 명이고, 지갑도 한 개인데, 아이들 모두 엄마 지갑을 생성하여 100원씩을 받았습니다.

200원 밖에 없는데 아이들이 받은 돈은 모두 300원입니다. 버그입니다.

그럼 이런 상황에서 생각할 수 있는 방법은 무엇일까요?

아주 쉽게 생각해 보면 엄마 지갑 객체가 세 아이들 모두에게 공유되면 됩니다.

그러면 첫째 아이가 100원 받고, 둘째 아이가 100원을 받으면, 지갑에는 동전이 없고, 셋째 아이는 100원을 받을 수 없게 됩니다.

그러면 어떻게 엄마 지갑을 모든 아이들이 공유할 수 있을까요?

바로 ‘static’ 키워드를 사용하는 것입니다.

우선 ‘static’ 키워드를 ‘공유(깍두기)’ 라고 생각 합니다.

<올바른 예>

**public** **class** Child {

**static** MomPouch *momPouch*= **new** MomPouch();

**private** String name;

**public** Child(String name) {//Child c1 = new Child("첫째 똘만이");

**this**.name = name;

}

**public** **void** takeMoney(**int** money) {

**if**(*momPouch*.*money* >= money) {

*momPouch*.*money* -= money;

System.***out***.println(name+"가 "+money+"원 가져가서 엄마 지값에 "+*momPouch*.*money*+"남음");

}**else** {

System.***out***.println(name+"가 돈을 못 받음. 엄마가 돈이 없어");

}//if

}//takeMoney

}

**public** **class** PouchMain {

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

Child firstChild = **new** Child("첫째 똘만이");

Child secondChild = **new** Child("둘째 갑돌이");

Child thirdChild = **new** Child("셋째 똘순이");

firstChild.takeMoney(1000);

secondChild.takeMoney(1000);

thirdChild.takeMoney(1000);

System.***out***.println("첫째 엄마 지갑 돈 : "+firstChild.*momPouch*.*money*);

System.***out***.println("둘째 엄마 지갑 돈 : "+secondChild.*momPouch*.*money*);

System.***out***.println("셋째 엄마 지갑 돈 : "+thirdChild.*momPouch*.*money*);

System.***out***.println("얘들 엄마 지갑 돈 : "+Child.*momPouch*.*money*);

}

}

* 또 다른 예제를 한번 살펴보자.

**public** **class** Product {

**public** **static** **int** *count* = 100; // 생성된 인스턴세의 수를 저장하기 위한 static 변수

**private** **int** serialNo; // 객체 고유의 번호

**public** Product() {

serialNo = ++ *count*;

}

**public** **int** getSerialNo() {

**return** serialNo;

}

**public** **void** infoString() {

System.***out***.println("serialNo : "+serialNo+"\t현재 count 공유번수 : "+*count*);

}

}

**public** **class** ProductMain {

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

Product p1 = **new** Product();

p1.infoString();

Product p2 = **new** Product();

p2.infoString();

Product p3 = **new** Product();

p3.infoString();

System.***out***.println(p1.*count* +"\t"+p2.*count* +"\t"+p3.*count*+"\t"+Product.*count*);

}

}

1. static과 객체와의 관계

※ static 변수를 클래스 변수(정적변수, 공용변수)라 부릅니다.

객체는 클래스에서 생성(복사)되어 진다고 배웠습니다.

클래스 하나에서 무한대로 객체를 생성할 수 있고 그러한 객체는 모양은 동일할지 모르지만, 전혀 다른 객체입니다. 하지만 static이 붙은 변수는 객체 변수가 아닌 클래스 변수로써 객체가 생성되기 전에 이미 존재합니다. (객체를 생성하지 않고 클래스 내 변수를 사용) 객체 생성과 상관없이 사용할 수 있는 필드와 메서드에 활용할 수 있습니다.

객체들이 모여있음

(new를 이용해서 생성)

클래스(설계도)들이

모여 있음

데이터 영역 heap 영역

(가비지 콜렉션이 관리하지 않는 영역) (가비지 컬렉션 관리영역)

자바에서는 개발자가 메모리 관리를 하지 않습니다. 단 사용한 메모리를 회수하고 사용할 수 있게 하는 것은 heap 영역이고 JVM이 가비지 콜렉션한다.

⑶ static의 장점과 단점

객체 생성을 하지 않고도 사용할 수 있는다는 것은 분명 장점이지만, 한편으로 생각해 보면 가비지콜렉터의 관리 밖에 있기 때문에 항상 메모리에 상주해 있습니다. 즉 프로젝트가 커지고, 시스템이 오랜 시간 동안 돌아가게 되면 시스템 운영속도가 점차 느려지다가 급기야 큰 재앙이 올 수도 있습니다. 따라서 static의 사용은 신중해야 됩니다.

④ [static/final]

final : 종단변수

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

**final** **double** PI = 3.141592;

**double** r = 3;

**double** area = PI \* r \* r;

**double** round = 2 \* PI \* r;

System.***out***.println("반지름 "+r+"인 원의 넓이는 "+area);

System.***out***.println("반지름 "+r+"인 원의 둘레는 "+round);

r = 5; area = PI \* r \* r; round = 2 \* PI \* r;

//PI = 3.14; final변수는 상수. 값을 변경하지 못한다

}

static 변수로 사용되는 대표적인 예를 아마도 상수(변하지 않는 수) 일 것입니다. 만약 프로그래밍을 하다가 원주율이 빈번이 사용되어야 한다면 이러한 값은 상수(final 변수)로 정의해 놓고 사용하는 편이 좋습니다.

원주율은 3.141592653589793238462643383279… 끝이 없습니다.

이러한 경우 원주율을 3.141592653589793238462643383279까지로 정의해 놓고 사용하자고 팀원들 사이 약속을 합니다.

**package** com.ch.cons;

**public** **class** PiClass {

**public** **static** **final** **double** ***PI*** = 3.141592653589793238462643383279;

}

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

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

System.***out***.println("원주율:"+PiClass.***PI***);

//PiClass.PI = 3.14; // final 클래스 변수(상수)이므로 변경 불가

}

}

**앞선 소스를 다음과 같이 변환해 구현해보자**

**Student.java(static이용한 번호 매김) 와**

**StudentTestMain.java(객체배열을 이용한 확장 for문 이용)**

**실행 콘솔화면 캡처 파일 1개,**

**전체 이클립스 캡처 파일 1개 (.jpg , .png)**

**총 4개 파일을 압축하여 첨부해 주세요(**[**yisy0703@naver.com**](mailto:yisy0703@naver.com)**)**

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

성 적 표

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

번호 이름 국어 영어 수학 총점 평균

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

**1**  정우성 90 80 95 265 88

**2**  김하늘 100 80 95 275 91

**3**  황정민 95 80 90 265 88

**4** 강동원 95 90 99 284 94

**5** 유아인 90 90 90 270 90

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

또는

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

성 적 표

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

번호 이름 국어 영어 수학 총점 평균

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

**1**  정우성 90 80 95 265 88

**2**  김하늘 100 80 95 275 91

**3**  황정민 95 80 90 265 88

**4** 강동원 95 90 99 284 94

**5** 유아인 90 90 90 270 90

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

총점 xx xx xx xxx xx

평균 xx.x xx.x xx.x xxx.x xx.x

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■