|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2019\_1\_java1\_14** | **학번 :** |  | **이름 :** |  |

* **내용 점검**

|  |
| --- |
| 1. 상속  * 기존 클래스를 확장해서 새로운 클래스를 만드는 기술 * 다중 상속 불가 * 필드와 메소드 재사용 * 코드 중복을 줄일 수 있음 * 수퍼(부모, 기본)클래스 * 서브(자식, 파생) 클래스: 상속을 받는 클래스   + 서브 클래스는 수퍼 클래스의 필드와 메소드 사용 가능   + ***서브 클래스에서 추가된 멤버 변수나 메소드는 수퍼 클래스에 의해 사용될 수 없다***  1. 메소드 재정의  * 메소드 재정의(method overriding): 서브 클래스가 필요에 따라 상속된 메소드를 다시 정의하는 것(동적 바인딩) * 메소드 재정의 조건   + 메소드의 이름, 반환형, 매개 변수의 개수와 데이터 타입이 일치하여야 한다 * **private, 또는 final 메소드는 재정의 불가** * 오버라이딩된 메소드의 접근 지정자는 슈퍼 클래스의 메소드의 접근 지정자 보다 좁아질 수 없다.   + public > protected > private 순으로 지정 범위가 좁아진다 * @Override : 재정의되는 메소드임을 의미  1. 상속과 생성자 : 수퍼 클래스의 생성자가 먼저 실행되고 서브 클래스의 생성자 실행됨 2. super 키워드  * 서브클래스에서 수퍼 클래스의 멤버를 접근할 때 사용되는 수퍼클래스 타입의 레퍼런스. * 상속관계에 있는 서브 클래스에서만 사용됨 * 사용 조건   + 오버라이딩 된 수퍼 클래스의 메소드 호출     - 형식 : super.메소드();   + 서브 클래스에서 동일한 이름의 수퍼 클래스 필드 참조     - 형식 :super.필드;   + 수퍼 클래스의 생성자 호출     - 형식 : super();  1. 다형성 : 객체들의 타입이 다르면 똑같은 메시지가 전달되더라도 서로 다른 동작을 하는 것 |
| **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 ( )이고 서브 클래스는 ( )이다.

(2) 각 필드의 접근자와 설정자를 작성하라.

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

(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]**

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]**

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?

(2) 어떤 메소드가 수퍼 클래스의 메소드를 가리고 있는가?.

(3) 컴파일 오류를 지적하라. 실제로 컴파일해 보아도 좋다.

**[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. 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();

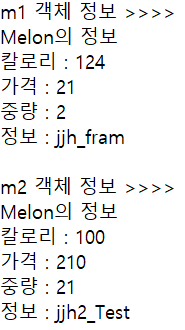
}

}

**[풀이]**

* **프로그램 과제**

1. 일반적인 음식을 나타내는 Food 클래스를 상속받아서 멜론을 나타내는 Melon 클래스를 작성하여 보자. Food 클래스는 칼로리, 가격, 중량 등의 정보를 가진다. Melon 클래스는 추가로 경작 농원 정보를 가진다. 생성자, 접근자, 설정자를 포함하여서 각각의 클래스를 작성한다. 이들 클래스들의 객체를 만들고 각 객체의 모든 정보를 출력하는 테스트 클래스를 작성하라.



|  |
| --- |
| Food |
| -cal : int  -cost : int  -kg : int |
| +getCal() : int  +getCost() : int  +getKg() : int  +setCal() : void  +setCost() : void  +setKg() :void |

|  |
| --- |
| Melon |
| -info : string |
| +getInfo() : string  +setInfo() : void  +toString(): string |

**public** **class** Melon\_Test {

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

Melon m1 = **new** Melon(124, 21, 2, "jjh\_fram");

Melon m2 = **new** Melon(1, 1, 1, "0");

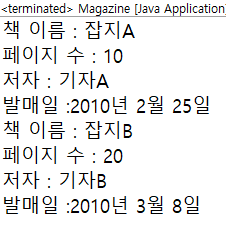
//본인 작성

}

}

|  |
| --- |
| **[프로그램 소스]** |
| **[실행 결과]** |

1. 일반적인 책을 나타내는 Book 클래스를 상속받아서 잡지를 나타내는 Magazine 클래스를 작성하여 보자. Book 클래스는 제목, 페이지수, 저자 등의 정보를 가진다. Magazine 클래스는 추가로 발매일 정보를 가진다. 생성자, 접근자, 설정자를 포함하여서 각각의 클래스를 작성한다. 이들 클래스들의 객체를 만들고 각 객체의 모든 정보를 출력하는 테스트 클래스를 작성하라.

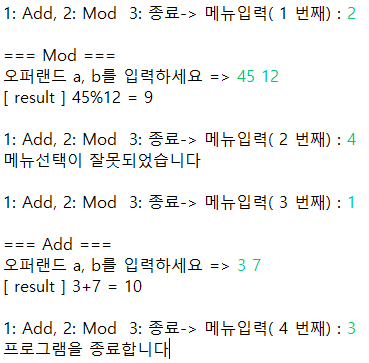


|  |
| --- |
| Book |
| -title  -pages  -writer |
| +getTitle()  +setTitle()  +getPages()  +setPages()  +getWriter()  +setWriter()  +toString |

|  |
| --- |
| Magazine |
| -date |
| +getDate()  +setDate()  +main() |

|  |
| --- |
| **[프로그램 소스]** |
| **[실행 결과]** |

1. 출력 결과를 보고 제시된 프로그램을 완성 하시오.



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

**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

}

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

}

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

**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:

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

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

obj =

**break**;

**case** 2:

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

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

obj =

**break**;

**case** 3:

flag = **false**;

**break**;

**default**:

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

}

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

}

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

}

}

|  |
| --- |
| **[프로그램 소스]** |
| **[실행 결과]** |