❑ 응용 프로그래밍

(1) **[생성형 AI활용]** 제시된 결과처럼 실행하려고 합니다. 아래의 프로그램에 존재하는 문제는 무엇인가요? 문제를 해결할 수 있는 방법과 결과를 제시하세요.

```
ClassB :B4
ClassA : A4
ClassA : A2
ClassB : B2
ClassB : B2
```

```
class ClassA {
        public void methodOne(int i) { }
        public void methodTwo(int i) { System.out.println("ClassA : A2"); }
        public static void methodThree(int i) { }
        public static void methodFour(int i) { System.out.println("ClassA : A4"); }
}
class ClassB extends ClassA {
        public static void methodOne(int i) { }
        public void methodtwo(int i) { System.out.println("ClassB : B2"); }
        public void methodThree(int i) { }
        public static void methodFour(int i) { System.out.println("ClassB :B4"); }
}
public class Test {
        public static void main(String args[]) {
                ClassA aa = new ClassA();
                ClassB bb = new ClassB();
                ClassA ab = new ClassB();
                ClassB.methodFour(0);
                ClassA.methodFour(0);
                aa.methodTwo(0);
                bb.methodTwo(0);
                ab.methodTwo(0);
        }
}
```

생성형 AI 플랫폼	
질문	
답변	
실행결과	

(2) **[생성형 AI활용]** 다형성과 재정의를 활용하여 다음 조건에 따라 프로그램을 작성하고 테스트 하세요.

- Fruit class : me() 메소드는 실행문을 가지고 있지 않습니다
- Fruit 상속 받은 Apple class : me() 메소드에서 "나는 사과" 리턴
- Fruit 상속 받은 Grape class : me() 메소드에서 "나는 포도" 리턴
  save() 메소드에서 "저장할 없습니다" 리턴
- Fruit 상속 받은 Pear class : me() 메소드에서 "나는 배" 리턴
- Test class :
  >> main() 메소드 : Fruit 타입의 배열을 생성해 Fruit, Apple, Grape, Pear 객체를 저장하고 printFruit() 호출

```
나는 포도
저장할 수 없습니다
나는 사과
나는 포도
저장할 수 없습니다
나는 배
```

Fruit [] fary = new Fruit [4];

>> printFruit() 메소드 : 매개변수로 전달된 객체에 대하여 me() 메소드를 호출합니다. 단, 객체가 Grape이면 "저장할 수 없습니다" 도 출력

생성형 AI 플랫폼	
질문	
답변	

```
// Fruit 클래스
abstract class Fruit {
    // me() 메서드는 실행문을 가지고 있지 않은 추상 메서드로 선언
    public abstract String me();
}

// Fruit 클래스를 상속받는 Apple 클래스
class Apple extends Fruit {
    // me() 메서드 오버라이딩: "나는 사과"를 리턴
    @Override
    public String me() {
        return "나는 사과";
    }
}

// Fruit 클래스를 상속받는 Grape 클래스
class Grape extends Fruit {
    // me() 메서드 오버라이딩: "나는 포도"를 리턴
    @Override
    public String me() {
        return "나는 포도";
    }

    // save() 메서드: "저장할 수 없습니다"를 리턴
    public String save() {
        return "저장할 수 없습니다";
    }
}

// Fruit 클래스를 상속받는 Pear 클래스
class Pear extends Fruit {
    // me() 메서드 오버라이딩: "나는 배"를 리턴
    @Override
    public String me() {
        return "나는 배";
    }
}
```

```
// Test 클래스
public class Test {
	public static void main(String[] args) {
		// Fruit 타입의 배열 생성
		Fruit[] fary = {new Grape(), new Apple(), new Grape(), new Pear()};

		 // printFruit() 메서드 호출
		for (Fruit fruit : fary) {
			printFruit(fruit);
		}
	}

	// printFruit() 메서드: 매개변수로 전달된 객체의 me() 메서드를 호출
	public static void printFruit(Fruit fruit) {
		System.out.println(fruit.me());
		// Grape 타입 객체인지 확인
		if (fruit instanceof Grape grape) {
			// Grape 객체일 경우 save() 메서드의 내용을 출력
			System.out.println(grape.save());
		}
	}
}
```

실행결과

(3) 제시된 결과처럼 실행될 수 있도록 Baby와 Dog 클래스를 작성하고 실행 결과를 제시하세요.

```
신생아 : 기어 다닙니다.
신생아 : 옹알이를 합니다.
강아지 : 네 발로 다닙니다.
강아지 : 멍멍 소리를 냅니다
```

```
abstract class Animal {
	private String type; // 유형
	public Animal(String type) {
		this.type = type;
	}
	public String getType() {
		return type;
	}
	abstract void run();
	abstract void sound();
}

public class Test {
	public static void play(Animal ani) {
	  //본인 작성
```

```
	}
	public static void main(String[] args) {
		Animal animal = new Baby("신생아");
		play(animal);
		animal = new Dog("강아지");
		play(animal);
	}
}
```

[프로그램 소스]

```
abstract class Animal {
	private String type; // 유형

	public Animal(String type) {
		this.type = type;
	}
	public String getType() {
		return type;
	}
	abstract void run();
	abstract void sound();
}

class Baby extends Animal {
	public Baby(String type) {
		super(type);
	}

	@Override
	void run() {
		System.out.println(getType() + " : 기어 다닙니다.");
	}

	@Override
	void sound() {
		System.out.println(getType() + " : 옹알이를 합니다. ");
	}
}

class Dog extends Animal {
	public Dog(String type) {
		super(type);
	}
```

```
        @Override
        void run() {
                System.out.println(getType() + " : 네 발로 다닙니다.");
        }

        @Override
        void sound() {
                System.out.println(getType() + " : 멍멍 소리를 냅니다");
        }
}

public class Test {
        public static void play(Animal ani) {
                ani.run();
                ani.sound();
        }

        public static void main(String[] args) {
                Animal animal = new Baby("신생아");
                play(animal);
                animal = new Dog("강아지");
                play(animal);
        }
}
```

[실행 결과]

(4) 수퍼 클래스인 Shape에 도형 이름을 저장하는 필드와 도형의 면적(double getArea())을 계산하는 메소드를 제공합니다. 삼각형을 나타내는 클래스 Triangle은 Shape을 상속받아 작성합니다. 삼각형에 맞도록 면적(x=밑변의 길이, y=높이)을 계산하는 메소드를 재정의 합니다. 삼각형의 x, y는 생성자에서 저장합니다. 클래스 Circle도 Shape을 상속받아 작성합니다. 원에 맞도록 면적(x=반지름, y=3.14)을 계산하는 메소드를 재정의 합니다. 원의 x, y는 생성자에서 저장합니다. Test 클래스를 작성하여서 삼각형, 원 객체를 생성하고 다형성을 이용하여 각 객체의 정보와 면적을 출력하세요. (힌트 abstract class 사용)

```
public class Test {
    public static void main(String args[]) {
        Shape[ ] objs = new Shape[2];
        objs[0] = new Circle("Circle",1, 3.14);
        objs[1] = new Triangle("Rectangle",1, 2);
        for( Shape obj : objs )
            System.out.println(obj.getName() + " area = " + obj.getArea());
    }

}
```

[프로그램 소스]

```
abstract class Shape   {
        protected String name;
        protected double x, y;
        public abstract double getArea();

        public Shape(String name) {
                this.name = name;
        }

        public String getName() {
                return name;
        }
}

class Triangle extends Shape {
        public Triangle() {
                this("none",0, 0);
        }
        public Triangle(String name, double x, double y) {
                super(name);
                this.x = x;
                this.y = y;
        }
        public double getArea() {
                return 0.5 * x * y;
        }
}
class Circle extends Shape {
        public Circle() {
                this("none",0, 0);
        }
        public Circle(String name, double x, double y) {
                super(name);
                this.x = x;
                this.y = y;
        }
        public double getArea() {
                return   x *x* y;
        }
}

public class Test {
    public static void main(String args[]) {
```

```
        Shape[ ] objs = new Shape[2];

        objs[0] = new Circle("Circle",1, 3.14);
        objs[1] = new Triangle("Rectangle",1, 2);

        for( Shape obj : objs )
            System.out.println(obj.getName() + " area = " + obj.getArea());
    }
}
```

[실행 결과]

(5) AnimalTest 클래스가 다음과 같이 동작하도록 클래스들을 완성하시오. 단 Animal 클래스는 객체 생성이 불가능 합니다. reaction()은 Animal 클래스에서 인스턴스를 확인하여 출력하고, sound()는 오버라이딩하여 사용합니다. Dog는 멍멍, Cat은 야~옹 합니다.

```
<terminated> AnimalTest [Java Application] C:₩Program Files₩

Nabee고양이_꼬리내리고 야~옹
Anonymous강아지_꼬리올리고 멍멍
Anonymous고양이_꼬리내리고 야~옹
Anonymous강아지_꼬리올리고 멍멍
Rock강아지_꼬리올리고 멍멍

Pet들 중 Dog는 3마리
```

```
package test;

public class AnimalTest {
   public static void main(String[] args) {
        Animal[] pets = {
                new Cat("Nabee"),
                new Dog(),
                new Cat(),
                new Dog(),
                new Dog("Rock")
        };

        for (Animal ani : pets) {
            System.out.print("₩n" + ani + "_" + ani.reaction(ani));
            ani.sound();
        }
        System.out.print("₩n₩nPet들 중 Dog는 " + Dog.getCountDog() + "마리₩n");
   }
}
```

[프로그램 소스]

```
abstract class Animal {
        private String name;

        public Animal() {
                this("Anonymous");
        }
        public Animal(String name) {
                this.name = name;
        }
        public String reaction(Object obj) { // public String reaction(Animal obj)
                if(obj instanceof Cat) {
                        return "고양이_꼬리내리고";
```

```
                }
                else if(obj instanceof Dog) {
                        return "강아지_꼬리올리고";
                }
                else {
                        return "Animal_어떻게?";
                }
        }
        abstract void sound();

        public String toString() {
                return "₩n" + this.name;
        }
}

class Cat extends Animal{

        public Cat() {
                this("Anonymous");
        }
        public Cat(String n) {
                super(n);
        }
        public void sound() {

                System.out.print("  야~옹");

        }
}

class Dog extends Animal{
        private static int count=0;

        public Dog() {
                this("Anonymous");
        }
        public Dog(String n) {
                super(n);
                count++;
        }
        public void sound() {
                System.out.print("  멍멍");
        }
        public static int getCountDog(){
                return count;
```

```
        }
}

public class AnimalTest {
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Animal[] pets = {
                                new Cat("Nabee"),
                                new Dog(),
                                new Cat(),
                                new Dog(),
                                new Dog("Rock")
                };
                for (Animal ani : pets) {
                        System.out.print("₩n" + ani + "_" + ani.reaction(ani));
                        ani.sound();
                }
                System.out.print("₩n₩nPet들 중 Dog는 "+ Dog.getCountDog()  + "마리₩n");
        }
}
```

[실행 결과]

(6) 상속과 재정의를 사용하여 제시된 조건대로 프로그램을 작성하고 테스트 하세요.

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

HourEmployee 객체
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
}
```

```
------------------- 직원 목록 출력 --------------------
이름 : soft,  사번 : 2345,  급여 : 70,  연봉 : 840
이름 : info,  사번 : 2345,  시간당 임금 : 60,  일한시간: 6,  금액 : 360
이름 : hallym,  사번 : 6534,  급여 : 85,  연봉 : 1020
이름 : computer,  사번 : 8546,  시간당 임금 : 55,  일한시간: 8,  금액 : 440
------------------- SalariedEmployee 연봉 출력 --------------------
이름 : soft,  연봉 : 840
이름 : hallym,  연봉 : 1020
```

```
    class HourEmployee {   //Person 클래스 상속
            // 필드 추가, computeSalary()메소드 재정의
            //필요한 부분 추가
    }

    public class Test {
        public static void main(String[] args) {
          //프로그램 종료 전 모든 객체 정보 출력
        }
    }
```

[프로그램 소스]

```
 import java.util.Scanner;
abstract class Person {
        private String pName; // 이름
        private String pNum; // 사번

        public Person(String pName, String pNum) {
                this.pName = pName;
                this.pNum = pNum;
        }

        public String getName() {
                return pName;
        }

        public String getNumber() {
                return pNum;
        }

        public String toString() {
                return "이름 : " + pName + ",    사번 : " + pNum;
        }

        abstract int computeSalary();
}

// Employee 클래스에서 SalariedEmployee 나 HourEmployee 를 집합 관계로 포함
class Employee {
        private Person[] employees;
        private int count;

        public Employee(int size) {
                employees = new Person[size];
```

```
                count = 0;
        }

        public void addEmployee(Person person) {
                employees[count++] = person;
        }

        public boolean isAdd() {
                if (count >= employees.length) {
                        System.out.println("더 이상 직원을 추가할 수 없습니다.");
                        return true;
                } else {
                        return false;
                }
        }

        public void printEmployees() {
                for (int i = 0; i < count; i++) {
                        System.out.println(employees[i]);
                }
        }
}

class SalariedEmployee extends Person {
        private int pay;

        public SalariedEmployee(String pName, String pNum, int pay) {
                super(pName, pNum);
                this.pay = pay;
        }

        public int computeSalary() {
                return pay * 12;
        }

        public String toString() {
                return super.toString() + ",    급여 : " + pay + ",    연봉 : " + computeSalary();
        }
}

class HourEmployee extends Person {
        private int tPay, hours;

        public HourEmployee(String pName, String pNum, int tPay, int hours) {
```

```
                super(pName, pNum);
                this.tPay = tPay;
                this.hours = hours;
        }

        public int computeSalary() {
                return tPay * hours;
        }

        public String toString() {
                return super.toString() + ",    시간당 임금 : " + tPay + ",    일한시간: " + hours + ",    금액 :
" + computeSalary();
        }
}

// 테스트 코드
public class Main {
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                Employee employee;
                boolean flag = true;

                System.out.print("배열 크기를 입력하세요 >> ");
                employee = new Employee(in.nextInt());

                while (flag) {
                        System.out.print("1: Salaried 객체 생성, 2: Hour 객체 생성 3. 종료 --> ");
                        int menu = in.nextInt();
                        if (employee.isAdd() || menu == 3)
                                break;

                        switch (menu) {
                        case 1:
                                System.out.println("이름, 사번, 급여를 입력하세요");
                                employee.addEmployee(new SalariedEmployee(in.next(), in.next(),
in.nextInt()));
                                break;
                        case 2:
                                System.out.println("이름, 사번, 시간당임금, 시간을 입력하세요");
                                employee.addEmployee(new HourEmployee(in.next(), in.next(), in.nextInt(),
in.nextInt()));
                                break;
                        default:
                                System.out.println("잘못된 입력입니다₩n");
```

```
                                flag = false;
                        }
                }

                System.out.println("------ 모든 정보를 출력합니다 ---------");
                employee.printEmployees();
        }
}
============================
public class Main {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                Person[] employees = new Person[] { new SalariedEmployee("soft", "2345", 70),
                                new HourEmployee("info", "2345", 60, 6), new SalariedEmployee("hallym",
"6534", 85), new HourEmployee("computer", "8546", 55, 8)};

                System.out.println("------------------- 직원 목록 출력 --------------------");
                for (Person person : employees) {
                        System.out.println(person);
                }

                System.out.println("------------------- SalariedEmployee 연봉 출력 --------------------");
                for (Person person : employees) {
                        if (person instanceof SalariedEmployee )
                        System.out.printf("이름 : %s,     연봉 : %d₩n", person.getName(),
person.computeSalary());
                }
        }
}
```

[실행 결과]

(7) 생성형 AI가 제시하는 다형성을 활용하는 문제를 프로그램하고 결과를 제시 하세요. 문제 해결을 위한 코드는 답변에서 제외 되어야 합니다

생성형 AI 플랫폼	
질문	
답변	

[프로그램 소스]

[실행 결과]