Power Java

제12장 인터페이스와 다형성

이번 장에서 학습할 내용

- 추상 클래스
- 인터페이스
- 다형성
- 내부클래스
- 무명클래스

추상 클래스

- 추상 클래스(abstract class): 몸체가 구현되지 않은 메소드를 가지고 있는 클래스
- 추상 클래스는 추상적인 개념을 표현하는데 적당하다.

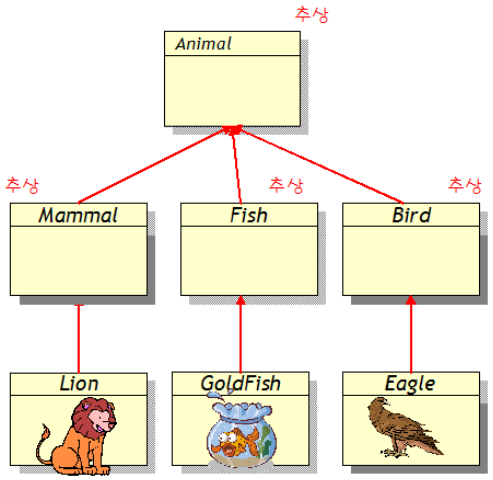


그림 12.1 추상 클래스의 개념

추상 클래스의 예

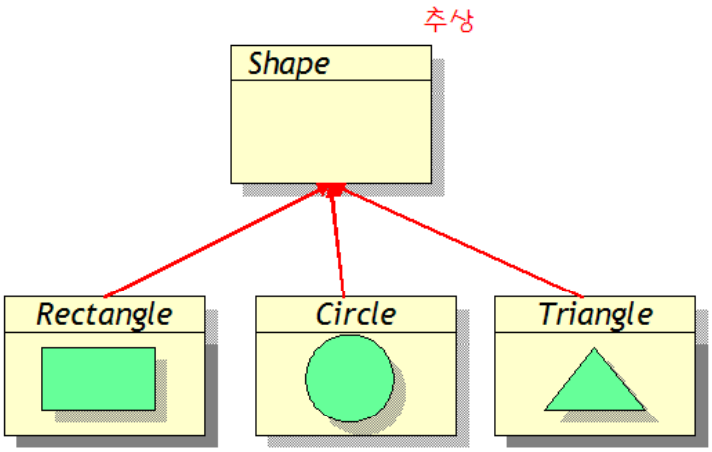


그림 12.2 도형을 나타내는 상속 계층도

추상 클래스의 예

```
abstract class Shape {                  // 추상 클래스 선언
        int x, y;

        public void move(int x, int y) // 일반 메소드 선언
        {
                this.x = x;
                this.y = y;
        }

        public abstract void draw();    // 추상 메소드 선언
};

public class Rectangle extends Shape {
        int width, height;

        public void draw() {            // 추상 메소드 구현
                System.out.println("사각형 그리기 메소드");
        }
};
```


중간 점검 문제

1. 추상 클래스의 주된 용도는 무엇인가?

2. 추상 클래스는 일반 메소드를 포함할 수 있는가?

3. 추상 클래스를 상속받으면 반드시 추상 메소드를 구현하여야 하는가?

인터페이스

- 인터페이스(interface): 추상 메소드들로만 이루어진다.

```
public interface 인터페이스_이름 {
    반환형   추상메소드1(...);
    반환형   추상메소드2(...);
    ...
}
```

```
public class 클래스_이름 implements 인터페이스_이름 {
    반환형   추상메소드1(...) {
             .....
    }
    반환형   추상메소드2(...) {
             .....
    }
}
```


인터페이스의 필요성

- 인터페이스는 객체와 객체 사이의 상호 작용을 위한 인터페이스이다.

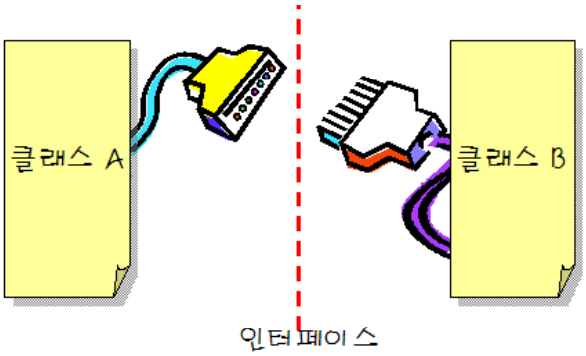


그림 12.3 인터페이스는 클래스 간의 상호 작용을 나타낸다.

인터페이스의 예

- 홈 네트워킹 예제

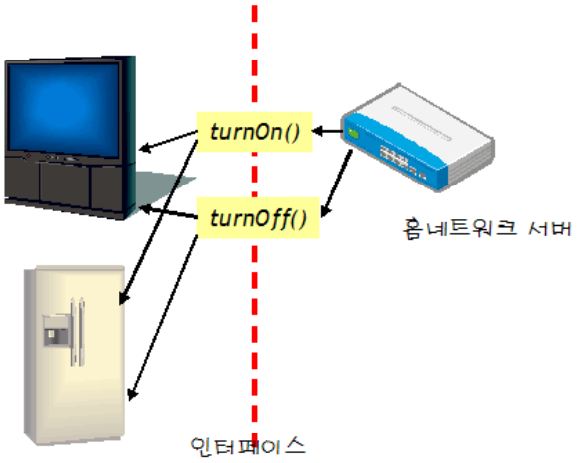


그림 12.4 홈 네트워크 예제

홈네트워킹 예제

```
public interface RemoteControl {
    // 추상 메소드 정의
    public void turnON();     // 가전 제품을 켠다.
    public void turnOFF();    // 가전 제품을 끈다.
}
```

인터페이스를 구현

```
public class Television implements RemoteControl {
    public void turnON()
    {
            // 실제로 TV의 전원을 켜기 위한 코드가 들어 간다.
            ...
    }
    public void turnOFF()
    {
            // 실제로 TV의 전원을 끄기 위한 코드가 들어 간다.
            ...
    }
}
```

홈네트워킹 예제

```
Television t = new Television();
t.turnOn();
t.turnOff();
```

인터페이스와 타입

- 인터페이스는 하나의 타입으로 간주된다.

```
public void isEqualSize(Object o1, Object o2) {
    Comparable  co1 = (Comparable)o1;       // 인터페이스 참조 변수
    Comparable  co2 = (Comparable)o2;       // 인터페이스 참조 변수
    if ( co1.compareTo(co2) == 0)
            System.out.println("두개의 객체는 같음");
    else
            System.out.println("두개의 객체는 같지 않음");
}
```

인터페이스로 참조 변수를
만들 수 있다.

다중 상속

- 다중 상속이란 여러 개의 수퍼 클래스로부터 상속하는 것
- 자바에서는 다중 상속을 지원하지 않는다.
- 다중 상속에는 어려운 문제가 발생한다.

```
class SuperA {   int x;   }
class SuperB {   int x;   }
class Sub extends SuperA, SuperB      // 만약에 다중 상속이 허용된다면
{
    ...
}
Sub obj = new Sub();
obj.x = 10;                           // obj.x는 어떤 수퍼 클래스의 x를 참조하는가?
```


다중 상속

- 인터페이스를 이용하면 다중 상속의 효과를 낼 수 있다.

```
class Shape {
        protected int x, y;
}

interface Drawable {
        void draw();
};

public class Rectangle extends Shape implements Drawable {
        int width, height;

        public void draw() {
                System.out.println("Rectangle Draw");
        }
};
```

상수 공유

```
interface Days {
    public static final int SUNDAY = 1, MONDAY = 2, TUESDAY = 3,
                        WEDNESDAY = 4, THURSDAY = 5, FRIDAY = 6,
                        SATURDAY = 7;
}

public class DayTest implements Days
{
    public static void main(String[] args)
    {
            System.out.println("일요일: " + SUNDAY);
    }
}
```

상수를 공유하려면
인터페이스를 구현하면 된다.

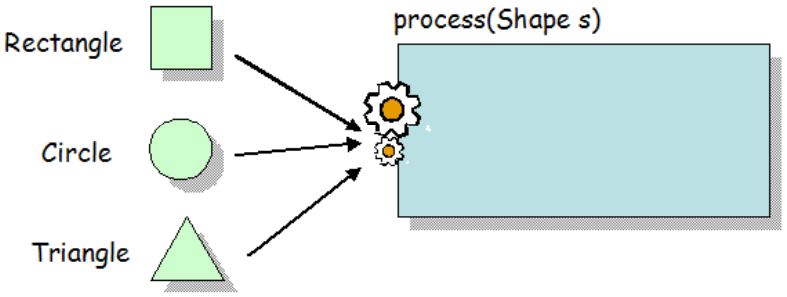
중간 점검 문제

1. 인터페이스로 참조 변수를 정의할 수 있는가?

2. 인터페이스를 이용하여서 상수를 공유하는 방법에 대해서 설명하라.

3. 자바에서는 어떻게 다중 상속을 도입하지 않고서도 다중 상속의 효과를 내는가?

다형성이란?



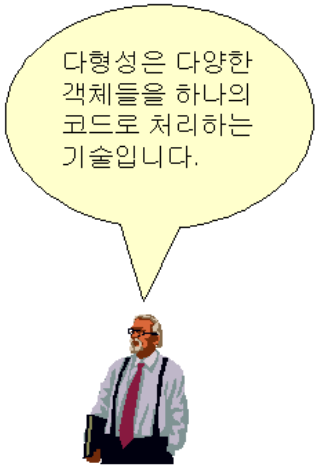


그림 12.5 다형성의 개념

상속과 객체 참조

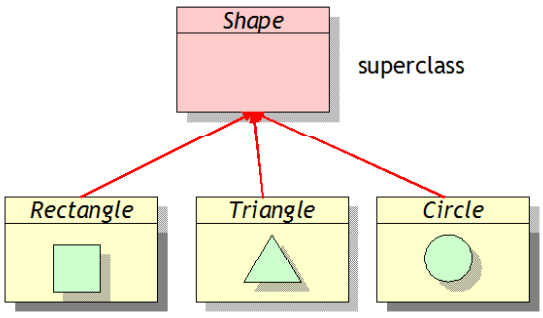
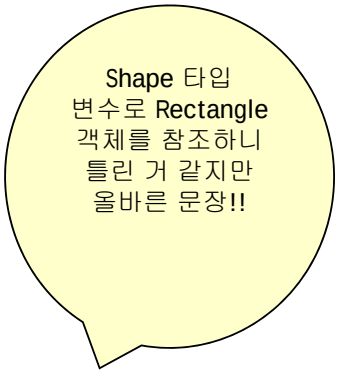


그림 12.6 도형의 상속 구조




```
Shape s = new Rectangle();// OK!
```

왜 그럴까?

- 서브 클래스 객체는 수퍼 클래스 객체를 포함하고 있기 때문이다.

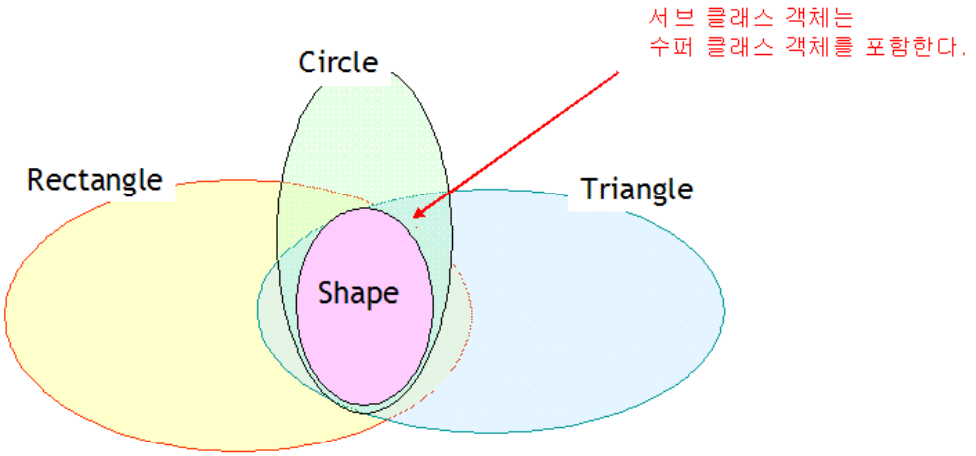


그림 12.7 서브 클래스와 수퍼 클래스의 포함 관계

동적 바인딩

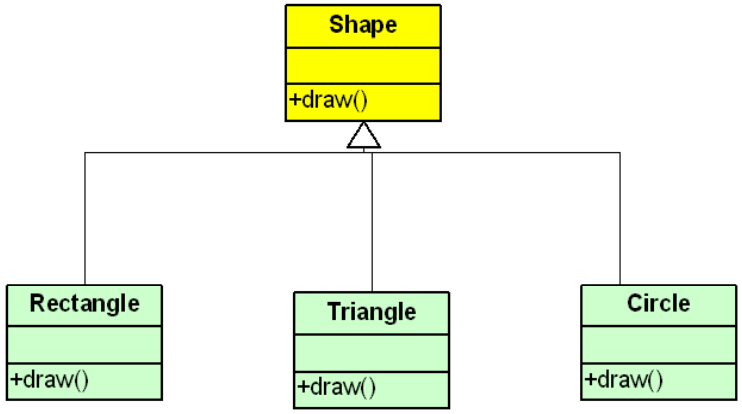


그림 12.8 도형의 UML

```
Shape s = new Rectangle();   //  OK!
s.draw();                    // 어떤 draw()가 호출되는가?
```

Shape의 draw()가 호출되는 것이 아니라 Rectangle의 draw()가 호출된다. s의 타입은 Shape이지만 s가 실제로 가리키고 있는 객체의 타입이 Rectangle이기 때문이다.

예제

ShapeTest.java

```
class Shape {
        protected int x, y;

        public void draw() {
                System.out.println("Shape Draw");
        }
};

class Rectangle extends Shape {
        private int width, height;

        public void setWidth(int w) {
                width = w;
        }

        public void setHeight(int h) {
                height = h;
        }

        public void draw() {
                System.out.println("Rectangle Draw");
        }
};
```


예제

```
class Triangle extends Shape {
        private int base, height;

        public void draw() {
                System.out.println("Triangle Draw");
        }
};

class Circle extends Shape {
        private int radius;

        public void draw() {
                System.out.println("Circle Draw");
        }
};
```

예제

```
public class ShapeTest {
        private static Shape arrayOfShapes[];

        public static void main(String arg[]) {
                init();
                drawAll();
        }

        public static void init() {
                arrayOfShapes = new Shape[3];
                arrayOfShapes[0] = new Rectangle();
                arrayOfShapes[1] = new Triangle();
                arrayOfShapes[2] = new Circle();
        }

        public static void drawAll() {
                for (int i = 0; i < arrayOfShapes.length; i++) {
                        arrayOfShapes[i].draw();
                }
        }
};
```

어떤 draw()가 호출되는가?

실행결과

```
Rectangle Draw
Triangle Draw
Circle Draw
```


다형성의 장점

- 만약 새로운 도형 클래스를 작성하여 추가한다고 해보자.

```
class Cylinder extends Shape {
    public void draw(){
            System.out.println("Cylinder Draw");
    }
};
```

- drawAll() 메소드는 수정할 필요가 없다.

```
public static void drawAll() {
        for (int i = 0; i < arrayOfShapes.length; i++) {
                arrayOfShapes[i].draw();
        }
}
```

객체의 실제 타입을 알아내는 방법

- instanceof 연산자를 사용한다.

```
Shape s = getShape();
if (s instanceof Rectangle) {
        System.out.println("Rectangle이 생성되었습니다");
} else {
        System.out.println("Rectangle이 아닌 다른 객체가 생성되었습니다");
}
```


메소드의 매개 변수

- 메소드의 매개 변수로 수퍼 클래스 참조 변수를 이용한다.

-> 다형성을 이용하는 전형적인 방법

그림 12.9 다형성을 이용하는 메소드의 매개 변수

예제

```
public static double calcArea(Shape s) {
        double area = 0.0;
        if (s instanceof Rectangle) {
                int w = ((Rectangle) s).getWidth();
                int h = ((Rectangle) s).getHeight();
                area = (double) (w * h);
        }
        ...     // 다른 도형들의 면적을 구한다.
        return area;
}
```


형변환

- Shape s = **new** Rectangle();
- s를 통하여 Rectangle 클래스의 필드와 메소드를 사용하고자 할 때는 어떻게 하여야 하는가?

```
((Rectangle) s).setWidth(100);
```

중간 점검 문제

1. 수퍼 클래스 참조 변수는 서브 클래스 객체를 참조할 수 있는가? 역은 성립하는가?

2. instanceof 연산자가 하는 연산은 무엇인가?

3. 다형성은 어떤 경우에 유용한가?

4. 어떤 타입의 객체라도 받을 수도 있게 하려면 메소드의 매개 변수를 어떤 타입으로 정의하는 것이 좋은가?

내부 클래스

- 내부 클래스(inner class): 클래스 안에 다른 클래스를 정의

```
public class OuterClass {
    // 클래스의 필드와 메소드 정의
    ...
    private class InnerClass {
        // 내부 클래스의 필드와 메소드 정의
        ...
    }
}
```

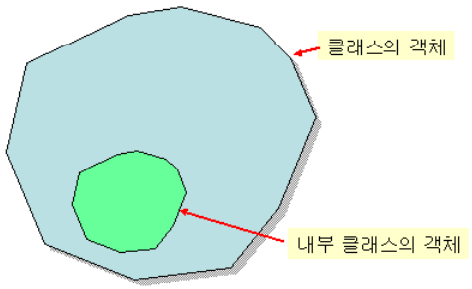


그림 12.10 내부 클래스 객체는 객체 안에 위치한다.

내부 클래스의 사용 목적

- 특정 멤버 변수를 private로 유지하면서 자유롭게 사용할 수 있다.
- 특정한 곳에서만 사용되는 클래스들을 모을 수 있다.
- 보다 읽기 쉽고 유지 보수가 쉬운 코드가 된다.

예제

```
class OuterClass {
        private String secret = "Time is money";

        public OuterClass() {
                InnerClass obj = new InnerClass();
                obj.method();
        }

        private class InnerClass {
                public InnerClass() {
                        System.out.println("내부 클래스 생성자입니다.");
                }

                public void method() {
                        System.out.println(secret);
                }
        }

}

public class OuterClassTest {
        public static void main(String args[]) {
                new OuterClass();
        }
}
```

내부 클래스

실행결과

```
내부 클래스 생성자입니다.
Time is money
```

중간 점검 문제

1. 내부 클래스와 일반 클래스의 차이점은 무엇인가?

2. 내부 클래스는 정의된 클래스의 전용 필드에 접근할 수 있는가?

무명 클래스

- 무명 클래스(anonymous class): 클래스 몸체는 정의되지만 이름이 없는 클래스

```
new ClassOrInterface() {    클래스 몸체    }
```

부모 클래스 이름이나
인터페이스 이름

무명 클래스의 예

```
interface RemoteControl {
        void turnOn();
        void turnOff();
}

public class AnonymousClassTest {
        public static void main(String args[]) {
                RemoteControl ac = new RemoteControl() {          // 무명 클래스 정의
                        public void turnOn() {
                                System.out.println("TV turnOn()");
                        }

                        public void turnOff() {
                                System.out.println("TV turnOff()");
                        }
                };
                ac.turnOn();
                ac.turnOff();
        }
}
```


중간 점검 문제

1. 무명 클래스 작성 시에 new 다음에는 적어야 하는 것은?

2. 무명 클래스를 사용하는 경우의 이점은 무엇인가?

3. Object 클래스를 상속받는 무명 클래스를 하나 정의하여 보자.

Q & A