**[ 8 ] 접근제한과 static, 패키지**

**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

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

마동석 99 99 99 210 70.0

**package** com.tj.ex1\_student;

**public** **class** Student {

**private** String name;

**private** **int** kor;

**private** **int** eng;

**private** **int** mat;

**private** **int** tot;

**private** **double** avg;

**public** Student() { }

**public** Student(String name, **int** kor, **int** eng, **int** mat) {

**this**.name = name;

**this**.kor = kor;

**this**.eng = eng;

**this**.mat = mat;

**this**.tot = kor+eng+mat;

**this**.avg = tot/3.0;

}

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

System.***out***.printf("\t%s \t %d \t %d \t %d \t %d \t %.1f\n",

name,kor, eng, mat, tot, avg);

}

//getter와 setter 추가

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

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

**public** **int** getKor() {**return** kor;}

**public** **void** setKor(**int** kor) {**this**.kor = kor;}

**public** **int** getEng() {**return** eng;}

**public** **void** setEng(**int** eng) {**this**.eng = eng;}

**public** **int** getMat() {**return** mat;}

**public** **void** setMat(**int** mat) {**this**.mat = mat;}

**public** **int** getTot() {**return** tot;}

**public** **void** setTot(**int** tot) {**this**.tot = tot;}

**public** **double** getAvg() {**return** avg;}

**public** **void** setAvg(**double** avg) {**this**.avg = avg;}

}

**package** com.tj.ex1\_student;

**public** **class** StudentMain {

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

Student s1 = **new** Student("정우성", 90, 90, 90);

Student s2 = **new** Student("김하늘", 90, 90, 91);

Student s3 = **new** Student("황정민", 80, 80, 80);

Student s4 = **new** Student("강동원", 80, 80, 81);

Student s5 = **new** Student("마동석", 70, 70, 70);

String[] title = {"이름","국어","영어","수학","총점","평균"};

System.***out***.println("\t\t\t성적표");

*line*('-',50);

**for**(String t : title) {

System.***out***.print('\t'+t);

}

System.***out***.println();

*line*('-', 50);

s1.print();

s2.print();

s3.print();

s4.print();

s5.print();

}

**private** **static** **void** line(**char** ch, **int** cnt) {

System.***out***.print('\t');

**for**(**int** i=0 ; i<cnt ; i++) {

System.***out***.print(ch);

}

System.***out***.println();

}

}

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

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

마동석 70 70 70 210 70.0

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

총점 411 410 411 1232 410

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

1. **멤버변수 선언**

[접근제한자] [static/final] 데이터형 변수명;

[접근제한자] [static/final] 리턴타입 메소드명(매개변수){ 명령어들; }

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 패키지에 있거나 상속받은 클래스일 경우에만 사용 가능

default : 접근제어자를 명시하지 않은 경우의 디폴드 접근제어자. 같은 패키지내의 클래스들은

public 권한을 갖고 접근가능

private : 해당 클래스만이 이 멤버를 사용할 수 있다. 외부객체에서는 접근 불가

**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();

}

}

1. 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** Mom {

**int** money = 2000;

// public MomPouch() {

// money = 2000;

// }

}

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

Mom mom;

**private** String name;

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

**this**.name = name;

mom= **new** Mom();

}

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

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

mom.money -= money;

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

+"원 가져가서 엄마 지값에 "+mom.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.*mom*.*money*);

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

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

}

}

위의 예제에서는 엄마 지갑(momPouch)에 2000원이 있습니다.

이러한 상황에서 아이들(Child형 객체)이 엄마한테 1000원을 달라고 합니다. 2명에게는 줄 수 있습니다.

그러나, 첫째 아이, 둘째 아이, 셋째 아이 모두 1000원을 받았습니다. 버그입니다.

그럼 이런 상황에서 생각할 수 있는 방법은 무엇일까요?

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

그러면 어떻게 엄마 지갑을 모든 아이들이 공유하는 방법이 ‘static’입니다.

<올바른 예>

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

**static** Mom *mom*= **new** Mom();

**private** String name;

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

**this**.name = name;

}

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

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

*mom*.*money* -= money;

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

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

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

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

}

}

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.

1. 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.

1. [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 3.141592653589793238462643383279 어디까지 정의해 놓을지 팀원들과 약속을 합니다.

**package** constant;

**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

}

}

1. 패키지의 이해

패키지란? 프로그래밍에서 여러 클래스를 관리하기 위해 기능적으로 영향을 미칠 수 있는 클래스끼리 묶어 놓고, 접근 범위 안에 효과적으로 호출하기 위해서 사용하는 개념이다.

윈도우에는 윈도우탐색기(window+E)가 있다. 윈도우 탐색기는 컴퓨터에 보관되어 있는 파일을 관리하기 위해 폴더를 만들고, 그러한 폴더를 효율적으로 다루기 위해 존재한다. 즉 폴더 구조를 한눈에 볼 수 있도록 만들어진 프로그램이다.

JAVA에도 많은 ~.java파일들이 존재 합니다. 많은 정도가 예상하기도 힘들 정도로 많이 존재합니다. 또한 우리가 프로그래밍을 하면서 계속적으로 .java파일을 만들 것입니다. 이러한 많은 파일을 효율적으로 관리하기 위해 JAVA에서는 ‘패키지’라는 구조를 사용 합니다.

우선 JAVA(jdk)에 존재하는 패키지를 살펴 보도록 하겠습니다. 이미 만들어 놓은 기능들입니다.

1. 아래 주소접속하여 자바(jdk)에서 제공하는 클래스(3,000개정도)들이 어떤 것들이 있는지 봅시다.
2. Jdk8: <http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html> 에 접속하여 다운받거나 온라인으로 <https://docs.oracle.com/javase/8/docs/api/index.html> 에 접속하여 API들을 볼 수 있다.
3. Jdk17 : <https://www.oracle.com/java/technologies/javase-jdk17-doc-downloads.html> 에서 접속하여 다운받거나, 온라인으로 <https://docs.oracle.com/en/java/javase/17/docs/api/index.html> 에 접속하여 API들을 볼 수 있다.
4. JAVA API(Application Programming Interface) : 자바에서 기본적으로 제공되는 클래스의 설명들을 볼 수 있다. 자바 사전이라고 생각하고, 자주 보는 습관을 갖자.
5. 모든 클래스들을 패키지로 관리되어 있는 것을 볼 수 있다. 해당 패키지에는 해당클래스를 볼 수 있습니다.
6. 패키지 : **관련있는 클래스들을 패키지로 그룹지어 관리하는 것(클래스를 모아 놓은 폴더)**

ex. Scanner 클래스는 java/util폴더에 있는 Scanner클래스

(java.util.Scanner)

ex. Math클래스는 java/lang폴더에 있는 Math클래스 : java.lang 패키지의 클래스는 import 생략.

(java.lang.Math)

1. 우리가 만든 패키지

JAVA(jdk)에 패키지가 있듯이 우리도 클래스를 만들 때 패키지로 관리하면 효율적인 작업을 할 수 있습니다.

이제 직접 패키지를 만들어 보도록 하겠습니다.

☞ 실제 폴더 구조를 보도록 합시다

1. 정답은 없지만, 효율적으로 해야 하는 패키지의 일반적인 설계방식
   1. 패키지 이름만 봐도 어떤 클래스들이 모여있는지 알 수 있도록 만들고 싶은데 그게 쉽지 않은 것이죠. 패키지 이름은 유일한 이름이여야 합니다. 이유는 우리회사에서 개발은 소스(.java파일)를 다른 곳에 이식하려 할 때 이름이 중복되면 되기 때문입니다. 그래서 주로 패키지 이름은 유니크한 도메인 주소를 역방향으로 하여 만듭니다. 예를 들어 회사 도메인이 www.lect.com 이라면, ‘com.lect.\*’라고 하고 뒤에 기능별 또는 개발팀에서 정한 규칙에 의해 패키지명을 만듭니다.

② 물론 패키지명은 개발 중 언제든지 수정이 가능, 패키지 간의 클래스 이동도 가능. 따라서 패키지명을 만들 때 너무 스트레스를 받지 말고, 쉽게 쉽게 생각하면 될 거 같습니다.

③ 명명규칙은 변수와 같이 숫자로 시작할 수 없고, \_나 $를 제외한 특수문자를 사용할 수 없다. 모두 소문자로 작성하는 것이 일반적이다.

1. import의 이해 : 다른 패키지의 클래스를 가져오자~

패키지를 만들고 클래스를 만든 후 서로 다른 패키지의 클래스를 사용하려고 할 때는 반드시 import 키워드를 사용해서 해당 클래스를 임포트 해야 합니다. 하지만 우리가 사용하는 이클립스는 자동으로 임포트되도록 지원합니다.

(cf) 동일한 패키지의 클래스는 임포트가 필요없음.

(cf) import com.lect.ex.\*; //com.lect.ex.\* 패키지 안에 있는 모든 것. 패키지내에 너무 많은 클래스가 있을 경우엔 메모리 환경에 안좋은 영향을 미치겠지만 요즘은 메모리 환경이 좋아 이런 방식을 많이 사용합니다.