제 27 강

인터페이스

교재:p143~150

목차

1. 인터페이스의 정의

인터페이스(interface)란?

: inter(사이의) + face(마주하다)의 합성어로 물체들 사이에서 상호작용 할 수 있도록 하는 매개 역할

클래스 - 설계도

인터페이스 - 규격

2. 인터페이스의 문법

인터페이스의 멤버

- 상수 : final 타입 변수명 = 값;
- 추상 메서드 : abstract 리턴 타입 메서드 명( );

인터페이스의 구현 방법

```
interface 인터페이스명{
        필드
};
```

2. 인터페이스의 문법

클래스로 인터페이스 구현

```
interface 인터페이스명{


}
class 클래스명 implements 인터페이스명{
        추상 메서드 구현
}
```

2. 인터페이스의 문법

<실습> Interface1.java

```
interface A{
    int a=4;
    void methodA();
    void methodB();
}
```

```
class B implements A{
    @Override
    public void methodA() {
        System.out.println("method A");
    }
    @Override
    public void methodB() {
        System.out.println("method B");
    }
}
```

```
public class Interface1 {
    public static void main(String[] args) {
//      A a1= new A(); 인터페이스 객체 생성 불가
        A a1= new B(); //인터페이스 업 캐스팅 가능
//      a1.a=20; 인터페이스 내 상수 변경 불가
        a1.methodA();
        a1.methodB();
    }
}
```

<출력 결과>

```
method A
method B
```

2. 인터페이스의 문법

인터페이스 주의할 점!

1. 인터페이스 내 일반 메서드 불가
2. 인터페이스 내 변수 불가(상수만 가능)
3. 인터페이스로 객체 생성 불가
4. 인터페이스 내 멤버 메서드 abstract 키워드 생략 가능
5. 인터페이스 내 멤버 변수 final 키워드 생략 가능

3. 인터페이스의 실제 활용

<실습> Exam-54.java

여행 가이드 객체를 만들고 그 객체는 레저, 관광, 음식 투어를 진행할 수 있도록 구현하기

```
interface Providable{
    void leisureSports();
    void sightseeing();
    void food();
}
```

```
class KoreaTour implements Providable{
    public void leisureSports() {
        System.out.println("한강에서 수상스키 투어");
    }
    public void sightseeing() {
        System.out.println("경복궁 관람 투어");
    }
    public void food() {
        System.out.println("전주 비빔밥 투어");
    }
}
```

```
class TourGuide{
    private Providable tour =new KoreaTour();
    public void leisureSports(){
        tour.leisureSports();
    }
    public void sightseeing(){
        tour.sightseeing();
    }
    public void food() {
        tour.food();
    }
}
```

3. 인터페이스의 실제 활용

<실습> Exam-54.java

여행 가이드 객체를 만들고 그 객체는 레저, 관광, 음식 투어를 진행할 수 있도록 구현하기

```
public class Interface3 {
    public static void main(String[] args) {
        TourGuide guide = new TourGuide();
        guide.leisureSports();
        guide.sightseeing();
        guide.food();
    }
}
```

<출력 결과>

```
한강에서 수상스키 투어
경복궁 관람 투어
전주 비빔밥 투어
```

3. 인터페이스의 실제 활용

<실습> Exam-55.java

여행 가이드 객체를 만들고 그 객체는 레저, 관광, 음식 투어를 진행할 수 있도록 구현하기

TourGuide 클래스가 만약 KoreaTour 클래스가 아닌 JapanTour 클래스로 변경이 되어야 한다면?

```
class JapanTour implements Providable{
    public void leisureSports() {
        System.out.println("도쿄타워 번지점프");
    }
    public void sightseeing() {
        System.out.println("오사카 관람투어");
    }
    public void food() {
        System.out.println("초밥 투어");
    }
}
```

```
class TourGuide{
//  private Providable tour =new KoreaTour();
    private Providable tour =new JapanTour();
    public void leisureSports(){
        tour.leisureSports();
    }
    public void sightseeing(){
        tour.sightseeing();
    }
    public void food() {
        tour.food();
    }
}
```

→ 인터페이스를 사용하여 업 캐스팅을 하면,
TourGuide 와 KoreaTour 클래스 간 관계가 느슨해 지므로
코드의 수정이 간편함

4. 인터페이스와 다형성

인터페이스의 다중 구현

<구조>

interface 인터페이스1{}

interface 인터페이스 2{}

interface 인터페이스 3{}

```
class 클래스명 implements 인터페이스1, 인터페이스2, 인터페이스3…{
//                              모든 추상메서드 구현


}
```

4. 인터페이스와 다형성

<실습> Exam-56.java

인터페이스를 사용하면 MyCellPhone 클래스가 여러 기능으로 정의된 인터페이스를 다중 구현

```
interface Camera{
    void Photo();
}
interface Call{
    void calling();
}
interface Memo{
    void write();
}
interface Clock{
    void clock();
}
```

```
class MyCellPhone implements Camera,Call,Memo,Clock{
    @Override
    public void clock() {
        System.out.println("clock()");
    }

    @Override
    public void write() {
        System.out.println("write()");
    }

    @Override
    public void calling() {
        System.out.println("calling()");
    }

    @Override
    public void Photo() {
        System.out.println("photo()");
    }
}
```

```
class PhoneUser{
    void call(Call c) {
        System.out.println("전화를 걸었습니다.");
    }
}
```

4. 인터페이스와 다형성

<실습> Exam-56.java

인터페이스를 사용하면 MyCellPhone 클래스가 여러 기능으로 정의된 인터페이스를 다중 구현

모두 실행 가능

```
public class Interface4 {
    public static void main(String[] args) {
        MyCellPhone phone1=new MyCellPhone();

        Camera phone2=new MyCellPhone();
        Call phone3=new MyCellPhone();
        Memo phone4=new MyCellPhone();
        Clock phone5=new MyCellPhone();

        PhoneUser user1=new PhoneUser();
        user1.call(phone3);
        user1.call(phone1);
    }
}
```

<출력 결과>

```
전화를 걸었습니다.
전화를 걸었습니다.
```