

클래스의 상속 3

16-1.

상속이 도움이 되는 상황의 소개

단순한 인맥 관리 프로그램: 관리 대상이 둘!

```
class UnivFriend {    // 대학 동창
    private String name;
    private String major;    // 전공
    private String phone;

    public UnivFriend
        (String na, String ma, String ph) {
        name = na;
        major = ma;
        phone = ph;
    }
    public void showInfo() {
        System.out.println("이름: " + name);
        System.out.println("전공: " + major);
        System.out.println("전화: " + phone);
    }
}
```

대학 동창 이름, 전공, 전화번호 정보 저장 및 관리

직장 동료 이름, 부서, 전화번호 정보 저장 및 관리

```
class CompFriend {    // 직장 동료
    private String name;
    private String department;    // 부서
    private String phone;

    public CompFriend
        (String na, String de, String ph) {
        name = na;
        department = de;
        phone = ph;
    }
    public void showInfo() {
        System.out.println("이름: " + name);
        System.out.println("부서: " + department);
        System.out.println("전화: " + phone);
    }
}
```

관리 대상이 둘이므로
두 개의 클래스가 정의되었다.

두 클래스를 대상을 하는 코드

```
public static void main(String[] args) {  
    UnivFriend[] ufrns = new UnivFriend[5];  
    int ucnt = 0;  
  
    CompFriend[] cfrns = new CompFriend[5];  
    int ccnt = 0;  
  
    ufrns[ucnt++] = new UnivFriend("LEE", "Computer", "010-333-555");  
    ufrns[ucnt++] = new UnivFriend("SEO", "Electronics", "010-222-444");  
  
    cfrns[ccnt++] = new CompFriend("YOON", "R&D 1", "02-123-999");  
    cfrns[ccnt++] = new CompFriend("PARK", "R&D 2", "02-321-777");  
  
    for(int i = 0; i < ucnt; i++) {  
        ufrns[i].showInfo();  
        System.out.println();  
    }  
  
    for(int i = 0; i < ccnt; i++) {  
        cfrns[i].showInfo();  
        System.out.println();  
    }  
}
```

대학 동창 관련 코드

직장 동료 관련 코드

■ 이러한 클래스 디자인 기반에서 관리 대상이 넷, 다섯으로 늘어나다면? 늘어나는 수 만큼 코드 복잡해짐

상속 기반의 문제 해결: 두 클래스 상속 관계로 묶기

```
class Friend {  
    protected String name;  
    protected String phone;  
  
    public Friend(String na, String ph) {  
        name = na;  
        phone = ph;  
    }  
    public void showInfo() {  
        System.out.println("이름: " + name);  
        System.out.println("전화: " + phone);  
    }  
}
```

“연관된 일련의 클래스들에 대해

공통적인 규약을 정의 및 적용할 수 있습니다.”

“CompFriend와 UnivFriend 클래스에 대해

Friend 클래스라는 규약을 정의하고 적용할 수 있습니다.”

```
class CompFriend extends Friend {  
    private String department;  
  
    public CompFriend(String na, String de, String ph) {  
        super(na, ph);  
        department = de;  
    }  
    public void showInfo() {  
        super.showInfo();  
        System.out.println("부서: " + department);  
    }  
}  
  
class UnivFriend extends Friend {  
    private String major;  
  
    public UnivFriend(String na, String ma, String ph) {  
        super(na, ph);  
        major = ma;  
    }  
    public void showInfo() {  
        super.showInfo();  
        System.out.println("전공: " + major);  
    }  
}
```

상속으로 묶은 결과

```
public static void main(String[] args) {  
    Friend[] frns = new Friend[10];  
    int cnt = 0;  
  
    frns[cnt++] = new UnivFriend("LEE", "Computer", "010-333-555");  
    frns[cnt++] = new UnivFriend("SEO", "Electronics", "010-222-444");  
    frns[cnt++] = new CompFriend("YOON", "R&D 1", "02-123-999");  
    frns[cnt++] = new CompFriend("PARK", "R&D 2", "02-321-777");  
  
    // 모든 동창 및 동료의 정보 전체 출력  
    for(int i = 0; i < cnt; i++) {  
        frns[i].showInfo();      // 오버라이딩 한 메소드가 호출된다.  
        System.out.println();  
    }  
}
```

이러한 클래스 디자인 기반에서 관리 대상이 넷, 다섯으로 늘어 난다면? 인스턴스 관리와 관련해서 코드가 복잡해지지 않는다.

16-2. Object 클래스와 final 선언 그리고 @Override

모든 클래스는 Object 클래스를 상속합니다.

```
class MyClass {...}
```

상속하는 클래스가 없다면

컴파일러에 의해 다음과 같이 `java.lang.Object` 클래스를 상속하게 코드가 구성된다.

```
class MyClass extends Object {...}
```

```
class MyClass extends OtherClass {...}
```

이렇듯 다른 클래스를 상속한다면 `Object` 클래스를 직접 상속하지는 않게 된다.

그러나 간접적으로(`Object` 클래스를 상속하는 클래스를 상속하는 형태로) `Object` 클래스를 상속하게 된다.

모든 클래스가 Object를 직접 또는 간접 상속하므로

```
// System.out.println
public void println(Object x) {
    . . .
    String s = x.toString();
    . . .
}
```

모든 클래스는 Object를 상속하므로 위 메소드의 인자로 전달이 가능하다.

toString 메소드는 Object 클래스의 메소드였음을 알 수 있다.

프로그래머가 정의하는 toString은 메소드 오버라이딩

```
class Cake {  
    // Object 클래스의 toString 메소드를 오버라이딩  
    public String toString() {  
        return "My birthday cake";  
    }  
}
```

```
class CheeseCake extends Cake {  
    // Cake 클래스의 toString 메소드를 오버라이딩  
    public String toString() {  
        return "My birthday cheese cake";  
    }  
}
```

클래스와 메소드의 final 선언

```
public final class MyLastCLS {...}
```

→ MyLastCLS 클래스는 다른 클래스가 상속할 수 없음

```
class Simple {
```

// 아래의 메소드는 다른 클래스에서 오버라이딩 할 수 없음

```
public final void func(int n) {...}
```

```
}
```

@Override

```
class ParentAdder {  
    public int add(int a, int b) {  
        return a + b;  
    }  
}  
  
class ChildAdder extends ParentAdder {  
    @Override  
    public double add(double a, double b) {  
        System.out.println("덧셈을 진행합니다.");  
        return a + b;  
    }  
}
```

오버라이딩이 아니라

상속으로 두 클래스에 걸쳐서 형성된 메소드 오버로딩이다.

따라서 컴파일 오류 발생

@Override

상위 클래스의 메소드를 오버라이딩 하는 것이 목적인다는 선언!

오버라이딩을 하는 형태가 아니면 컴파일러가 오류 메시지 전달.