|  |  |  |
| --- | --- | --- |
| 2019\_2\_java1\_14 | 학번 : 20195124 | 이름 : 김민석 |

* **실행 결과 제출**

|  |
| --- |
| 1. 상속과 다형성 |
| **public** **class** Person {  **private** String name, jumin;  **public** Person(String name, String jumin) {  **this**.name = name;  **this**.jumin = jumin;  }  **void** disPlay() {  System.***out***.println("name=" + name + ", jumin=" + jumin);  }  }  **public** **class** Student **extends** Person{  **private** **int** sc; //장학금 유형  **public** Student(String name, String jumin, **int** sc) {  **super**(name, jumin); //부모 클래스 생성자 호출  **this**.sc=sc;  }  @Override //메소드 재정의  **void** disPlay() {  **super**.disPlay();  System.***out***.println("장학금 유형 : " + sc);  }  }  **public** **class** Graduate **extends** Person{  **private** **int** ass;  **public** Graduate(String name, String jumin, **int** ass) {  **super**(name, jumin);  **this**.ass = ass;  }  @Override //메소드 재정의  **void** disPlay() {  **super**.disPlay();  System.***out***.println("조교 유형 : " + ass);  }  }  **import** java.util.\*;  **public** **class** InheritanceTest {  **public** **static** **void** main(String[] args) {  Person[] p = **new** Person[5];  Scanner in = **new** Scanner(System.***in***);  **for** (**int** i = 0; i < p.length; i++) {  System.***out***.print("1. 학부생 2.대학원생 선택>> ");  **switch** (in.nextInt()) {  **case** 1:  System.***out***.println("이름, 주민번호, 장학금유형을 입력하세요 : ");  p[i] = **new** Student(in.next(), in.next(), in.nextInt());  **break**;  **case** 2:  System.***out***.println("이름, 주민번호, 조교유형을 입력하세요 : ");  p[i] = **new** Graduate(in.next(), in.next(), in.nextInt());  **break**;  **default**:  System.***out***.println("잘못 입력하였습니다");  --i;  }  }  System.***out***.println("======== 객체 내용 출력 =========");  **for**(Person obj : p)  obj.disPlay();  }  } |
| **[실행 결과]** |

* **내용 점검**

1. 다음과 같은 클래스 정의에 대하여 답하라.

class Student {

private int number;

public String name;

}

public class GraduateStudent extends Student {

public String lab;

}

(1) 위의 코드에서 수퍼 클래스는 **Student** 이고 서브 클래스는 **GraduateStudent** 이다.

(2) 각 필드의 접근자와 설정자를 작성하라.

**[풀이]**

class Student {

//필드 ~~

**public void setNumber(int n) { number = n; }**

**public int getNumber() { return number; }**

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

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

}

class GraduateStudent extends Student {

//필드~~~

**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.

**[풀이]**

**//Student 쪽 생성자**

public Student(int number, String name) {

this.number = number;

this.name = name;

}

**//GraduateStudent 쪽 생성자**

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“;

**[풀이]**

**1번이 잘못되었다. 이유는 number 이 private 이기 떄문이다.**

1. 다음 프로그램의 출력은?

class A {

final public int myMethod(int a, int b) { return 0; }

}

class B extends A {

public int myMethod(int a, int b) {return 1; }

}

public class Test {

public static void main(String args[]){

B b = new B();

System.out.println("x = " + b.myMethod(0, 1));

}

}

**[풀이]**

**오버라이드 에러가 뜬다. 부모 클래스 A에서 메소드를 final 로 선언했기 때문에 오버라이드 할 수 없다.**

1. 다음과 같은 클래스 정의에서 질문에 답하라.

**class** ClassA {

**public** **void** methodOne(**int** i) { }

**public** **void** methodTwo(**int** i) { }

**public** **static** **void** methodThree(**int** i) { }

**public** **static** **void** methodFour(**int** i) { }

}

**class** ClassB **extends** ClassA {

**public** **static** **void** methodOne(**int** i) { }

**public** **void** methodTwo(**int** i) { }

**public** **void** methodThree(**int** i) { }

**public** **static** **void** methodFour(**int** i) { }

}

* + - * 1. 어떤 메소드가 수퍼 클래스의 메소드를 재정의하고 있는가?

**ClassB가 수퍼 클래스 A의 메소드를 정의하고 있다.**

(2) 어떤 메소드가 수퍼 클래스의 메소드를 가리고 있는가?.

**methodTwo와 methodFour가 수퍼 클래스의 메소드를 가리고 있다.**

* + - * 1. 컴파일 오류를 지적하라. 실제로 컴파일해 보아도 좋다.

This static method cannot hide the instance method from A

**스태틱 메소드와 일반 메소드를 오버라이드 할 수 없다. 수퍼 클래스의 메소드에 맞춰서 static 을 붙이고 말아야한다.**

**[풀이]**

1. 다음 소스에서 발생하는 오류를 제거한 소스를 제시하시오

**public** **class** Bike{

**private** **int** gear;

**public** **int** speed;

}

**public** **class** MountainBike expands Bike {

**public** **int** seatHeight;

**public** **void** MountainBike(**int** g){

**super**();

gear=g;

}

}

**[풀이]**

**1. 설정자, 접근자 이용**

class Bike {

private int gear;

public int speed;

public int getGear() { return gear; }

public void setGear(int i) { gear = i; }

}

class MountainBike extends Bike {

public int seatHeight;

public void MountainBike(int g) {

this.setGear(g);

}

}

1. **생성자 이용**

class Bike {

private int gear;

public int speed;

public Bike(int g) {

gear = g;

}

}

class MountainBike extends Bike {

public int seatHeight;

public MountainBike(int g) {

super(g);

}

}

1. 다음 프로그램의 출력을 쓰시오.

**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();

}

}

**[풀이]**

**동물입니다:Brave**

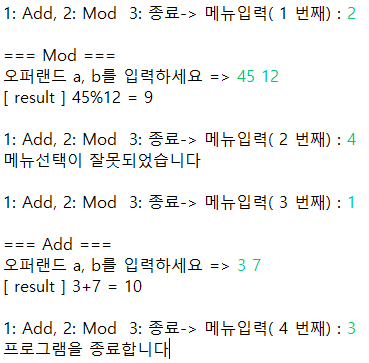
**사자입니다.**

**동물입니다:UNKNOWN**

**사자입니다.**

* **프로그램 과제**

1. 출력 결과를 보고 제시된 프로그램을 완성 하시오.



**class** Cal { //각 클래스에서 공통되는 부분을 수퍼클래스로 정의

**protected** **int** a, b; //연산을 위한 피 연산자

**protected** **char** op; //연산자

//매개변수로 필드값을 초기화 하는 생성자 – 본인작성

**public** String toString(){

**return** "[ result ] " + a + op + b + " = " ;

}

}

**class** Add **extends** Cal{ //Cal 클래스를 상속받는 서브클래스 Add - 본인 작성

//생성자

//toString() 재정의

}

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

//생성자

//toString() 재정의

}

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

//input() 메소드 구현 – 본인작성

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

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

Cal obj = **null**;

**boolean** flag = **true**;

**int** i = 1;

**while** (flag) {

System.***out***.print("\n1: Add, 2: Mod 3: 종료-> 메뉴입력( " + (i++) + " 번째) : ");

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

**switch** (menu) {

**case** 1:

***input*("\n=== Add ===");**

obj = //빈 공란을 채워 넣으시오

**break**;

**case** 2:

***input*("\n=== Mod ===");**

obj = //빈 공란을 채워 넣으시오

**break**;

**case** 3:

flag = **false**;

**break**;

**default**:

System.***out***.println("메뉴선택이 잘못되었습니다");

}

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

}

System.***out***.println("프로그램을 종료합니다");

}

}

|  |
| --- |
| **[프로그램 소스]**  import java.util.Scanner;  class Cal { // 각 클래스에서 공통되는 부분을 수퍼클래스로 정의  protected int a, b; // 연산을 위한 피 연산자  protected char op; // 연산자  //매개변수로 필드값을 초기화 하는 생성자 – 본인작성  public Cal(int a, int b, char op) {  this.a = a;  this.b = b;  this.op = op;  }  *@Override*  public String toString() {  return "[ result ] " + a + op + b + " = ";  }  }  class Add extends Cal { // Cal 클래스를 상속받는 서브클래스 Add - 본인 작성  // 생성자  // toString() 재정의  public Add(int a, int b, char op) { super(a, b, op); }    *@Override*  public String toString() {  String str = super.toString();  str += (a+b);  return str;  }  }  class Mod extends Cal { // Cal 클래스를 상속받는 서브클래스 Mod  //생성자  // toString() 재정의  public Mod(int a, int b, char op) { super(a, b, op); }    *@Override*  public String toString() {  String str = super.toString();  str += (a%b);  return str;  }  }  public class J1\_1203\_HW1 {  // input() 메소드 구현 – 본인작성  public static void input(String p) {  System.***out***.println(p);  System.***out***.print("오퍼랜드 a, b를 입력하세요 ==> ");  }    public static void main(String[] args) {  Scanner in = new Scanner(System.***in***);  Cal obj = null;  boolean flag = true, cnd;  int i = 1;    while (flag) {  System.***out***.print("\n1: Add, 2: Mod 3: 종료-> 메뉴입력( " + (i++) + " 번째) : ");  int menu = in.nextInt();  cnd = true;  switch (menu) {  case 1:  *input*("\n=== Add ===");  obj = new Add(in.nextInt(), in.nextInt(), '+');  break;  case 2:  *input*("\n=== Mod ===");  obj = new Mod(in.nextInt(), in.nextInt(), '%');  break;  case 3:  flag = false;  cnd = false;  break;  default:  System.***out***.println("메뉴선택이 잘못되었습니다");  cnd = false;  break;  }  if(cnd) System.***out***.println(obj);  }  System.***out***.println("프로그램을 종료합니다");  }  } |
| **[실행 결과]** |