컴퓨터 프로그래밍1

실습 14주차

- 인터페이스와 다형성 -

인터페이스 (1/6)

추상 클래스

- 추상클래스는 완전하게 구현되어 있지 않은 메소드를 가지고 있는 클래스를 의미
- 객체지향 언어에서 개념적인 설명을 위해 사용

추상클래스 : 추상메소드를 가지고 있는 클래스

```
public abstract class Animal {
    public abstract void move();
}
```

추상메소드 정의 :;으로 종료됨을 주의

인터페이스 (2/6)

추상 클래스의 예

```
public abstract class Shape {
    int x, y;

    public void move(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public abstract void draw();
}

public class Rectangle extends Shape {
    int width, height;
    public void draw(){
        System.out.println("사각형 그리기 메소드");
    }
}
```

추상클래스 Shape 을 선언. 추상클래스로는 객체 생성을 할 수 없다.

추상클래스라 하더라도 추상메소드 외에 일반 메소드도 가질수 있음에 유의

추상메소드 선언. 추상메소드를 하나라도 가지면 추상클래스가 된다. 추상메소드를 가지고 있는데도 abstract를 class 앞에 붙이지 않으면 오류 발생

서브클래스 Ractangle에서 수퍼클래스의 추상 메소드 draw()가 실제 메소드로 구현한다.

인터페이스 (3/6)

인터페이스

- 인터페이스는 추상클래스를 극단적으로 사용한 경우
- 인터페이스는 추상 메소드로만 이루어짐

```
Public interface 인터페이스 이름{
        반환형 추상메소드1(...);
        반환형 추상메소드2(...);
        ...
}
Public class 클래스이름 interface 인터페이스 이름{
        반환형 추상메소드1(...){
                ...
        }
        반환형 추상메소드2(...){
                ...
        }
        ...
}
```

인터페이스 안에는 추상메소드들이 정의된다.

인터페이스를 구현하는 클래스는 추상 메소드의 몸체를 구현하여야 한다.

인터페이스 (4/6)

인터페이스의 예

```
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.
        ...
    }
}
```

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

인터페이스 (5/6)

인터페이스의 역할

- 인터페이스는 하나의 타입으로 보아야 함
- 이 타입은 인터페이스를 구현한 클래스들을 하나로 묶는 역할

```
RemoteControl obj = new Television();
obj.turnOn();
obj.turnOff();
```

Television 객체이지만 RemoteControl 인터페이스를 구현하기 때문에 RemoteControl타입의 변수로 가리킬 수 있다.

obj를 통해서는 RemoteControl 인터페이스에 정의된 메소드만을 호출할 수 있다.

인터페이스 (6/6)

프로그래밍 예제

```
public interface Comparable {
    int compareTo(Object other);
}
```

```
public class Box implements Comparable {
    private double volume = 0;

    public Box(double v) {
        volume = v;
    }

    public int compareTo(Object otherObject) {
        Box other = (Box) otherObject;
        if (this.volume < other.volume)
            return -1;
        else if (this.volume > other.volume)
            return 1;
        else
            return 0;
    }
}
```

```
public class BoxTest {
    public static void main(String[] args) {
        Box b1 = new Box(100);
        Box b2 = new Box(85.0);
        if (b1.compareTo(b2) < 0)
            System.out.println("b1이 b2보다 더 크다");
        else
            System.out.println("b1이 b2와 같거나 작다.");
    }

}
```

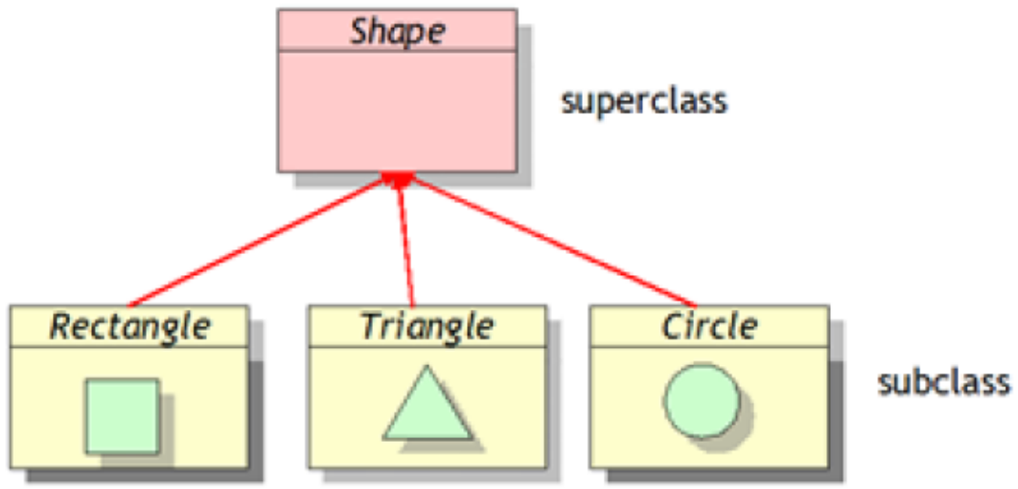
다형성 (1/6)

다형성

- 다형성이란 객체들의 타입이 다르면 똑같은 메시지가 전달되더라도 서로 다른 동작을 하는 것을 말함

상향 형변환

- 수퍼클래스 객체로 서브 클래스의 객체를 사용하는 것 (부모 클래스 obj = new 자식 클래스();)




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

다형성 (2/6)

프로그래밍 예제

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

class Rectangle extends Shape {
    public int width, height;
};
```

```
public class ShapeTest {
    public static void main(String arg[]) {
        Shape s;
        Rectangle r = new Rectangle();
        s = r;
        s.x = 0;
        s.y = 0;
        s.width = 100;
        s.height = 100;
    }
};
```

수퍼 클래스의 참조변수로 서브 클래스의 객체를 가리키는 것은 합법적이다.

shape클래스의 필드와 메소드에 접근하는 것은 OK

컴파일 오류가 발생한다. s를 통해서는 Rectangle클래스의 필드와 메소드에 접근할 수 없다.

다형성 (3/6)

다형성의 이용

- 매소드의 매개변수를 선언할 때

```
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;
}
```

다형성 (4/6)

다형성의 이용

- 다형성을 메소드의 재정의와 연결시켜서 객체들이 동일한 메시지를 받더라도 각 객체의 타입에 따라서 서로 다른 동작을 하게 함

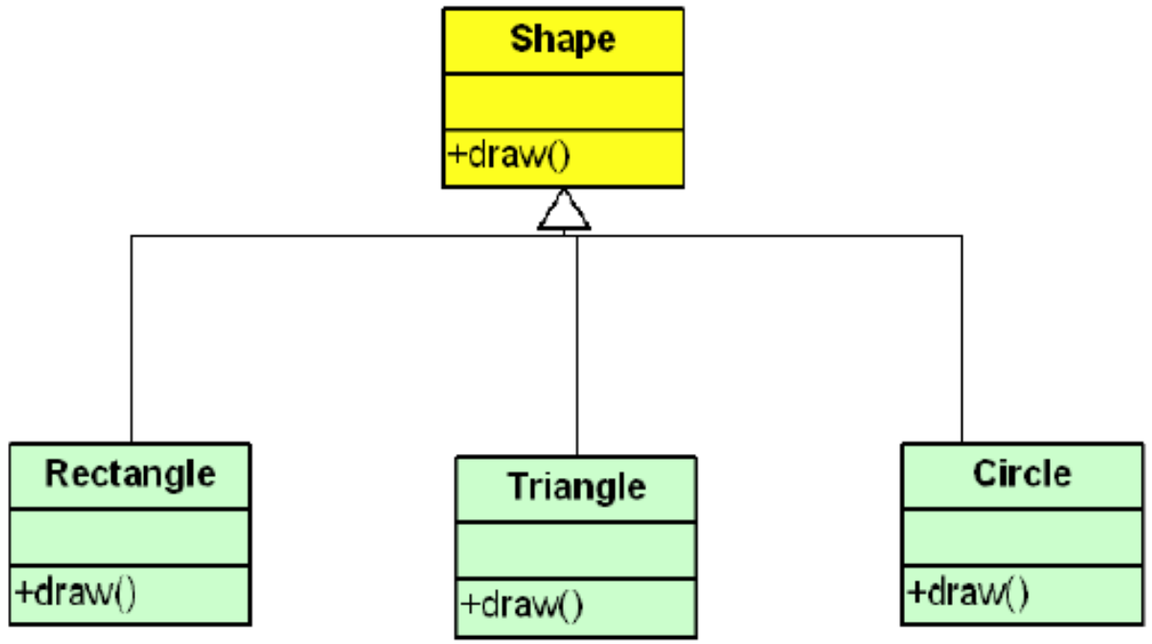


그림 12.8 도형의 UML

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

다형성 (5/6)

프로그래밍 예제

```
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");
        }
};
```

```
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");
        }
};
```

다형성 (6/6)

프로그래밍 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();
                }
        }
};
```

실행결과

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

실습

프로그래밍 실습

- 다음과 같은 인터페이스들을 정의하라

```
public interface Movable {
        void move(int dx, int dy);
}
```

- 슬라이드 12페이지에 등장하는 2차원 도형인 원, 사각형, 삼각형이 위의 인터페이스를 사용하도록 수정
- Move() 메소드는 도형의 기준점을 이동시킴
- Move() 메소드 호출 시 : 사각형-위로 한 칸, 원-아래로 한 칸, 오른쪽으로 한 칸, 삼각형 – 왼쪽으로 한 칸 이동
- main()에서 Movable 객체 배열을 생성하고배열의 각 원소에 대하여 move(정수,정수)를 호출하여 객체를 이동시키는 프로그램을 작성하라

과제

다형성 이용한 도형의 면적 출력 프로그램 작성

- 도형(삼각형, 사각형, 원)의 넓이를 계산하는 프로그램 작성
- 슈퍼 클래스 : Shape
- 서브 클래스 : Triangle, Rectangle, Circle
- 슈퍼 클래스의 멤버 변수 : int width, int height
- 슈퍼 클래스의 메소드 : Shape(int width, int height), int area()
- 서브 클래스의 메소드 : int area()

- main 함수
  - 사용자에게 width와 height를 입력 받아 삼각형, 사각형, 원형의 객체 생성
  - Shape의 배열 shape[3]를 선언하여 삼각형, 사각형, 원형을 배열에 저장
  - 반복문과 배열 shape[3]를 이용하여 각 도형의 넓이를 출력

※ 실제 도형의 넓이를 구하는 함수는 각 서브 클래스의 int area() 메소드에 구현됨

Q&A

수고하셨습니다