|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[유헬스2\_4주\_실습과제]** | **학번:** | **20175105** | **이름:** | **곽영주** |

* 강의 내용

abstract class Shape { //추상 클래스

abstract void draw(); //추상 메소드

}

class Rectangle extends Shape {

private int width;

public void setWidth(int w) { width = w; }

//메소드 재정의

public void draw() { System.out.println("Rectangle Draw"); }

}

class Triangle extends Shape {

public void draw() { System.out.println("Triangle Draw"); }

}

class Circle extends Shape {

public void draw() { System.out.println("Circle Draw"); }

}

1. 메소드 재정의

* 메소드 재정의(method overriding): 서브 클래스가 필요에 따라 상속된 메소드를 다시 정의하는 것
* 메소드 재정의 조건
* 메소드의 이름, 반환형, 매개 변수의 개수와 데이터 타입이 일치하여야 한다
* 오버라이딩된 메소드의 접근 지정자는 슈퍼 클래스의 메소드의 접근 지정자 보다 좁아질 수 없다.

public > protected > private 순으로 지정 범위가 좁아진다.

* private, 또는 final 메소드는 재정의 불가

1. 추상 클래스

* abstract 키워드가 붙은 클래스
* 추상 클래스는 객체를 생성하지 못한다(인스턴스화 금지)
* 추상 클래스는 다른 클래스를 파생하기 위한 기초 클래스로만 사용
* 추상클래스는 일반 메소드도 포함할 수 있으며, 추상 메소드가 포함되면 반드시 abstract 키워드를 사용하여야 한다.
* 추상 메소드로 정의하면 파생클래스에서 필요한 동작을 정의

1. 추상 메소드

* 추상 메소드(abstract method) : 메소드 본체가 없는 메소드
* 서브 클래스에서 반드시 메소드를 구현하도록 하기 위해 사용

1. 다형성

* 객체들의 타입이 다르면 똑같은 메시지가 전달되더라도 서로 다른 동작을 하는 것

1. 형변환

* **부모 클래스 참조 변수를 통하여 자식 클래스의 필드와 메소드를 사용하고자 할 때는 하향 형변환을 하여야 한다**

Shape s = new Rectangle();

**//객체 타입 알아내기 : 참조변수 s가 Rectangle 타입의 객체를 참조하면 true**

if(s **instanceof** Rectangle)

((Rectangle) s).setWidth(100);

* 내용 점검 문제

1. 다음에서 서술 내용이 맞으면 0, 틀리면 x 하시오.
2. 키워드 protected 사용한 클래스는 다른 모든 클래스에서 사용할 수 있다. ( x )
3. 키워드 this는 객체 자신을 의미하는 키워드이다. ( o )
4. 사용자가 직접 구현하는 생성자에서 첫 줄이 상위 생성자의 호출인 super() 또는 super(인자)가 아니면 자동으로 기본 생성자 super()를 호출한다. ( o )
5. 생성자 호출에 의해 객체가 생성될 때 항상 상위 객체를 위한 필드와 메소드가 먼저 생성된 후 하위 객체가 생성된다. ( o )
6. 구현 없이 자동으로 사용되는 기본 생성자의 첫 줄은 무조건 super()를 호출한다.( o )
7. 상위 클래스에 이미 선언된 이름이 동일한 필드가 필요하더라도 하위 클래스에서 다시 선언하여 사용할 수 없다. ( x )
8. 클래스에서 인자가 있는 생성자가 적어도 하나 구현되었다면 더 이상 기본 생성자는 자동으로 사용할 수 없다. ( o )
9. 메소드 오버로딩의 목적은 상위 클래스에서 이미 정의한 메소드를 다시 수정하지 않고 하위 클래스에서 좀 더 보완 수정하거나 완전히 새로운 것으로 대체하기 위한 방법이다. ( o )
10. 메소드 오버라이딩에서 메소드의 반환 값과 메소드 이름, 매개변수는 반드시 같아야 한다. ( x )
11. 추상 클래스는 개념적으로 클래스 간의 계층구조에서 상위에 존재하여 하위 클래스를 대표하는 클래스이다. ( o )
12. 추상 클래스도 직접 객체화(instantiation)될 수 있다. ( x )
13. 추상 클래스는 생성자 구현이 필요 없다. ( x )
14. 추상 메소드 정의 시 반환형 앞에 키워드 abstract를 기술한다. ( o )
15. 추상 클래스는 일반 메소드를 가질 수 없다. ( x )
16. 다음 문장에서 비어있는 부분을 채우시오.
17. ( 상속 )은 하위 클래스는 상위 클래스의 특징인 필드와 메소드를 그대로 물려받을 수 있는 특성이다.
18. 필드와 메소드인 멤버의 접근을 제한하는 접근 지정자로는 public, ( protected ), default, ( private )가 있다.
19. 접근 지정자 ( protected )는 같은 패키지와 다른 패키지라도 하위 클래스에서는 모두 사용될 수 있도록 하는 지정자이다.
20. ( super(); )는 상위 클래스의 기본 생성자를 호출하는 문장이다.
21. 상위 클래스의 동일한 메소드를 하위 클래스에서 다시 정의하는 것을 메소드 ( 오버라이딩 )이라 한다.
22. 메소드 지정자 ( final )은 더 이상 하위 클래스에서 메소드 오버라이딩을 허용하지 않도록 지정하는 키워드이다.
23. 추상 클래스는 클래스 정의 시 키워드 class 앞에 ( abstract ) 키워드를 기술하여 구현한다.
24. 다음과 같은 클래스 정의에 대하여 답하라.

class Student {

private int number;

protected String name;

}

public class GraduateStudent extends Student {

public String lab;

}

(1) 위의 코드에서 부모 클래스는 ( Student )이고 자식 클래스는 ( GraduateStudent )이다.

(2) 각 필드의 접근자와 설정자를 작성하라.

public void setNumber(int number) {

this.number = number;

}

public int getNumber() {

return number;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setLab(String lab) {

this.lab = lab;

}

public String getLab() {

return lab;

}

(3) 생성자를 추가하여 보라. 서브 클래스의 생성자에서 수퍼 클래스의 생성자를 명시적으로 호출하게 하라.

public Student(int number, String name) {

this.number = number;

this.name = name;

}

public GraduateStudent(int number, String name, String lab) {

super(number, name);

this.lab = lab;

}

(4) GraduateStudent s = **new** GraduateStudent();와 같이 객체를 생성하였다고 하자. 다음 중 필드를 잘못 접근한 것은?

① s.number = 10;

② s.lab = "Image Processing Lab";

③ s.name = "홍길동“;

1,2,3번 모두 접근 불가

이유 : main메소드는 상속받은 메소드가 아니기 때문에 생성자로 접근하거나 접근자로 접근해야 한다.

1. 다음 프로그램의 실행결과를 제시하시오.

**class** Animal{

String name;

**public** Animal() {

name = "UNKNOWN";

System.*out*.println("동물입니다:" + name);

}

**public** Animal(String name) {

**this**.name = name;

System.*out*.println("동물입니다:" + name);

}

}

**class** Lion **extends** Animal{

**public** Lion() {

System.*out*.println("사자입니다.");

}

**public** Lion(String name) {

**super**(name);

System.*out*.println("사자입니다.");

}

}

**public** **class** Test {

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

Lion lion=**new** Lion("Brave");

Lion lion2=**new** Lion();

}

}

|  |
| --- |
| **[실행 결과]** |

* 상속 & 다형성 & 추상 클래스 사용 예

|  |
| --- |
| Add(+) |
| 피연산자(int) a, b |
| calculate() : +연산 실행 후 결과 반환 |

|  |
| --- |
| Mod(%) |
| 피연산자(int) a, b |
| calculate() : %연산 실행 후 결과 반환 |

**abstract** **class** Cal{ //각 클래스에서 공통되는 부분을 수퍼클래스로 정의, 추상클래스

**protected** **int** a, b; //연산을 위한 피연산자

**public** Cal(**int** a, **int** b){

**this**.a=a;

**this**.b=b;

System.***out***.println("수퍼클래스 생성자");

}

//추상 메소드 메소드 - 서브 클래스에서 반드시 재정의하여 사용

**abstract** **int** calculate();

**public** String toString(){

**return** "a = " + a + ", b =" + b ;

}

}

**class** Add **extends** Cal{ //Cal 클래스를 상속받는 서브클래스 Add

**public** Add(**int** a, **int** b){

**super**(a, b); //수퍼 클래스 생성자 호출 - 필드 초기화

System.***out***.println("Add 생성자");

}

@Override

**public** **int** calculate(){ //수퍼클래스의 calculate() 메소드 재정의

**return** a+b;

}

//수퍼 클래스 toString()메소드를 호출하여 필드 내용 출력

@Override

**public** String toString() {

**return** "[+] " +**super**.toString() + " => " + calculate();

}

}

**class** Mod **extends** Cal { //Cal 클래스를 상속받는 서브클래스 Mod

**public** Mod(**int** a, **int** b) {

**super**(a, b); //수퍼클래스 생성자 호출 – 필드 초기화

System.***out***.println("Mod 생성자");

}

@Override

**int** calculate() { //수퍼클래스의 calculate() 재정의

**return** a%b;

}

@Override

**public** String toString() {

**return** "[%] " +**super**.toString() + " => " + calculate();

}

}

**import** java.util.\*;

**public** **class** InherTest {

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

Scanner in=**new** Scanner(System.***in***);

Cal[] obj = **new** Cal[3]; //객체 배열 선언 & 생성

**for**(**int** i=0; i<obj.length;i++){

System.***out***.print( "\n1: Add, 2: Mod -> 메뉴입력( " + (i+1) + " 번째) : ");

**int** menu=in.nextInt();

**switch**(menu){

**case** 1:

System.***out***.println("\n=== Add ===" );

System.***out***.print("오퍼랜드 a, b를 입력하세요 => ");

obj[i] = **new** Add(in.nextInt(), in.nextInt()); //객체 배열 초기화

**break**;

**case** 2:

System.***out***.println("\n=== Mod ===" );

System.***out***.print("오퍼랜드 a, b를 입력하세요 => ");

obj[i] = **new** Mod(in.nextInt(), in.nextInt()); //객체 배열 초기화

**break**;

}

}

*disPlay*(obj); //객체 배열을 매개변수로 전달

}

**public** **static** **void** disPlay(Cal[] obj){

**for**(Cal temp : obj)

System.***out***.println(temp.toString()); //다형성

}

}

|  |
| --- |
| **[실행 결과]** |

* 프로그램 과제 1 : 도형의 면적을 구하여 출력하는 프로그램입니다. 빈 공란을 채워 넣은 후 결과를 확인하세요.

abstract class Polygon {

protected double data1; //가로 또는 밑변

protected double data2; //세로 또는 높이

//매개변수로 필드 초기화 하는 생성자 - 본인 작성

abstract double areaPolygon(); //넓이를 계산하는 추상 메소드

}

class Rectangle extends Polygon{

//매개변수로 필드 초기화 하는 생성자, 수퍼클래스 필드 초기화도 함께 - 본인 작성

//사각형 넓이를 계산하여 반환하는 areaPolygon() 재정의 - 본인 작성

}

class Triangle extends Polygon{

//매개변수로 필드 초기화 하는 생성자, 수퍼클래스 필드 초기화도 함께 - 본인 작성

//삼각형 넓이를 계산하여 반환하는 areaPolygon() 재정의 - 본인 작성

}

class Ladder extends Polygon{

private double data3; //아랫변

//매개변수로 필드 초기화 하는 생성자, 수퍼클래스 필드 초기화도 함께 - 본인 작성

//사다리꼴 넓이를 계산하여 반환하는 areaPolygon() 재정의 - 본인 작성

}

import java.util.\*;

public class PolygonArea {

public static void main(String[] args) {

Scanner key=new Scanner(System.in);

int menu;

Polygon obj;

System.out.println("1:직사각형, 2:삼각형, 3:사다리꼴-> 번호를 입력 하세요");

menu=key.nextInt();

//1번이 선택되면 Rectangle 객체를 생성하고 직사각형 넓이를 계산하여 출력한다.

//2번은 삼각형 넓이, 3번은 사다리꼴 넓이

//객체 참조 변수는 obj를 사용할 것 – 다형성

}

}

|  |
| --- |
| **[프로그램 소스]**  **abstract** **class** Polygon {  **protected** **double** data1; //가로  **protected** **double** data2; //세로    **public** Polygon(**double** data1, **double** data2){  **this**.data1 = data1;  **this**.data2 = data2;  }    **abstract** **double** areaPolygon();  }  **public** **class** Rectangle **extends** Polygon {    **public** Rectangle(**double** data1, **double** data2){  **super**(data1, data2);  }    **public** **double** areaPolygon(){  **return** data1 \* data2;  }  }  **public** **class** Triangle **extends** Polygon {    **public** Triangle(**double** data1, **double** data2){  **super**(data1, data2);  }    **public** **double** areaPolygon(){  **return** (data1 \* data2) \* 0.5;  }  }  **public** **class** Ladder **extends** Polygon {  **private** **double** data3; //아랫변    **public** Ladder(**double** data1, **double** data2, **double** data3){  **super**(data1, data2);  **this**.data3 = data3;  }    **public** **double** areaPolygon(){  **return** 0.5 \* (data1 + data3) \* data2;  }  }  **import** java.util.\*;  **public** **class** PolygonArea {  **public** **static** **void** main(String[] args){  Scanner key = **new** Scanner(System.***in***);  **int** menu;    Polygon obj = **null**;  System.***out***.println("1:직사각형, 2:삼각형, 3:사다리꼴-> 번호를 입력 하세요");  menu = key.nextInt();    **switch**(menu){    **case** 1 :  obj = **new** Rectangle(1,2);  **break**;    **case** 2 :  obj = **new** Triangle(2,3);  **break**;  **case** 3 :  obj = **new** Ladder(3,2,4);  **break**;  }    System.***out***.println(obj.areaPolygon());  }  } |
| **[실행 결과]** |

* 프로그램 과제 2 : 사각형과 삼각형, 사다리꼴 도형 중 사용자가 선택한 도형을 출력하는 프로그램을 작성하시오. 단 도형을 그리는데 사용하는 문자는 도형별로 다르게 사용한다.
* ShapeDraw 추상 클래스

- 추상 메소드 : draw(int x, char ch) : 도형을 그리는 행 크기와 출력할 문자를 매개변수 x, ch로 받는다, 반환 값은 없다

* Rectangle 클래스 : ShapeDraw 클래스 상속

- draw() : 메소드 오버라이딩, 사각형 출력

* Triangle 클래스 : ShapeDraw 클래스 상속

- draw() : 메소드 오버라이딩, 삼각형 출력

* Ladder 클래스 : ShapeDraw 클래스 상속

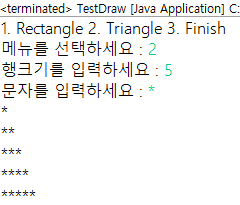
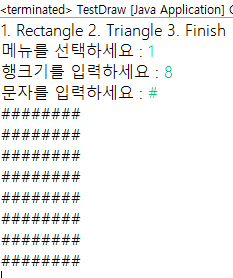
- draw() : 메소드 오버라이딩, 사다리꼴 출력

* TestDraw 클래스

- main() 메소드

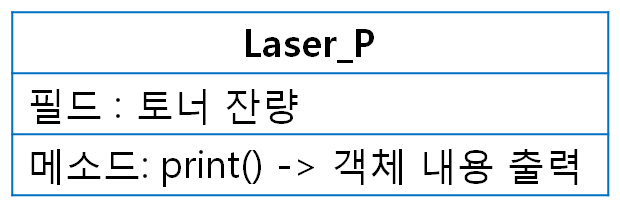
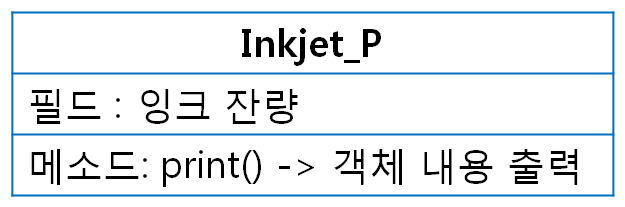
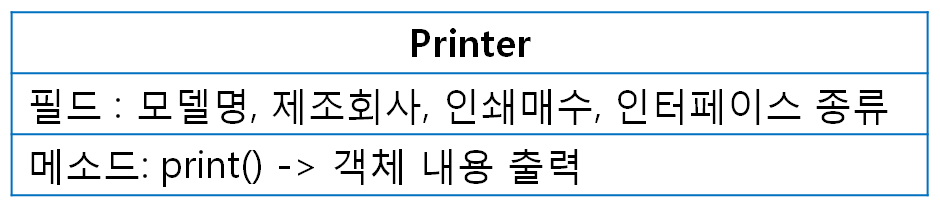
메뉴항목에서 출력하고자 하는 도형과, 행 크기, 문자를 입력 받는다

다형성을 적용할 것.

|  |
| --- |
| **[프로그램 소스]**  **abstract** **class** ShapeDraw {    **abstract** **void** draw(**int** x, **char** ch);  }  **public** **class** Rectangle1 **extends** ShapeDraw {  **public** **void** draw(**int** x, **char** ch){  **for**(**int** i=0; i<x; i++){  **for**(**int** j=0; j<x; j++){  System.***out***.print(ch);  }  System.***out***.println();  }  }  }  **public** **class** Triangle1 **extends** ShapeDraw {    **public** **void** draw(**int** x, **char** ch){  **for**(**int** i=0; i<x; i++){  **for**(**int** j=x-i; j<=x; j++){  System.***out***.print(ch);  }  System.***out***.println();  }  }  }  **public** **class** Ladder1 **extends** ShapeDraw {  **public** **void** draw(**int** x, **char** ch){  **for**(**int** i=0; i<x; i++){  **for**(**int** k=i; k<x; k++){  System.***out***.print(" ");  }  **for**(**int** j=0; j<x+(i\*2); j++){  System.***out***.print(ch);  }  System.***out***.println();  }  }  }  **import** java.util.\*;  **public** **class** TestDraw {  **public** **static** **void** main(String[] args){  Scanner key = **new** Scanner(System.***in***);  **int** x;  **char** ch;  ShapeDraw obj = **null**;    System.***out***.print("1. Rectangle 2. Triangle 3. Ladder 4. Finish ");  **int** menu = key.nextInt();    **switch**(menu){    **case** 1 :  System.***out***.print("행크기를 입력하세요 : ");  x = key.nextInt();  System.***out***.print("문자를 입력하세요 : ");  ch = key.next().charAt(0);  obj = **new** Rectangle1();  obj.draw(x,ch);  **break**;    **case** 2 :  System.***out***.print("행크기를 입력하세요 : ");  x = key.nextInt();  System.***out***.print("문자를 입력하세요 : ");  ch = key.next().charAt(0);  obj = **new** Triangle1();  obj.draw(x,ch);  **break**;    **case** 3 :  System.***out***.print("행크기를 입력하세요 : ");  x = key.nextInt();  System.***out***.print("문자를 입력하세요 : ");  ch = key.next().charAt(0);  obj = **new** Ladder1();  obj.draw(x,ch);  **break**;    **case** 4 :  System.***out***.println("종료");  **break**;  }      }  } |
| **[실행 결과]** |

* 프로그램 과제 3 : 다음 그림과 같은 상속 구조를 갖는 클래스를 구현하고 테스트 하시오

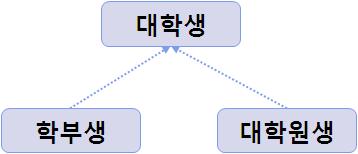


**상속**

**상속**

|  |
| --- |
| **[프로그램 소스]**  **public** **class** Printer {  **private** String model;  **private** String company;  **private** **int** prinCnt;  **private** String interf;    **public** Printer(String model,String company,**int** prinCnt,String interf) {  **this**.model = model;  **this**.company = company;  **this**.prinCnt = prinCnt;  **this**.interf = interf;  }  **public** String getModel() {  **return** model;  }  **public** **void** setModel(String model) {  **this**.model = model;  }  **public** String getCompany() {  **return** company;  }  **public** **void** setCompany(String company) {  **this**.company = company;  }  **public** **int** getPrinCnt() {  **return** prinCnt;  }  **public** **void** setPrinCnt(**int** prinCnt) {  **this**.prinCnt = prinCnt;  }  **public** String getInterf() {  **return** interf;  }  **public** **void** setInterf(String interf) {  **this**.interf = interf;  }    **public** **void** print() {  System.***out***.println("모델명 : " + model);  System.***out***.println("제조회사 : " + company);  System.***out***.println("인쇄매수 : " + prinCnt);  System.***out***.println("인터페이스 종류 : " + interf);  }  }  **public** **class** Inkjet\_P **extends** Printer {  **private** **int** ink;    **public** Inkjet\_P(String model,String company,**int** prinCnt,String interf,**int** ink) {  **super**(model,company,prinCnt,interf);  **this**.ink = ink;  }  **public** **int** getInk() {  **return** ink;  }  **public** **void** setInk(**int** ink) {  **this**.ink = ink;  }    **public** **void** print() {  System.***out***.println();  **super**.print();  System.***out***.println("잉크 잔량 : " + ink + "%");  }  }  **public** **class** Laser\_P **extends** Printer {  **private** **int** toner;    **public** Laser\_P(String model,String company,**int** prinCnt,String interf,**int** toner) {  **super**(model,company,prinCnt,interf);  **this**.toner = toner;  }  **public** **int** getToner() {  **return** toner;  }  **public** **void** setToner(**int** toner) {  **this**.toner = toner;  }    **public** **void** print() {  System.***out***.println();  **super**.print();  System.***out***.println("토너 잔량 : " + toner + "%");  }  }  **public** **class** PrinterTest {  **public** **static** **void** main(String[] args) {  Printer pri = **new** Printer("S-02354","삼성",20,"pr-a56");  Inkjet\_P ink = **new** Inkjet\_P("L-548","LG",15,"pr-b68",60);  Laser\_P las = **new** Laser\_P("N-9874","Nikon",30,"pr-c45",45);    pri.print();  ink.print();  las.print();  }  } |
| **[실행 결과]** |

* 프로그램 과제 4 : 다음 그림에 해당하는 클래스를 작성하시오



* 모든 학생은 이름, 학번, 학과, 학년, 이수학점 수를 갖는다
* 학부생은 동아리명을 갖는다
* 대학원생은 조교유형과 장학금 비율을 갖는다. 조교유형은 교육조교와 연구조교가 있고, 장학금 비율은 0과1사이의 값
* 각 클래스는 생성자, 접근자, 변경자 메소드를 갖는다
* 각 클래스는 객체 내용을 반환하는 toString() 메소드를 갖는다
* 자식 클래스 객체 생성시 부모 클래스 필드 초기화에 필요한 값도 생성자 매개변수로 전달
* 이러한 클래스들의 객체를 만들고 정보를 출력하는 테스트 클래스를 작성

|  |
| --- |
| **[프로그램 소스]**  **public** **class** Student {  **private** String name;  **private** String hakNum;  **private** String major;  **private** **int** year;  **private** **int** complete;    **public** Student(String name,String hakNum,String major,**int** year,**int** complete) {  **this**.name = name;  **this**.hakNum = hakNum;  **this**.major = major;  **this**.year = year;  **this**.complete = complete;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getHakNum() {  **return** hakNum;  }  **public** **void** setHakNum(String hakNum) {  **this**.hakNum = hakNum;  }  **public** String getMajor() {  **return** major;  }  **public** **void** setMajor(String major) {  **this**.major = major;  }  **public** **int** getYear() {  **return** year;  }  **public** **void** setYear(**int** year) {  **this**.year = year;  }  **public** **int** getComplete() {  **return** complete;  }  **public** **void** setComplete(**int** complete) {  **this**.complete = complete;  }    **public** String toString() {  String str = "이름 : " + name;  str += "\n학번 : " + hakNum;  str += "\n학과 : " + major;  str += "\n학년 : " + year;  str += "\n이수학점 : " + complete;  **return** str;  }  }  **public** **class** Understd **extends** Student {  **private** String dongName;    **public** Understd(String name,String hakNum,String major,**int** year,**int** complete,String dongName) {  **super**(name,hakNum,major,year,complete);  **this**.dongName = dongName;  }  **public** String getDongName() {  **return** dongName;  }  **public** **void** setDongName(String dongName) {  **this**.dongName = dongName;  }    **public** String toString() {  **return** **super**.toString() + "\n동아리명 : " + dongName;  }  }  **public** **class** Graduate **extends** Student {  **private** String jo;  **private** **double** janghak;    **public** Graduate (String name,String hakNum,String major,**int** year,**int** complete,String jo) {  **super**(name,hakNum,major,year,complete);  **if**(jo.equals("교육")) {  **this**.jo = "교육조교";  }  **else** **if**(jo == "연구") {  **this**.jo = "연구조교";  }  **this**.janghak = (Math.*random*());  }  **public** String getJo() {  **return** jo;  }  **public** **void** setJo(String jo) {  **this**.jo = jo;  }  **public** **double** getJanghak() {  **return** janghak;  }  **public** **void** setJanghak(**double** janghak) {  **this**.janghak = janghak;  }    **public** String toString() {  **return** **super**.toString() + "\n조교유형 : " + jo + "\n장학금비율 : " + String.*format*("%.2f",janghak);  }  }  **public** **class** StudentTest {  **public** **static** **void** main(String[] args) {  Student stu = **new** Student("김재석","20175100","컴퓨터공학과",1,18);  Understd und = **new** Understd("조성윤","20175100","컴퓨터공학과",1,17,"일레븐");  Graduate gra = **new** Graduate("곽영주","20175105","컴퓨터공학과",1,17,"교육");    System.***out***.println("수퍼클래스 출력\n" + stu);  System.***out***.println("\n서브클래스(Understd) 출력\n" + und);  System.***out***.println("\n서브클래스(Graduate) 출력\n" + gra);  }  } |
| **[실행 결과]** |

