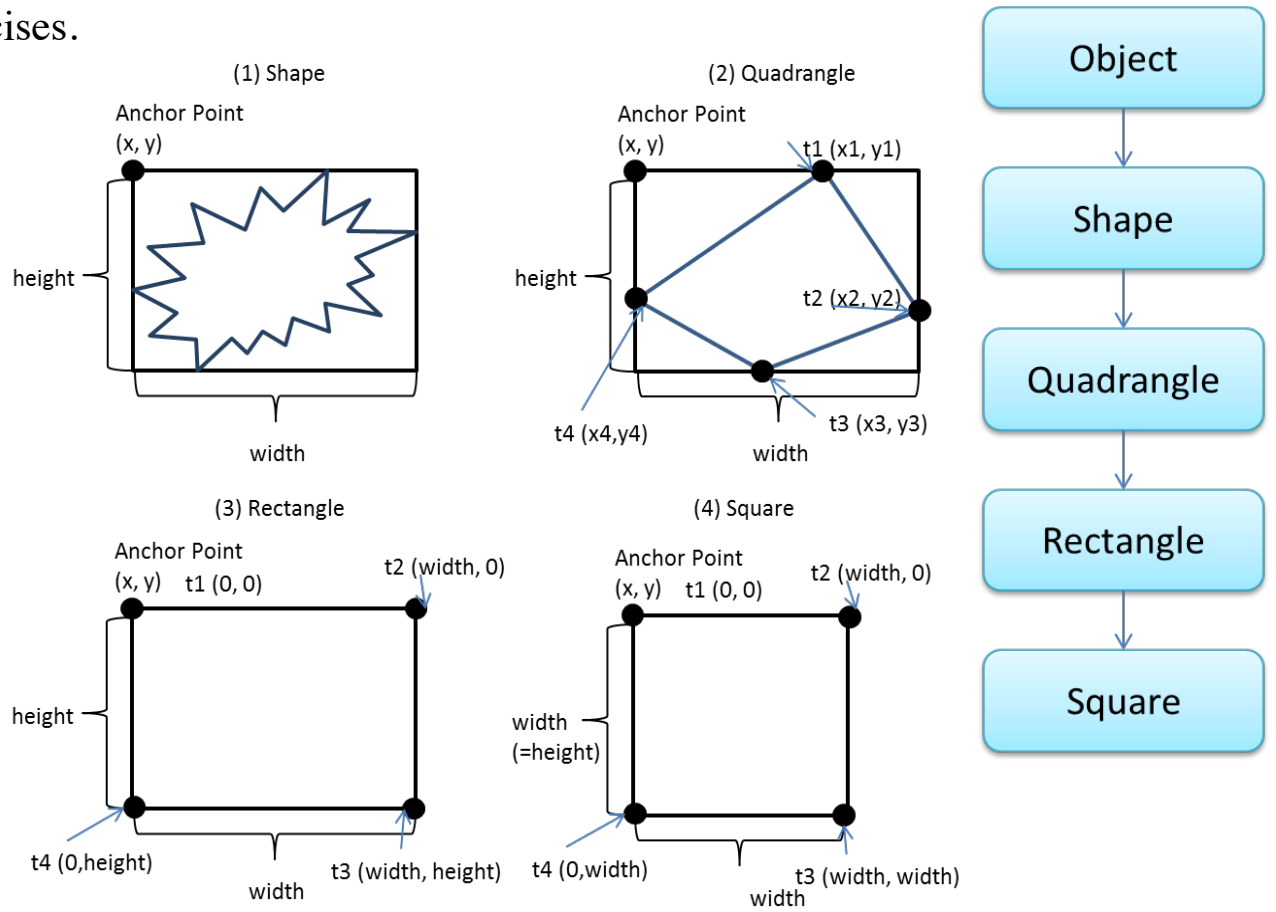
Special Care Set for Java Begineers/ Ex 07

・Notice!
The motivation of this page is to lead regular exercises.
We prepare many useful information in this page as hints of exercises, but this page has no answer.
If you have question, please don't hesitate to ask and discuss your teacher and teaching assistants.
このページは、通常の演習問題への導きとして作られています。
多くの有用な情報をヒントとして準備しておりますが、ここに回答が隠されているとかはありません。
もし、演習に対して質問等ございましたら、遠慮なく教員・TAに質問・議論をしてください。



Example Problem: 多様性 - モノのカタチはそれぞれ違う -
Polymorphism - Shape is different each other -

前回に引き続いて、今回もカタチの定義、継承から、今度は『出力』のMethodに焦点を当てて考えてみましょう。

1. 前回のおさらい - Shapeクラスとその派生クラス

前回まで、Shapeクラスとその派生クラス、『Quadrangle』,『Rectangle』,『Square』について見てきました（右図）。前回までの実装では、いかにそのクラスを『定義するか？』に焦点を当てて見てきました。実際にコーディングしてみた感想はいかがだったでしょうか？Shapeクラスにほとんどの機能が当てられていて、他のクラスにはさほどこれといった機能の追加が無かったと思います。

前回までのクラスは次の通りです。

```
public class ShapeTest {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Object obj1 = new Shape(0,0,10,10);
		Object obj2 = new Quadrangle(10,10,5,0, 10,10, 5,15, 0,5);
		Object obj3 = new Rectangle(20,20,15,10);
		Object obj4 = new Square(30,30,15);

		System.out.println(obj1.toString());
		System.out.println(obj2.toString());
		System.out.println(obj3.toString());
		System.out.println(obj4.toString());
	}
```

```
}

/* Shapeクラス */
class Shape{
    protected int x;
    protected int y;
    protected int width;
    protected int height;

    public Shape(){
        this.x = 0;
        this.y = 0;
        this.width = 0;
        this.height = 0;
    }

    public Shape(int w, int h){
        this.x = 0;
        this.y = 0;
        this.width = w;
        this.height = h;
    }

    public Shape(int x, int y, int w, int h){
        this.x = x;
        this.y = y;
        this.width = w;
        this.height = h;
    }

    public int getx(){
        return this.x;
    }

    public int gety(){
        return this.y;
    }

    public int getWidth(){
        return this.width;
    }

    public int getHeight(){
        return this.height;
    }

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

    public String toString(){
                return "x: " + x + ", y: " + y + ", Width: " + width + ", Height: " + height;
    }
}

/* Quadrangleクラス */
class Quadrangle extends Shape{

    protected int x1, y1, x2, y2, x3, y3, x4, y4;

    public Quadrangle(){
        super();
        this.x1=0;  this.y1=0;
        this.x2=0;  this.y2=0;
        this.x3=0;  this.y3=0;
        this.x4=0;  this.y4=0;
    }

    public Quadrangle(int x, int y){
        super(x,y);
        this.x1=0;  this.y1=0;
        this.x2=0;  this.y2=0;
        this.x3=0;  this.y3=0;
```

```
        this.x4=0;  this.y4=0;
    }

    public Quadrangle(int x, int y, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
        super(x, y, x2-x4, y3-y1);
        this.x1=x1;  this.y1=y1;
        this.x2=x2;  this.y2=y2;
        this.x3=x3;  this.y3=y3;
        this.x4=x4;  this.y4=y4;
    }

    public void setQuadrangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
        this.x1=x1;  this.y1=y1;
        this.x2=x2;  this.y2=y2;
        this.x3=x3;  this.y3=y3;
        this.x4=x4;  this.y4=y4;
        this.width = x2-x4;  this.height= y3-y1;
    }

    public String toString(){
        return super.toString() + "\n (x1,y1)=(" +(x+x1)+","+(y+y1)+"), (x2,y2)=(" +(x+x2)+","+(y+y2)+
                               "), (x3,y3)=(" +(x+x3)+","+(y+y3)+"), (x4,y4)=(" +(x+x4)+","+(y+y4)+")";
    }
}

/* Rectangleクラス */
class Rectangle extends Quadrangle{

    public Rectangle(){
        super();
    }

    public Rectangle(int x, int y){
        super(x,y);
    }

    public Rectangle(int x, int y, int width, int height){
        super(x, y, 0, 0, width, 0, width, height, 0, height);
    }

    public void setRectangle(int width, int height){
        super.setQuadrangle(0,0,width,0,width,height,0,height);
    }

    public String toString(){
        return "[Rectangle] " + super.toString();
    }
}

/* Squareクラス */
class Square extends Rectangle{

    public Square(){
        super();
    }

    public Square(int x, int y){
        super(x,y);
    }

    public Square(int x, int y, int width){
        super(x, y, width, width);
    }

    public void setSquare(int width){
        super.setRectangle(width,width);
    }
    public String toString(){
        return "[Square]" + super.toString();
    }

}
```

2. 同じメソッドでも中身が違う - 『多様性』

Shapeクラス群では、出力にtoString()メソッドを用いて出力をしていました。
この様な形のメソッドは、Objectの本来の型の中に含まれるメソッドを用いて実行されます。
実際、今の状態での出力は次の通りになっています。

```
x: 0, y: 0, Width: 10, Height: 10
x: 10, y: 10, Width: 10, Height: 15
 (x1,y1)=(15,10), (x2,y2)=(20,20), (x3,y3)=(15,25), (x4,y4)=(10,15)
[Rectangle] x: 20, y: 20, Width: 15, Height: 10
 (x1,y1)=(20,20), (x2,y2)=(35,20), (x3,y3)=(35,30), (x4,y4)=(20,30)
[Square][Rectangle] x: 30, y: 30, Width: 15, Height: 15
 (x1,y1)=(30,30), (x2,y2)=(45,30), (x3,y3)=(45,45), (x4,y4)=(30,45)
```

宣言は全てObjectクラスで行われていますが、新たに作ったオブジェクトはそれぞれ別な型を使いました。
なので、全てObjectクラスから呼ばれているにもかかわらず、それぞれの本来のクラス内メソッドが呼ばれることになりました。

3. Drawableインターフェース - 多様性を担保するための『インターフェース』

さて、多様性を担保させるために、実装時に『必ずメソッドやフィールドを含ませてくださいね』と教えるように出来るものがあります。
それを『インターフェース』と言います。
継承は上位の1つしか継承出来ないのに対して、インターフェースは複数持たせることが出来ます。
今回は、形（Shape）を描けるようにするDrawableインターフェースを考えてみたいと思います。
Drawableインターフェースを追加したソースコードは次の通りになります。

```
public class ShapeTest {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Object obj1 = new Shape(0,0,10,10);
		Object obj2 = new Quadrangle(10,10,5,0, 10,10, 5,15, 0,5);
		Object obj3 = new Rectangle(20,20,15,10);
		Object obj4 = new Square(30,30,15);

		System.out.println(obj1.toString());
		System.out.println(obj2.toString());
		System.out.println(obj3.toString());
		System.out.println(obj4.toString());
		System.out.println(((Rectangle)obj4).toString());
		((Quadrangle) obj2).draw();
		((Rectangle) obj3).draw();
		((Square) obj4).draw();
		((Rectangle) obj4).draw();
	}

}
interface Drawable{
	void draw();
}
/* Shapeクラス */
class Shape{
    protected int x;
    protected int y;
    protected int width;
    protected int height;

    public Shape(){
```

```
    this.x = 0;
    this.y = 0;
    this.width = 0;
    this.height = 0;
}

public Shape(int w, int h){
    this.x = 0;
    this.y = 0;
    this.width = w;
    this.height = h;
}

public Shape(int x, int y, int w, int h){
    this.x = x;
    this.y = y;
    this.width = w;
    this.height = h;
}

public int getx(){
    return this.x;
}

public int gety(){
    return this.y;
}

public int getWidth(){
    return this.width;
}

public int getHeight(){
    return this.height;
}

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

public String toString(){
            return "x: " + x + ", y: " + y + ", Width: " + width + ", Height: " + height;
}
}

/* Quadrangleクラス */
class Quadrangle extends Shape implements Drawable{

    protected int x1, y1, x2, y2, x3, y3, x4, y4;

    public Quadrangle(){
        super();
        this.x1=0;  this.y1=0;
        this.x2=0;  this.y2=0;
        this.x3=0;  this.y3=0;
        this.x4=0;  this.y4=0;
    }

    public Quadrangle(int x, int y){
        super(x,y);
        this.x1=0;  this.y1=0;
        this.x2=0;  this.y2=0;
        this.x3=0;  this.y3=0;
        this.x4=0;  this.y4=0;
    }

    public Quadrangle(int x, int y, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
        super(x, y, x2-x4, y3-y1);
        this.x1=x1;  this.y1=y1;
        this.x2=x2;  this.y2=y2;
        this.x3=x3;  this.y3=y3;
        this.x4=x4;  this.y4=y4;
    }
```

```
    public void setQuadrangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){

        this.x1=x1;  this.y1=y1;
        this.x2=x2;  this.y2=y2;
        this.x3=x3;  this.y3=y3;
        this.x4=x4;  this.y4=y4;
        this.width = x2-x4;  this.height= y3-y1;
    }

    public String toString(){
        return super.toString() + "\n (x1,y1)=(" +(x+x1)+","+(y+y1)+"), (x2,y2)=(" +(x+x2)+","+(y+y2)+
             "), (x3,y3)=(" +(x+x3)+","+(y+y3)+"), (x4,y4)=(" +(x+x4)+","+(y+y4)+")";
    }

        @Override
        public void draw() {
                // TODO Auto-generated method stub
                System.out.println("Draw Quadrangle:" + toString());
        }
}

/* Rectangleクラス */
class Rectangle extends Quadrangle implements Drawable{

    public Rectangle(){
        super();
    }

    public Rectangle(int x, int y){
        super(x,y);
    }

    public Rectangle(int x, int y, int width, int height){
        super(x, y, 0, 0, width, 0, width, height, 0, height);
    }

    public void setRectangle(int width, int height){
        super.setQuadrangle(0,0,width,0,width,height,0,height);
    }
    public String toString(){
        return "[Rectangle] " + super.toString();
    }

        @Override
        public void draw() {
                // TODO Auto-generated method stub
                System.out.println("Draw Rectangle:" + toString());
        }

}

/* Squareクラス */
class Square extends Rectangle{

    public Square(){
        super();
    }

    public Square(int x, int y){
        super(x,y);
    }

    public Square(int x, int y, int width){
        super(x, y, width, width);
    }

    public void setSquare(int width){
        super.setRectangle(width,width);
    }
    public String toString(){
        return "[Square]" + super.toString();
    }
```

```
	@Override
	public void draw() {
		// TODO Auto-generated method stub
		System.out.println("Draw Square:" + toString());
	}

}
```

implements Drawable　とした所のクラス内にそれぞれ、draw()メソッドが追加されていると思います。
このメソッドを通して出力を行えば、出力はわざわざクラスが何かを判別することなく、ただdraw()メソッドを動かすだけで動作することになります。
出力結果は次の通りです。

```
x: 0, y: 0, Width: 10, Height: 10
x: 10, y: 10, Width: 10, Height: 15
 (x1,y1)=(15,10), (x2,y2)=(20,20), (x3,y3)=(15,25), (x4,y4)=(10,15)
[Rectangle] x: 20, y: 20, Width: 15, Height: 10
 (x1,y1)=(20,20), (x2,y2)=(35,20), (x3,y3)=(35,30), (x4,y4)=(20,30)
[Square][Rectangle] x: 30, y: 30, Width: 15, Height: 15
 (x1,y1)=(30,30), (x2,y2)=(45,30), (x3,y3)=(45,45), (x4,y4)=(30,45)
[Square][Rectangle] x: 30, y: 30, Width: 15, Height: 15
 (x1,y1)=(30,30), (x2,y2)=(45,30), (x3,y3)=(45,45), (x4,y4)=(30,45)
Draw Quadrangle:x: 10, y: 10, Width: 10, Height: 15
 (x1,y1)=(15,10), (x2,y2)=(20,20), (x3,y3)=(15,25), (x4,y4)=(10,15)
Draw Rectangle:[Rectangle] x: 20, y: 20, Width: 15, Height: 10
 (x1,y1)=(20,20), (x2,y2)=(35,20), (x3,y3)=(35,30), (x4,y4)=(20,30)
Draw Square:[Square][Rectangle] x: 30, y: 30, Width: 15, Height: 15
 (x1,y1)=(30,30), (x2,y2)=(45,30), (x3,y3)=(45,45), (x4,y4)=(30,45)
Draw Square:[Square][Rectangle] x: 30, y: 30, Width: 15, Height: 15
 (x1,y1)=(30,30), (x2,y2)=(45,30), (x3,y3)=(45,45), (x4,y4)=(30,45)
```