|  |  |  |
| --- | --- | --- |
| **2019\_2\_java2\_05** | **학번 : 20135151** | **이름 : 이갑성** |

* **내용 점검**

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. ( O )
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, ( private ), default, ( protected )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** **class** Student {  **private** **int** number;  **protected** String name;    **public** **int** getNumber() {  **return** number;  }  **public** **void** setNumber(**int** number)  {  **this**.number = number;  }    **public** String name() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  }  **public** **class** GraduateStudent **extends** Student {  **public** String lab;    **public** String getLab() {  **return** lab;  }    **public** **void** setLab(String lab) {  **this**.lab = lab;  }  } |

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

|  |
| --- |
| 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. 다음 프로그램의 실행 결과를 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]**

|  |
| --- |
|  |

* 실행결과 확인 1 : 열거형 사용하기

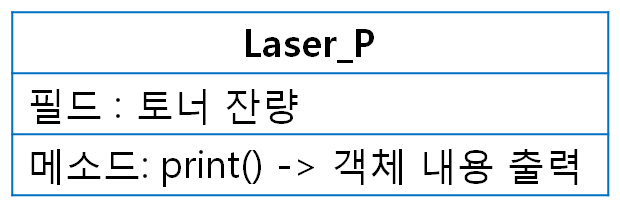
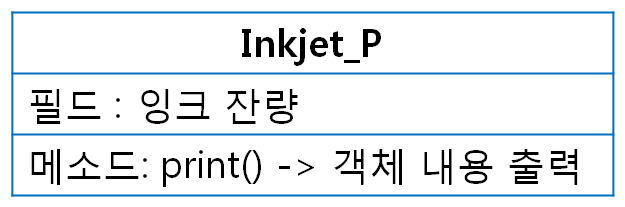
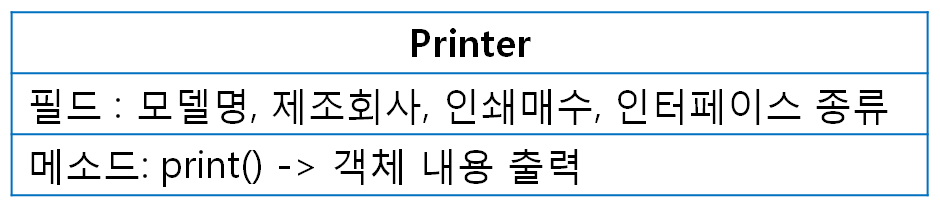
|  |
| --- |
| **[프로그램 소스]**  **enum Week { *일요일, 월요일,화요일, 수요일, 목요일, 금요일, 토요일}***  **public class Enum\_Exam {**  **public static void main(String[] args) {**  Week[] days=**Week**.**values();**  *//열거 객체를 배열로 반환*  Week tomorrow=**null; //열거형 변수 선언과 초기화**  Calendar cr=Calendar.*getInstance();*  **int week = cr.get(Calendar.*DAY\_OF\_WEEK); //일(1) ~토(7)까지 숫자 리턴***  String today = **days[week]**.**name();** //열거 객체의 문자열 반환  System.***out.println("오늘은 " + today + " 입니다");***  tomorrow = **Week.valueOf(today);** //주어진 문자열의 열거 객체로 반환  **if(tomorrow == Week.*토요일 || tomorrow == Week.일요일)***  System.***out.println("주말입니다");***  **else**  System.***out.println("평일 입니다");***  System.**out.println(tomorrow.ordinal()); //열거 객체의 순번 반환**  }  } |
| **[실행 결과]** |

* 실행결과 확인 2 : 재정의 & 다형성 & 추상클래스 & 열거형 사용하기

|  |
| --- |
| **[프로그램 소스]**  **enum** Poly { //열거형  ***Rectangle***, ***Triangle***, ***Circle***  }  **abstract** **class** Shape { // 추상 클래스, 필드와 메소드, 생성자를 가질 수 있음  **protected** **int** x, y;  **protected** **final** **static** **double** ***PI*** = 3.14;  Shape(**int** x, **int** y) {  **this**.x = x;  **this**.y = y;  }  **void** coorPrint() {  System.***out***.println("중심 좌표 : x=" + x + ", y=" + y);  }  **abstract** **void** draw();  }  **class** Rectangle **extends** Shape {  **private** **int** area; // 넓이  Rectangle(**int** x, **int** y) {  **super**(x, y);  }  **public** **void** setArea(**int** ga, **int** se) {  area = ga \* se;  }  **public** **int** getArea() {  **return** area;  }  // 메소드 재정의  @Override // 재정의 되는 메소드임을 나타내는 annotation  **public** **void** draw() {  System.***out***.println("Rectangle Draw");  coorPrint();  }  }  **class** Triangle **extends** Shape {  Triangle(**int** x, **int** y) {  **super**(x, y);  }  **public** **void** draw() {  System.***out***.println("Triangle Draw");  coorPrint();  }  }  **class** Circle **extends** Shape {  **private** **double** circum; // 원 둘레  Circle(**int** x, **int** y) {  **super**(x, y);  }  **public** **void** draw() {  System.***out***.println("Circle Draw");  coorPrint();  }  **public** **void** setCircum(**double** r) {  circum = r \* r \* ***PI***;  }  **public** **double** getCircum() {  **return** circum;  }  }  **public** **class** PolyTest {  **public** **static** **void** draw(Shape[] shape) {  **for** (Shape s : shape) {  s.draw();  **if** (s **instanceof** Rectangle) // 객체 s가 Rectangle 타입이면  System.***out***.println("넓이 : " + ((Rectangle) s).getArea());  **else** **if** (s **instanceof** Circle)  System.***out***.printf("원 둘레 : %.2f\n", ((Circle) s).getCircum());  }  }  **public** **static** **void** main(String[] args) {  Poly[] choice= Poly.*values*(); //열거 객체를 배열로 반환  Shape[] shape = **new** Shape[4];  **for** (**int** i = 0; i < shape.length; i++) {  **int** rnd = (**int**) (Math.*random*() \* 3);  **int** x = (**int**) (Math.*random*() \* 100) + 1;  **int** y = (**int**) (Math.*random*() \* 100) + 1;  **switch** (choice[rnd]) {  **case** ***Rectangle*** :  shape[i] = **new** Rectangle(x, y);  ((Rectangle) shape[i]).setArea(x, y);  **break**;  **case** ***Triangle*** :  shape[i] = **new** Triangle(x, y);  **break**;  **case** ***Circle*** :  shape[i] = **new** Circle(x, y);  ((Circle) shape[i]).setCircum(Math.*random*() \* 10);  }  }  *draw*(shape);  }  } |
| **[실행 결과]** |

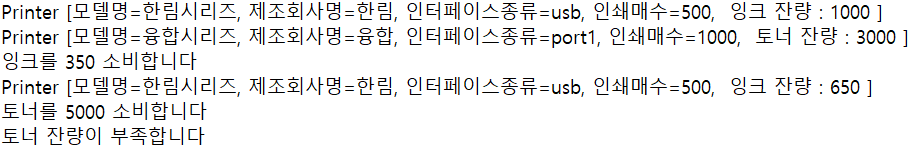
* 실습 과제

1. 다음 그림과 같은 상속 구조를 갖는 클래스를 구현하고 테스트 하시오



**상속**

**상속**



**public** **class** Answer1 {

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

Inkjet\_P inkject=**new** Inkjet\_P("한림시리즈", "한림","usb", 500, 1000);

Laser\_P laser = **new** Laser\_P("융합시리즈", "융합", "port1", 1000, 3000);

inkject.print();

laser.print();

System.***out***.println("잉크를 소비합니다");

**if**(inkject.inkdec(350) == 0)

System.***out***.println("잉크 잔량이 부족합니다");

**else**

inkject.print();

System.***out***.println("토너를 소비합니다");

**if**(laser.inkdec(5000) == 0)

System.***out***.println("토너 잔량이 부족합니다");

**else**

laser.print();

}

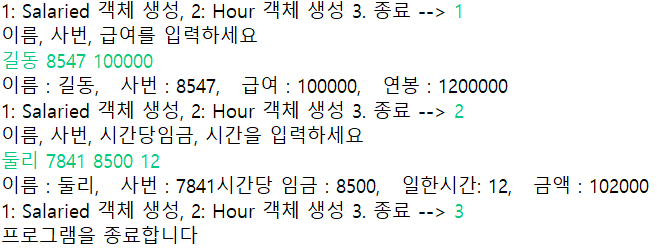
}

|  |
| --- |
| **[프로그램 소스]**    public class Printer {  protected String model;  protected String company;  protected String interType;  protected int printCnt;    public Printer(String model, String company, String interType, int printCnt) {  this.model = model;  this.company = company;  this.interType = interType;  this.printCnt = printCnt;  }  public void print() {  System.***out***.print("Printer[모델명=" + model + ", 제조회사명=" + company + ", 인터페이스종류=" + interType +  ", 인쇄매수=" + printCnt);  }  }  public class Inkjet\_P extends Printer{  int restInk;    public Inkjet\_P(String model, String company, String interType, int printCnt, int restInk) {  super(model, company, interType, printCnt);  this.restInk = restInk;  }    public int inkdec(int cnt) {  restInk -= cnt;  if(restInk < 0) {  restInk += cnt;  return 0;  }  return restInk;  }    *@Override*  public void print() {  super.print();  System.***out***.println(", 잉크 잔량: " + restInk + " ]");  }  }  public class Laser\_P extends Printer{  int restToner;    public Laser\_P(String model, String company, String interType, int printCnt, int restToner) {  super(model, company, interType, printCnt);  this.restToner = restToner;  }    public int inkdec(int cnt) {  restToner -= cnt;  if(restToner < 0)  {  restToner += cnt;  return 0;  }  return restToner;  }    *@Override*  public void print() {  super.print();  System.***out***.println(", 토너 잔량: " + restToner + " ]");  }  } |
| **[실행 결과]** |

1. 제시된 클래스를 상속과 재정의를 이용하여 설계 하시오 - 각 클래스에 대한 객체를 생성하고 테스트 하시오

|  |
| --- |
| SalariedEmployee |
| 필드 : 이름, 사번, 월급 -전용 멤버 |
| 메소드: computeSalary()=> 연봉 계산(월급 \* 12)하고 반환  설정자, 접근자 메소드, 생성자, toString()-객체 내용 출력 |

|  |
| --- |
| HourEmployee |
| 필드 : 이름, 사번, 시간당 임금, 일한 시간 - 전용 멤버 |
| 메소드: computeSalary()=> 임금 계산(시간당 임금 \* 일한 시간)하고 반환  설정자, 접근자 메소드, 생성자, toString()-객체 내용 출력 |



class Person{ //수퍼클래스

//공통되는 필드와 메소드를 작성

}

class SalariedEmployee{ //Person 클래스 상속

// computeSalary()메소드 재정의

//필요한 부분 추가

}

class HourEmployee { //Person 클래스 상속

// 필드 추가, computeSalary()메소드 재정의

//필요한 부분 추가

}

class EmployeeTest{

public static void main(String[] args){

//1번이 입력되면 SalariedEmployee 객체 생성하고 내용 출력

//2번이 입력되면 HourEmployee 객체 생성하고 내용 출력

//객체 초기화를 위해 필요한 값은 입력 받도록 한다.

}

}

|  |
| --- |
| **[프로그램 소스]**    class Person { // 수퍼클래스  // 공통되는 필드와 메소드를 작성  protected String name; //이름  protected int salaryNum; //사번    public Person(String name, int salaryNum) {  this.name = name;  this.salaryNum = salaryNum;  }    public int computeSalary() {  return 0;  }  }  class SalariedEmployee extends Person {  int monthReward;    public SalariedEmployee(String name, int salaryNum, int monthReward) {  super(name, salaryNum);  this.monthReward = monthReward;    }    public void setMonthReward(int money) {  this.monthReward = money;  }    public int getMonthReward() {  return monthReward;  }    //computeSalary()메소드 재정의  *@Override*  public int computeSalary() {  return monthReward \* 12;  }    public String toString() {  String msg = "이름: " + name + ", 사번: " + salaryNum + ", 급여: " + monthReward + ", 연봉: " + computeSalary();    return msg;  }    }  class HourEmployee extends Person {// Person 클래스 상속  int hourReward; // 시간당 임금  int workHour; // 일한 시간  public HourEmployee(String name, int salaryNum, int hourReward, int workHour) {  super(name, salaryNum);  this.hourReward = hourReward;  this.workHour = workHour;  }  public void setHourReward(int hourReward) {  this.hourReward = hourReward;  }  public int getHourReward() {  return hourReward;  }  public void setWorkHour(int workHour) {  this.workHour = workHour;  }  public int getWorkHour() {  return workHour;  }  // 필드 추가, computeSalary()메소드 재정의  *@Override*  public int computeSalary() {  return hourReward \* workHour;  }  public String toString() {  String msg = "이름: " + name + ", 사번: " + salaryNum + ", 1시간당 임금: " + hourReward +  ", 일한시간: " + workHour + ", 금액: " + computeSalary();    return msg;  }  }  import java.util.\*;  class EmployeeTest {  public static void main(String[] args) {  // 1번이 입력되면 SalariedEmployee 객체 생성하고 내용 출력  // 2번이 입력되면 HourEmployee 객체 생성하고 내용 출력  // 객체 초기화를 위해 필요한 값은 입력 받도록 한다.  Scanner sc = new Scanner(System.***in***);  int menu;  do {  System.***out***.print("1: Salaried 객체 생성, 2: Hour 객체 생성 3: 종료 --> ");  menu = sc.nextInt();  switch (menu) {  case 1:  System.***out***.println("이름, 사번, 급여를 입력하세요.");  SalariedEmployee s1 = new SalariedEmployee(sc.next(), sc.nextInt(), sc.nextInt());  System.***out***.println(s1);  break;  case 2:  System.***out***.println("이름, 사번, 시간당임금, 시간을 입력하세요.");  HourEmployee h1 = new HourEmployee(sc.next(), sc.nextInt(), sc.nextInt(), sc.nextInt());  System.***out***.println(h1);  break;  }  } while (menu != 3);  System.***out***.println("프로그램을 종료합니다.");    }  } |
| **[실행 결과]** |

1. 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

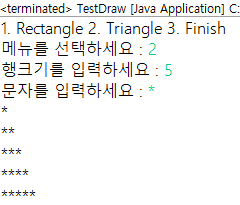
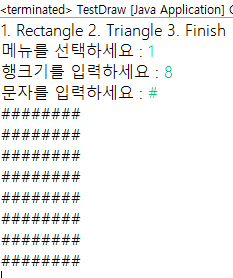
- draw() : 메소드 오버라이딩, 사다리꼴 출력

* TestDraw 클래스

- main() 메소드

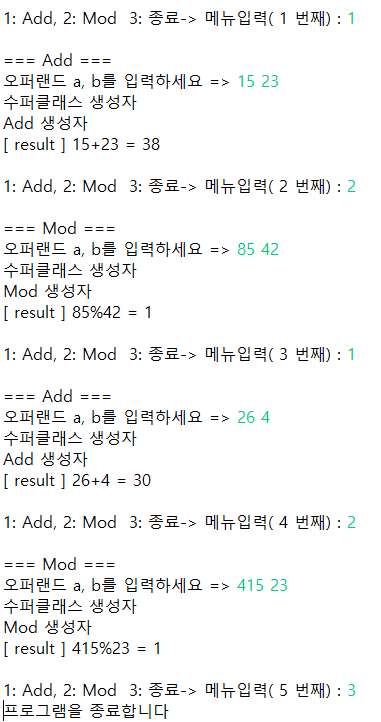
메뉴항목에서 출력하고자 하는 도형과, 행 크기, 문자를 입력 받는다

다형성을 적용할 것.

|  |
| --- |
| **[프로그램 소스]**  abstract public class ShapeDraw {    abstract public void draw(int x, char ch);  }  public class Rectangle extends ShapeDraw{    *@Override*  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 Triangle extends ShapeDraw{    *@Override*  public void draw(int x, char ch) {  for(int i = 0; i < x; i++) {  for(int j = 0; j < i; j++) {  System.***out***.print(ch);  }  System.***out***.println();  }  }  }  public class Ladder extends ShapeDraw{    *@Override*  public void draw(int x, char ch) {  for(int i = 0; i < x; ++i) {  for(int j = 0; j < i; ++j) {  System.***out***.print(" ");  }  for(int k = 0; k < x; ++k)  {  System.***out***.print(ch);  }  System.***out***.println();  }  }  }  import java.util.\*;  public class TestDraw {  public static void main(String[] args) {  Scanner sc = new Scanner(System.***in***);  int x, menu;  char ch;    do {  System.***out***.println("1.Rectangle 2.Triangle 3.Ladder 4.Finish");  System.***out***.print("만들고자하는 도형을 선택하세요: ");  menu = sc.nextInt();  if(menu == 4)  break;  System.***out***.print("행크기를 입력하세요: ");  x = sc.nextInt();  System.***out***.print("문자를 입력하세요: ");  ch = sc.next().charAt(0);  switch(menu)  {  case 1:  Rectangle r = new Rectangle();  r.draw(x, ch);  break;  case 2:  Triangle t = new Triangle();  t.draw(x, ch);  break;  case 3:  Ladder l = new Ladder();  l.draw(x, ch);  break;  }  }while(true);  System.***out***.println("프로그램을 종료합니다.");  }  } |
| **[실행 결과]** |

1. 출력 결과를 보고 제시된 프로그램을 완성 하시오. – 다형성 사용



**abstract** **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

//추상 메소드 calculate() - 필드 a, b에 대하여 연산을 실행한 후 결과를 반환 - 본인 작성

**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**;

**int** i=1;

**while**(**true**) {

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

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

**if**(menu ==3)

**break**;

**switch**(menu){

**case** 1:

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

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

obj = **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 = **new** Mod(in.nextInt(), in.nextInt(), '%'); //객체 배열 초기화

**break**;

}

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

}

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

}

}

|  |
| --- |
| **[프로그램 소스]**    abstract 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 Cal(int a, int b, char op) {  this.a = a;  this.b = b;  this.op = op;  System.***out***.println("수퍼클래스 생성자");  }  //추상 메소드 calculate() - 필드 a, b에 대하여 연산을 실행한 후 결과를 반환 - 본인 작성  abstract public int calculate();    public String toString() {  return "[ result ] " + a + op + b + " = ";  }  }  class Add extends Cal{  //Cal 클래스를 상속받는 서브클래스 Add - 본인 작성  int result;    public Add(int a, int b, char op) {  super(a, b, op);  result = 0;  }    *@Override*  public int calculate() {  result = a + b;    return result;  }    public String toString() {  String msg = super.toString();  msg += calculate();    return msg;  }    }  class Mod extends Cal { //Cal 클래스를 상속받는 서브클래스 Mod  int result;    public Mod(int a, int b, char op) {  super(a, b, op);  result = 0;  }    *@Override*  public int calculate() {  result = a % b;    return result;  }    public String toString() {  String msg = super.toString();  msg += calculate();    return msg;  }  }  import java.util.\*;  public class InherTest {  public static void main(String[] args) {  Scanner in=new Scanner(System.***in***);  Cal obj=null;  int i=1;  while(true) {  System.***out***.print( "\n1: Add, 2: Mod 3: 종료-> 메뉴입력( " + (i++) + " 번째) : ");  int menu=in.nextInt();  if(menu ==3)  break;  switch(menu){  case 1:  System.***out***.println("\n=== Add ===" );  System.***out***.print("오퍼랜드 a, b를 입력하세요 => ");  obj = 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 = new Mod(in.nextInt(), in.nextInt(), '%'); //객체 배열 초기화  break;  }  System.***out***.println(obj);  }  System.***out***.println("프로그램을 종료합니다");  }  } |
| **[실행 결과]** |

1. 다음을 만족하는 클래스 Person을 작성하시오. 단, Person 클래스에 대한 객체 생성은 불가

* 필드 :
* 이름(name, String), 나이(age, int), 자식클래스에서만 접근
* 메소드
* 생성자 : 매개변수로 받은 값을 이름과 나이 필드로 초기화
* disPlay() : 이름과 나이 출력, 형식매개변수, 반환값 없음

1. Customer 클래스 : Person 클래스 상속

* 필드:
* 고객구분(gubun, String), 포인트(point, int), private
* 메소드 :
* 생성자 : 이름과 나이는 매개변수로 받은 값으로 초기화 하고(부모클래스 생성자 사용), 포인트와, 고객구분을 나타내는 한 개의 문자를 매개변수로 받는다. 단 고객구분은 매개변수로 받은 문자를 이용하여 다음과 같이 처리한다

|  |  |
| --- | --- |
| 문자 | 고객구분 |
| g 또는 G | 일반 |
| v 또는 V | 우수 |
| s 또는 S | 최우수 |

* 포인트 필드에 대한 접근자
* disPlay() : 부모 클래스 disPlay() 재정의

이름과 나이(부모클래스 메소드 사용) 고객구분, 포인트 출력

* addPoint(int) : 반환값 없고, 매개변수 값만큼 포인트를 증가한다.

1. Student 클래스 : Person 클래스 상속

* 필드:
* 학번(num, String), 자바성적(grade, int), private
* 메소드 :
* 생성자 : 이름, 나이(부모클래스 생성자 사용), 학번, 자바성적을 매개변수로 받은 값으로 초기화
* disPlay() : 부모 클래스 disPlay() 재정의

이름과 나이(부모클래스 메소드 사용) 학번, 자바성적 출력

1. 위의 문제에서 정의한 부모클래스와 자식클래스를 사용하여 아래의 프로그램을 완성하시오. 주석에 해당하는 문장을 작성하면 됨

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

**public** **class** Answer8 {

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

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

//부모 클래스 타입으로 객체 배열을 선언하고 생성한다. 배열크기는 5 - 본인작성

//1번을 입력하면 Customer 객체를 2번을 입력하면 Student객체를 생성하여 배열에 //저장한다. 단, 필요한 값은 입력을 받으며

//이름과 나이, 학번, 점수를 입력할 때는 제시된 메소드를 사용한다 - 본인작성.

System.***out***.println("\n==== 객체 내용 출력 ====");

//배열에 저장된 객체내용을 출력하면서 Customer 객체의 포인트를 500씩 증가, 객체 내용 출력 시 foreach 구문을 사용할 것 – 본인작성

System.***out***.println("\n==== Customer 객체만 출력 ====");

//배열에 저장 된 객체 중 Customer 객체만 출력

**static** String input(String str, Scanner key){

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

**return** key.next();

}

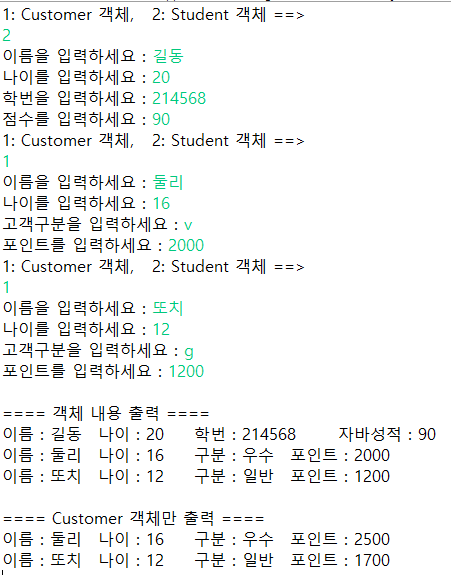
**static** **int** digit(String str, Scanner key){

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

**return** key.nextInt();

}

}



|  |
| --- |
| **[ 5~8 프로그램 소스]**  public abstract class Person {  protected String name; //이름  protected int age; //나이    public Person(String name, int age) {  this.name = name;  this.age = age;  }    public void display() {  System.***out***.print("이름: " + name + ", 나이: " + age);  }  }  public class Customer extends Person {  private String gubun; //고객구분  private int point; //포인트    public Customer(String name, int age, char rank, int point)  {  super(name, age);  this.point = point;  if(rank == 'g' || rank == 'G') {  gubun = "일반";  }  else if(rank == 'v' || rank == 'V') {  gubun = "우수";  }  else if(rank == 's' || rank == 'S') {  gubun = "최우수";  }    }    //포인트 필드에 대한 접근자  public void setPoint(int point) {  this.point += point;  }  public int getPoint() {  return point;  }    *@Override*  public void display() {  super.display();  System.***out***.println(" 고객구분: " + gubun + ", 포인드: " + point);  }    public void addPoint(int p) {  this.point += p;  }  }  public class Student extends Person{  private String num; //학번  private int grade;//자바 성적    public Student(String name, int age, String num, int grade) {  super(name, age);  this.num = num;  this.grade = grade;  }    *@Override*  public void display() {  super.display();  System.***out***.println(" 학번: " + num + ", 자바성적: " + grade);  }      }  import java.util.\*;  public class Answer8 {  public static void main(String[] args) {  Scanner key = new Scanner(System.***in***);  int menu;  String name, num;  char rank;  int age, grade, point;  // 부모 클래스 타입으로 객체 배열을 선언하고 생성한다. 배열크기는 5 - 본인작성  Person[] per = new Person[5];  // 1번을 입력하면 Customer 객체를 2번을 입력하면 Student객체를 생성하여 배열에 저장한다.  // 단, 필요한 값은 입력을 받으며 이름과 나이, 학번, 점수를 입력할 때는 제시된 메소드를 사용한다 - 본인작성.  for (int i = 0; i < 4; i++) {  System.***out***.print("1.Customer객체, 2.Student객체 ===>");  menu = key.nextInt();  switch (menu) {  case 1:  name = *input*("이름을 입력하세요: ", key);  age = *digit*("나이를 입력하세요: ", key);  System.***out***.print("고객구분을 입력하세요: ");  rank = key.next().charAt(0);  point = *digit*("포인트를 입력하세요: ", key);  per[i] = new Customer(name, age, rank, point);  break;  case 2:  name = *input*("이름을 입력하세요: ", key);  age = *digit*("나이를 입력하세요: ", key);  num = *input*("학번을 입력하세요: ", key);  grade = *digit*("점수를 입력하세요: ", key);  per[i] = new Student(name, age, num, grade);  break;  }  }  System.***out***.println("\n==== 객체 내용 출력 ====");  // 배열에 저장된 객체내용을 출력하면서 Customer 객체의 포인트를 500씩 증가, 객체 내용 출력 시 foreach 구문을 사용할 것 –  // 본인작성  for (Person p : per) {  if (p instanceof Customer) {  ((Customer)p).setPoint(500);  ((Customer)p).display();  }  else if(p instanceof Student) {  ((Student)p).display();  }  }  System.***out***.println("\n==== Customer 객체만 출력 ====");  // 배열에 저장 된 객체 중 Customer 객체만 출력  for (Person p : per) {  if (p instanceof Customer) {  ((Customer) p).display();  }  }  }  static String input(String str, Scanner key) {  System.***out***.print(str);  return key.next();  }  static int digit(String str, Scanner key) {  System.***out***.print(str);  return key.nextInt();  }  } |
| **[실행 결과]** |

1. 다음과 같이 메뉴를 처리할 수 있도록 제시된 프로그램을 완성하세요

**public** **class** MenuTest {

**public** **static** **int** menu(Scanner key) {

String menu = "1. 아메리카노 2. 카페라떼 3.레모네이드 4.케모마일 5.와플 6. 종료 --> ";

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

**return** key.nextInt();

}

**public** **static** **int** count(Scanner key, String message) {

System.***out***.print(message + " --> ");

**return** key.nextInt();

}

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

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

Menu[] menu=**new** Menu[10];

**boolean** b;

**int** hap = 0, t, i=0;

**while** (**true**) {

**int** choice = *menu*(key);

**if**(choice == 6)

**break**;

**int** cnt = *count*(key, "수량을 입력하세요 : ");

**switch** (choice) {

**case** 1:

b = *count*(key, "1. 시럽추가 2.시럽추가 없음") == 1 ? **true** : **false**;

menu[i++] = **new** Coffee("아메리카노", cnt, b);

**break**;

**case** 2:

b = *count*(key, "1. 시럽추가 2.시럽추가 없음") == 1 ? **true** : **false**;

menu[i++] = **new** Coffee("카페라떼", cnt, b);

**break**;

**case** 3:

b = *count*(key, "1. 따뜻한 음료 2. 시원한 음료") == 1 ? **true** : **false**;

menu[i++] = **new** Nade("레모네이드", cnt, b);

**break**;

**case** 4:

b = *count*(key, "1. 따뜻한 음료 2. 시원한 음료") == 1 ? **true** : **false**;

menu[i++] = **new** Nade("케모마일", cnt, b);

**break**;

**case** 5:

t = *count*(key, "1. 바닐라 2. 블루베리 3. 바나나");

menu[i++] = **new** Icecream("와플", cnt, t);

**break**;

**default**:

System.***out***.println("잘못 입력하였습니다");

}

}

System.***out***.println("\n주문하신 총 내역을 출력합니다");

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

menu[j].write();

hap += menu[j].getTotal();

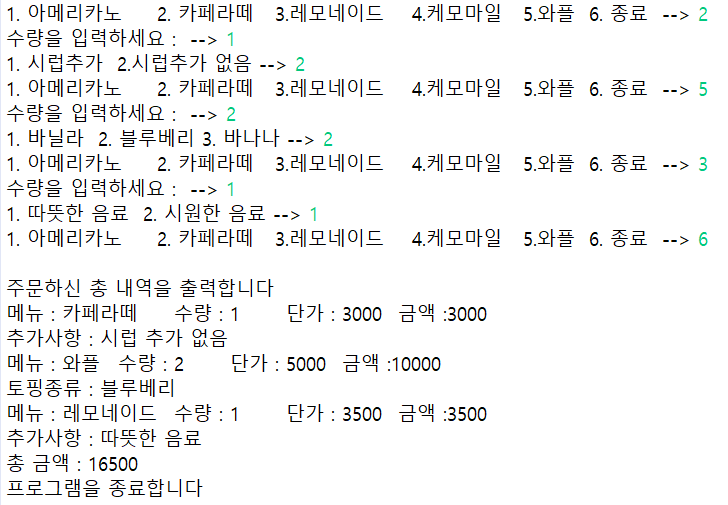
}

System.***out***.println("총 금액 : " + hap);

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

}

}



|  |
| --- |
| **[프로그램 소스]**  abstract public class Menu {  protected String name; //이름  protected int cnt; //수량    public Menu(String name, int cnt) {  this.name = name;  this.cnt = cnt;  }    public void write() {  System.***out***.print("메뉴: " + name + ", 수량: " + cnt);    }  abstract public int getTotal();  }  public class Coffee extends Menu {  String extra;  int price;  int total;    public Coffee(String name, int cnt, boolean ice) {  super(name, cnt);  if(ice)  {  if(name.equals("아메리카노"))  {  price = 2000;  }  else  {  price = 2500;  }  extra = "시럽 추가 없음";  }  else  {  if(name.equals("아메리카노"))  {  price = 2000;  }  else  {  price = 2500;  }  extra = "시럽 추가";  }  total = price \* cnt;  }    *@Override*  public void write() {  super.write();  System.***out***.println(", 단가: " + price + ", 금액: " + total +  ", 추가사항: " + extra);    }  //(수량\*가격)을 리턴해줌  *@Override*  public int getTotal()  {  return total;  }    }  public class Nade extends Menu{  String extra;  int price;  int total;    public Nade(String name, int cnt, boolean syrup) {  super(name, cnt);  if(syrup) {  extra = "따뜻한 음료";    }  else {  extra = "시원한 음료";  }  price = 3000;  total = price \* cnt;  }    *@Override*  public void write() {  super.write();  System.***out***.println(", 단가: " + price + ", 금액: " + total +  ", 추가사항: " + extra);    }  *@Override*  public int getTotal()  {  return total;  }  }  public class Icecream extends Menu{  String extra;  int price;  int total;    public Icecream(String name, int cnt, int taste) {  super(name, cnt);  if(taste == 1) {  extra = "바닐라";  }  else if(taste == 2) {  extra = "블루베리";  }  else if(taste == 3) {  extra = "바닐라";  }  price = 4000;  total = price \* cnt;  }    *@Override*  public void write() {  super.write();  System.***out***.println(", 단가: " + price + ", 금액: " + total +  ", 추가사항: " + extra);    }    *@Override*  public int getTotal()  {  return total;  }  }  import java.util.\*;  public class MenuTest {  public static int menu(Scanner key) {  String menu = "1. 아메리카노 2. 카페라떼 3.레모네이드 4.케모마일 5.와플 6. 종료 --> ";  System.***out***.print(menu);  return key.nextInt();  }  public static int count(Scanner key, String message) {  System.***out***.print(message + " --> ");  return key.nextInt();  }  public static void main(String[] args) {  Scanner key = new Scanner(System.***in***);  Menu[] menu = new Menu[10];  boolean b;  int hap = 0, t, i = 0;  while (true) {  int choice = *menu*(key);  if (choice == 6)  break;  int cnt = *count*(key, "수량을 입력하세요 : ");  switch (choice) {  case 1:  b = *count*(key, "1. 시럽추가 2.시럽추가 없음") == 1 ? true : false;  menu[i++] = new Coffee("아메리카노", cnt, b);  break;  case 2:  b = *count*(key, "1. 시럽추가 2.시럽추가 없음") == 1 ? true : false;  menu[i++] = new Coffee("카페라떼", cnt, b);  break;  case 3:  b = *count*(key, "1. 따뜻한 음료 2. 시원한 음료") == 1 ? true : false;  menu[i++] = new Nade("레모네이드", cnt, b);  break;  case 4:  b = *count*(key, "1. 따뜻한 음료 2. 시원한 음료") == 1 ? true : false;  menu[i++] = new Nade("케모마일", cnt, b);  break;  case 5:  t = *count*(key, "1. 바닐라 2. 블루베리 3. 바나나");  menu[i++] = new Icecream("와플", cnt, t);  break;  default:  System.***out***.println("잘못 입력하였습니다");  }  }  System.***out***.println("\n주문하신 총 내역을 출력합니다");  for (int j = 0; j < i; j++) {  menu[j].write();  hap += menu[j].getTotal();  }  System.***out***.println("총 금액 : " + hap);  System.***out***.println("프로그램을 종료합니다");  }  } |
| **[실행 결과]** |