제 08 장 인터페이스(실습)

□ 개념 확인 학습

1. 인터페이스 안에 정의할 수 있는 메소드 선언을 모두 선택한 후, 이유를 간단히 설명하세요.

   1) private int getArea();

   2) public float getValue(float x);

   3) public void main(String [] args);

   4) public static void main(String [] args);

   5) boolean setValue(Boolean [] test);

2. 다음 내용에 해당하는 Rectangle 클래스 정의의 첫 번째 문장만을 쓰세요.

   "클래스 Rectangle은 IDrawable 인터페이스와 IMovable 인터페이스를 구현한다."

3. 다음 내용에 해당하는 Audio 클래스 정의의 첫 번째 문장만을 쓰세요.

   "Audio는 Sound를 확장하고(=상속받고) IMP3Play 인터페이스와 ITurnTablePaly 인터페이스를 구현한다."

4. 다음 내용에 해당하는 IDrawable 인터페이스 정의의 첫 번째 문장만을 쓰세요.
   "IDrawable 인터페이스는 IPaint 인터페이스를 상속 받는다."

5. 인터페이스 안에 정의할 수 선언을 모두 선택하세요.

   1) public double PI = 3.14;

   2) public double getValue();

   3) public default double getValue() { return 0.0; };

   4) static void hello() { System.out.println("Hello") }

6. 다음의 인터페이스 정의에서 오류가 있으면 오류가 있는 문장을 올바르게 수정하세요.

```
public interface IMyInterface {
    void IMyMethod(int value) {
        System.out.println("인터페이스의 메소드 안 입니다.");
    }
}
```

7. 다음과 같은 인터페이스와 클래스가 있다고 할 때, MyClass에서 반드시 구현해야 하는 메소드를 모두 고르세요.

```
interface IA {
   public float mA(int a);
}

interface IB extends IA {
   public default int mB1(int a) { System.out.println("Here is mB1()"); };
   public Object mB2(int a);
}

class C {
   public void mC(int a) { System.out.println("Here is mC()"); }
}

public class MyClass extends C implements IB {

}
```

1) mA()
2) mB1()
3) mB2()
4) mC()

□ 적용 확인 학습 & 응용 프로그래밍

1. 다음의 인터페이스 선언과 사용에서 잘못된 점을 모두 지적하세요.

```
public interface IEdible {
   boolean amount;
   final int TYPE=10;
   public void eat() {   };
};

public class Sandwitch extends IEdible {
   public void eat() {   }
}
```

2. 클래스 Desk가 IMovable 인터페이스를 구현한다고 가정 했을 때, 다음의 문장들이 순차적으로 실행되면서 오류가 발생하는 문장을 찾고 이유를 설명하세요.

```
Desk desk = new Desk();
IMovable m = desk;
desk = m;
```

3. IControll 인터페이스를 아래와 같이 정의할 수 있습니다. IControll 인터페이스를 구현하는 IControllTest 클래스의 main() 메소드에서 **무명 내부 클래스**로 play()와 stop()을 작성하고 테스트 해 보세요.

```
public interface IControll {
  void play();
  void stop();
}
```

4. IControll 인터페이스를 아래와 같이 정의할 수 있습니다. 그리고 VideoPlayer는 play()와 stop() 조작을 할 수 있습니다. IControll 인터페이스를 구현하는 VideoPlayer 클래스를 구현하고 VideoPlayerTest 클래스의 main()에서 테스트 해 보세요.

```
public interface Icontroll {
  void play();
  void stop();
}
```

5. 추상 메소드 sound()를 가지고 있는 interface ISound를 작성하고 ISound를 구현하는 Dove 클래스를 작성하세요. Dove 클래스의 sound()에서는 “coo coo"를 출력합니다.

6. 다음과 같은 조건을 만족하는 프로그램을 작성하세요.

- interface ISound의 추상 메소드 sound()는 객체의 소리를 리턴 합니다.

- ISound를 구현한 Cat 클래스에서는 "야옹"을 리턴 합니다.

- ISound를 구현한 Dog 클래스에서는 "멍멍"을 리턴 합니다.

- SoundExample 클래스의 main() 메소드에서는 Cat 객체를 매개변수로 printSound()를 호출할 때와 Dog 객체를 매개변수로 printSound()를 호출할 때 각각 “야옹”과 “멍멍”이 출력되어야 합니다.

- SoundExample 클래스의 printSound() 메소드는 ISound 인터페이스 타입의 매개변수 전달 받아 sound() 호출 결과를 출력합니다.

7. 아래의 설명에 따라 인터페이스와 클래스들을 정의하고 프로그램을 테스트하세요.

- IDrawable 인터페이스는 draw() 추상 메소드를 가진다.

- class Shape은 interface IDrawable 구현 시 "Shape Draw" 메시지를 출력한다.

- Rectangle, Triangle, Circle 클래스들은 class Shape을 상속받는다.
- interface IDrawable 구현 시 Rectangle, Triangle, Circle 클래스들은 각각 "Rectangle Draw", "Triangle Draw", "Circle Draw"를 출력한다.

- ShapeTest 클래스의 main()에서 IDrawable 객체 배열 arrayOfShapes을 생성하고 배열의 각 원소로 Rectangle, Triangle, Circle 클래스 타입의 객체를 저장한다.
- arrayOfShapes 배열의 모든 원소가 draw()를 호출하고 결과를 확인한다.

8. 다음 인터페이스 구현의 특징을 설명하세요.

```
public interface Action {
   void work();
}
public class ActionExample {
   public static void main(String[] args) {
        Action action = new Action() {
             @Override
             public void work() {
                  System.out.println("복사를 합니다.");
             }
        };
        action.work();
   }
}
```

9. 아래의 설명에 따라 인터페이스와 클래스들을 정의하고 프로그램을 테스트하세요.

- IComparable 인터페이스는 int compareTo(Object other) 형태의 추상메소드를 가지며 현재 객체가 other 객체보다 키가 크면 1, 같으면 0, 작으면 -1을 반환한다.

- Person 클래스는 이름(name), 키(height) 필드와 객체의 정보를 출력하는 toString()을 가진다. Person 클래스의 생성자에서는 전달된 값을 이름과 키 필드에 저장한다. Person 클래스는 IComparable 인터페이스를 구현한다.

- class PersonTest는 main()과 getMaximum()으로 구성 된다.

- main()에서는 Person 타입의 배열을 선언하여 세 사람의 이름과 키를 저장하고 getMaximum()을 호출하여 가장 키가 큰 사람의 정보를 출력한다.

- getMaximum()에서는 이 IComparable 인터페이스를 이용하여서 가장 키가 큰 사람의 객체를 반환한다.

- PersonTest 클래스는 다음과 같은 형태를 가진다.

```
public class PersonTest {
   public static Person getMaximum(Person[] arr)
   {
    ....
   }

   public static void main(String arg[])
  {
    ....
   }
}
```

- 프로그램을 수행하면 아래의 그림과 같은 결과를 보인다.

```
Person [name=Benny, height=180.0]
Person [name=Daniel, height=178.0]
Person [name=Joon, height=188.0]
가장 키 큰 : Person [name=Joon, height=188.0]
```

10. 다음 인터페이스 구현의 특징을 설명하세요.

```
public interface InterA {
   public void methodA();
}
public interface InterB {
   public void methodB();
}
public interface InterC extends InterA, InterB {
   public void methodC();
}
public class ImplClass implements InterC {
   public void methodA() {
        System.out.println("ImplClass-methodA() 실행");
   }
   public void methodB() {
        System.out.println("ImplClass-methodB() 실행");
   }
   public void methodC() {
        System.out.println("ImplClass-methodC() 실행");
   }
}

public class InterExtendsTest {
   public static void main(String[] args) {
        ImplClass ic = new ImplClass();
        ic.methodA();
        ic.methodB();
        ic.methodC();
   }
}
```

11. 다음과 같은 조건을 만족하는 프로그램을 작성하세요.

(a) 인터페이스 IGraphics를 작성한다.
- 인터페이스 IGraphics에서는 int getArea()과 void draw()가 선언된다.
- 추상메소드 getArea()에서는 면적을 리턴 한다.
- 인터페이스 IGraphics의 draw()에서는 "그립니다."가 출력된다.

(b) IGraphics을 구현하는 Rectangle 클래스를 작성한다.
- Rectangle 클래스는 2개의 private 정수 필드인 length와 width을 가진다.
- Rectangle 클래스의 setValue(int l, int w) 메소드는 length와 width의 값을 설정한다.
- draw()에서는 "사각형을 그립니다."가 출력된다.

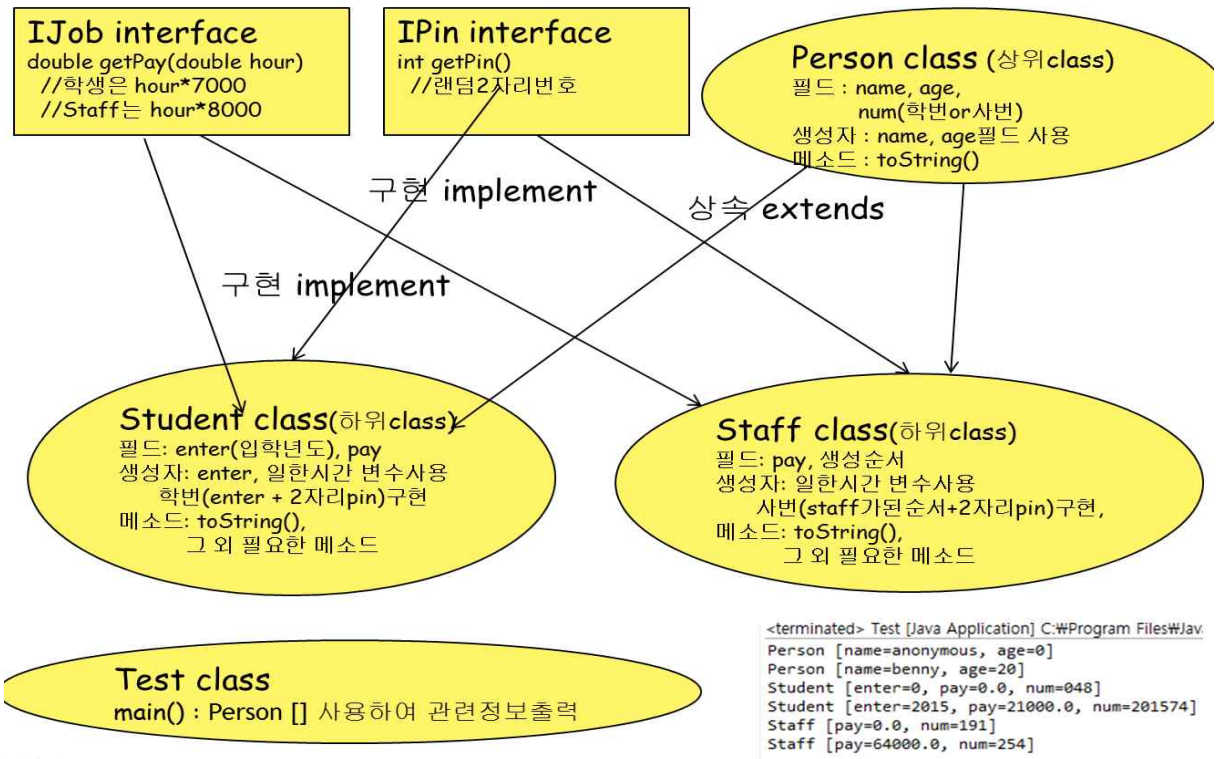
(c) RectangleTest 클래스는 다음과 같은 형태를 가진다.
- RectangleTest 클래스의 메인 메소드에서는 Rectangle 타입의 객체를 생성하고,
- 작성된 모든 메소드를 호출하여 결과를 출력한다.

```
public class RectangleTest {
   public static void main(String arg[]) {
        Rectangle r = new Rectangle();
        System.out.println("면적 = " + r.getArea());

        r.setValue(10, 10);
        System.out.println("면적 = " + r.getArea());

        r.draw();
   }
}
```

12. 다음 그림과 같은 프로그램을 완성하시오.



```
// Ipin.java
public interface IPin {
   public int getPin();
}

// Ijob.java
public interface IJob {
   public double getPay(double hour);
}

// Person.java
public class Person {
   private String name;
   private int age;
   protected String num;

   Person() {
        this("anonymous", 0);
   }

   Person(String n, int a) {
        this.name = n;
        this.age = a;
   }

   @Override
   public String toString() {
        return "Person [name=" + name + ", age=" + age + "]";
   };
}

// Staff.java
import java.util.Random;

public class Staff extends Person implements IJob, IPin {
   private static int order;
   private double pay;

   Staff() {
        this(0);
   };
   Staff(double h) {
        order++;
        this.pay = getPay(h);
        super.num = Integer.toString(order) + Integer.toString(getPin());
   };
   public double getPay(double h) {
        return h*8000;
   }
   public int getPin() {
        Random ran = new Random();
        return ran.nextInt(100);
```

```
    }
    @Override
    public String toString() {
        return "Staff [pay=" + pay + ", num=" + super.num + "]";
    }
}

// Student.java
public class Student extends Person implements IJob, IPin {
    private int enter;
    private double pay;

    Student() {
        this(0, 0);
    };
    Student(int e, double h) {
        this.enter = e;
        this.pay = getPay(h);
        super.num = Integer.toString(this.enter) + Integer.toString(getPin());
    };
    public double getPay(double h) {
        return h*7000;
    }
    public int getPin() {
        Random ran = new Random();
        return ran.nextInt(100);
    }
    @Override
    public String toString() {
        return "Student [enter=" + enter + ", pay=" + pay + ", num=" + super.num + "]";
    }
}

// Test.java
public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Person [] per = new Person[6];

        per[0] = new Person();
        per[1] = new Person("benny", 20);
        per[2] = new Student();
        per[3] = new Student(2015, 3);
        per[4] = new Staff();
        per[5] = new Staff(8);

        for(Person obj : per) {
            System.out.println(obj);
        }
    }
}
```