

12. 인터페이스(interface)

〈인터페이스란?〉

- 자바에서 인터페이스는 상수와 추상메서드를 선언한 집합이다.
- 인터페이스는 상수와 추상메서드만을 갖는 추상클래스라 할 수 있다.
- 인터페이스는 추상클래스보다 더욱 완벽한 추상화를 제공할 수 있는 요인이다.

```

interface A                                //인터페이스
{
    static final int a = 1;                //인터페이스에서 속성은 final이면서 static이다.
    public int aa();
}
interface B
{
    int b = 2;                             //final과 static이 생략된 것이다. b++; 같은 연산은 불가
    public int bb();
}
interface C extends A, B                  //인터페이스가 인터페이스를 상속받을 때는 extends 사용
{
    int c = 3;
    public int cc();
}
class D implements C                     //클래스가 인터페이스를 상속받을 때는 implements 사용
{
    public int aa(){ return a; }
    public int bb(){ return b; }
    public int cc(){ return c; }
}
public class InterfaceDemo{
    public static void main(String args[]){
        D d = new D();

        [실행결과]
        System.out.println(d.aa());        1
        System.out.println(d.bb());        2
        System.out.println(d.cc());        3
    }
}
    
```

-
- 자바는 다중상속을 지원하지 않는다.
 - 자바는 여러 개의 인터페이스를 사용하여 다중상속을 흉내낼 수 있다.

2 <http://cafe.daum.net/pass365>(홍재연)

// 인터페이스를 이용한 다중상속 흉내

페가수스(pegasus;천마)는 그리스 신화에 나오는 날개를 가진 말이다.
따라서, 페가수스는 '새와 말'의 특징, 즉 다중상속을 가지는 구조이다.

```
interface Bird
{
    void fly();
}
interface Horse
{
    String owner = "포세이돈 아들 벨레로폰";
    void run();
}
public class Pegasus implements Bird, Horse{
    public void fly(){ System.out.println("새처럼"); }
    public void run(){ System.out.println("말처럼"); }
    public static void main(String args[]){
        Pegasus p = new Pegasus();
        System.out.println(p.owner);
        p.fly();
        p.run();
    }
}
```

[실행결과]

포세이돈 아들 벨레로폰

새처럼

말처럼

Tip

페가수스는 페르세우스가 메두사의 목을 자를 때 흘린 핏방울에서 생겼다. 벨레로폰이 이를 타고 키마이라를 퇴치하였고, 그 뒤로 하늘로 올라가 별자리 페가수스자리가 되었다. (시험에 나오지 않는 팁)

기출문제 분석

1. 다음 설명에 해당하는 용어는? [2009년 국가 7급]

클래스나 컴포넌트가 제공하는 서비스들을 나타내는 오퍼레이션들의 집합을 나타낸다. 함수들의 시그니처(signature)만 명세할 뿐 함수 구현은 전혀 존재하지 않으며, 속성 값도 존재할 수 없다. 구현은 상속한 클래스에서 하며, 자체는 객체를 생성할 수 없다. 클래스와의 관계는 실체화 관계로 표현한다.

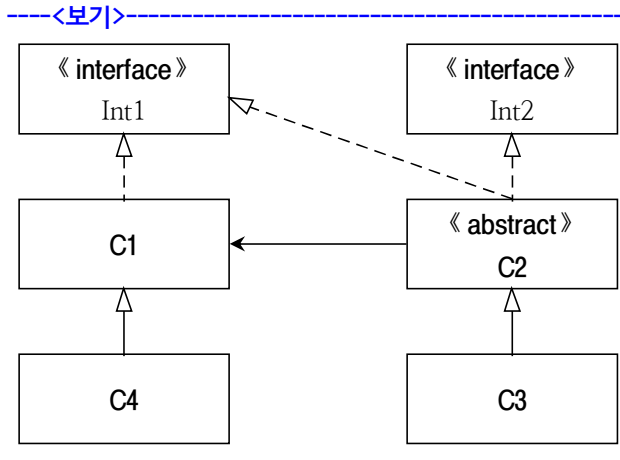
- ① 추상클래스 ② 액티브 클래스
- ③ 인터페이스 ④ 상대머신

☞ 인터페이스

- Java에서 인터페이스 구현(실체화)을 예로 들면 다음과 같다.
- 메서드 구현은 상속받은 클래스에서 처리한다.(실체화라고 함)

```
interface Bird{ void fly(); }           //메서드 시그니처만 명세할 뿐 메서드 구현은 없다.
interface Horse{ void run(); }        //메서드 시그니처만 명세할 뿐 메서드 구현은 없다.
public class Pegasus implements Bird, Horse //Pegasus는 날개 달린 말
{
    public void fly(){ System.out.println("새처럼"); } //메서드 구현은 상속받은 클래스에서
    public void run(){ System.out.println("말처럼"); } //메서드 구현은 상속받은 클래스에서
    public static void main(String args[])
    {
        Pegasus p = new Pegasus();
        p.fly();
        p.run();
    }
}
//실행 결과
새처럼
말처럼
```

2. <보기>의 클래스 다이어그램에 대한 자바코드를 컴파일할 때 오류가 발생하지 않는 것은?
[2020년 서울 7급]



- ① Int1 a = new C1(); Int2 b = new C1();
- ② Int1 c = new C2(); C3 d = new C2();
- ③ Int1 e = new C3(); C2 f = new C3();
- ④ Int1 g = new C4(); Int2 h = new C4();

♣ 객체(instance) 생성

-
- ① Int1 a = new C1(); Int2 b = new C1();
↳ **오류 발생**, Int2와 C1은 상속 관계가 아니다.
 - ② Int1 c = new C2(); C3 d = new C2();
↓
↳ **오류 발생**, 자식 = 부모; 형태는 객체를 생성 불가
오류 발생, C2는 추상클래스이므로 객체를 생성할 수 없다.
 - ③ Int1 e = new C3(); C2 f = new C3();
↳ **부모 = 자식**; 형태는 정상적으로 객체가 생성된다.
 - ④ Int1 g = new C4(); Int2 h = new C4();
↳ **오류 발생**, Int2와 C4는 상속 관계가 아니다.

// 추상클래스와 추상메서드

- 추상클래스에 선언된 추상메서드는 자식클래스에서 정의하여 사용하게 된다.
- 추상클래스는 객체(인스턴스)를 생성할 수 없다. 완성된 구조가 아니므로
- 하지만, 추상클래스 형의 객체 변수는 자신의 자식클래스의 객체를 참조할 수 있다.
- 추상클래스의 자식클래스는 일반 또는 추상클래스가 될 수 있다.

일반클래스	상속받는 추상메서드를 재정의해야 함(재정의하지 않으면 오류 발생)
추상클래스	상속받는 추상메서드를 반드시 재정의할 필요는 없음

// 인터페이스(interface)

- 자바에서 인터페이스는 상수와 추상메서드를 선언한 집합이다.
- 인터페이스는 상수와 추상메서드만을 갖는 추상클래스라 할 수 있다.
- 인터페이스는 추상클래스보다 더욱 완벽한 추상화를 제공할 수 있는 요인이다.

// 객체(instance) 생성 - A는 부모클래스, B는 자식클래스일 때

구 분	객체 생성	비 고
객체가 정상적으로 생성되는 경우	A a = new A(); B b = new B(); a = (A)b;	너무나 당연함 너무나 당연함 부모클래스 = (부모클래스)자식클래스
	A a = new B();	부모클래스 = 자식클래스 (부모 = 자식)
컴파일 오류가 발생하는 경우	B b = new A();	자식클래스 = 부모클래스 (자식 = 부모)
	A a = new A(); B b = new B(); b = (A)b;	"incompatible types" 오류 발생
실행 오류가 발생하는 경우	A a = new A(); B b = new B(); b = (B)a;	자식클래스 = 부모클래스 (자식 = 부모) "ClassCastException" 예외 발생

// 항목 ④ Int1 g = new C4();를 자바코드로 구현

클래스 다이어그램	클래스 다이어그램에 대한 것을 자바코드로 구현한 것
<pre> classDiagram class Int1 { <<interface>> } class C1 class C4 Int1 .. > C1 C1 < -- C4 </pre>	<pre> interface Int1 //인터페이스 { public void m(); //추상메서드 } class C1 implements Int1 //인터페이스를 상속받음 { public void m(){ System.out.println(100); } //추상메서드 구현 } class C4 extends C1 //클래스를 상속받음 { public void m(){ System.out.println(200); } //메서드 재정의 } public class Test { public static void main(String args[]) { Int1 g = new C4(); //객체 g 생성 g.m(); //재정의된 메서드 m() 호출 : 200 출력 } } </pre>

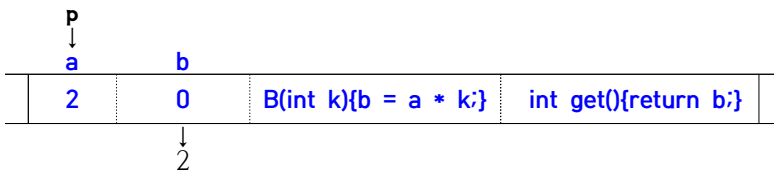
3. 다음 Java 프로그램의 실행 결과는? [2009년 지방 9급]

```
interface A{ int a = 2; }
class B implements A
{
    int b;
    B(int k){ b = a * k; }
    int get(){ return b; }
}
public class Test
{
    public static void main(String args[])
    {
        int a = 3;
        B p = new B(1);
        System.out.println(p.get());
    }
}
```

- ① 1 ② 2
③ 3 ④ 4

☞ 인터페이스

- 메모리 구조를 간단하게 그려보면



- k = 1
• a = 2
• b = a * k = 2 * 1 = 2