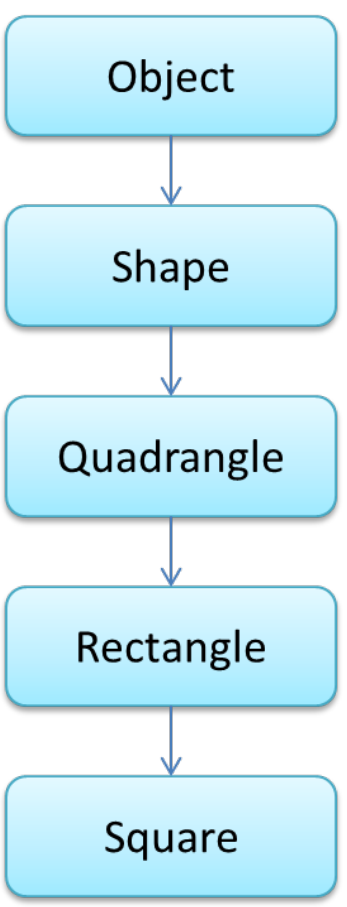
Introduction to Java

Special Care Set for Java Begineers/ Ex 06

・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: クラスの設計と継承 - Shapeを司るクラス群 -
Class Design and Inheritance - Shape related class group -



前回はクラスの再利用に対して、継承をうまく使っていくことで、同じような動作をするクラスを簡単に作ることを学びました。
今回は、オブジェクト指向の根幹でもある、『クラスの設計』について考えて見たいと思います。

1.『形』をどのように扱うか？

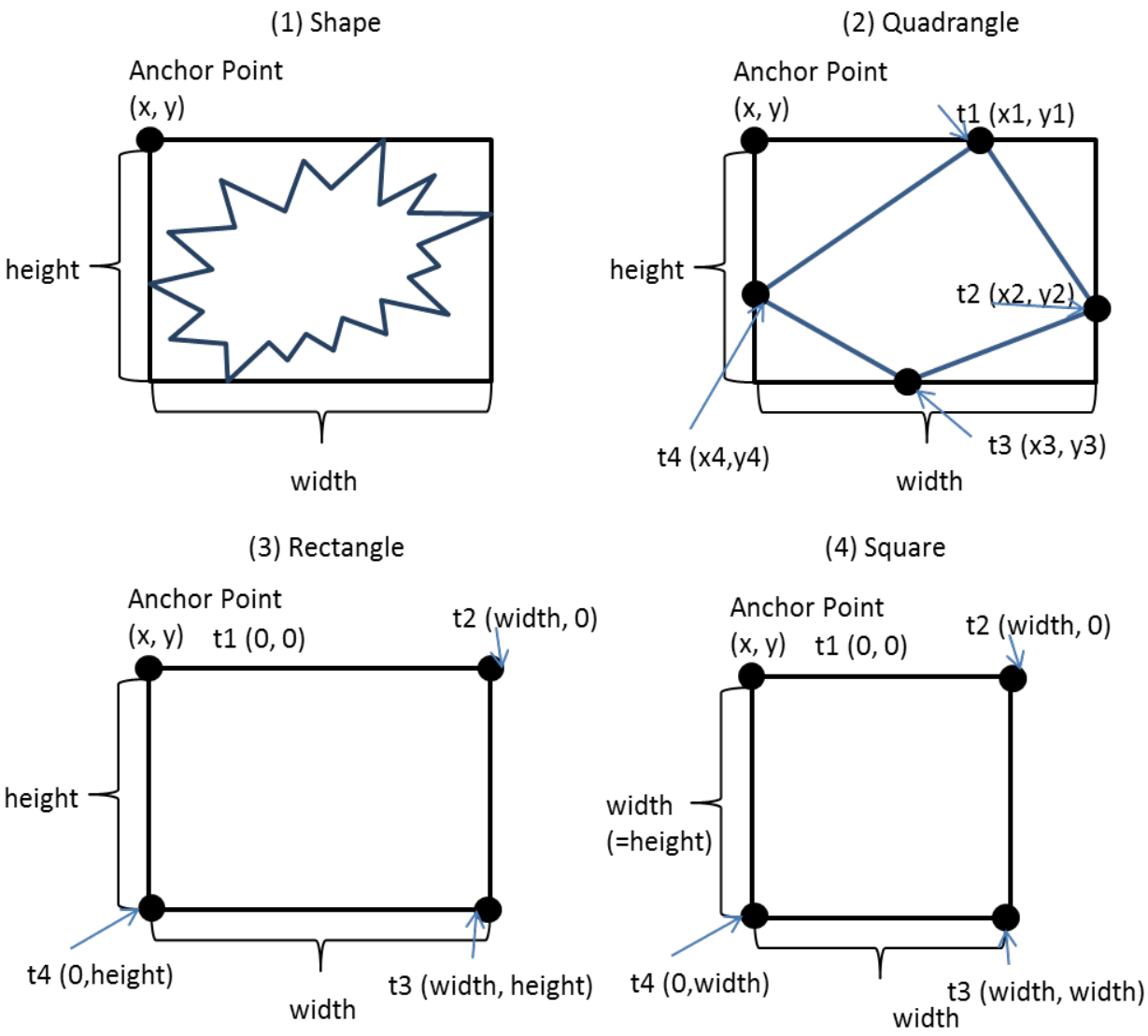
2次元の絵を書こうとするとき、様々な形を書くことになります。星型や菱形、円や楕円、
様々な形を組み合わせて、絵になることを考えましょう。
形を書く時、基本になるのが『点』です。2次元なので、キャンバス上にxとyの2つの要素に対して、値が入ることになるでしょう。
点と点を結ぶと『直線』になります。また、点と点の間に『曲率』を含めると『曲線』になるでしょう。
そして、3点を直線で結べば『三角形』、4点を直線で結べば『四角形』になります。
三角形や四角形は特殊な形があります。『直角三角形』、『正三角形』などの、各頂点に特殊な角度を持つ三角形、
『長方形』や『正方形』、『平行四辺形』、『台形』、『ひし型』などの特殊な関係を持つ四角形などがあります。
全部を細かく設計する必要はありませんが、作りたいものに応じて、クラスをうまく継承することが出来れば、非常に簡単にいろいろなオブジェクトを設計できます。

点や線はあまりに一般的過ぎるので、ここでは『四角形』と『長方形』、『正方形』について考えてみましょう。
四角形（Quadlangle）は、4つの点、4つの辺によって作られる図形です。4つの点は、必ず2つの辺に接しています。四角形では、辺と辺の間の角は不定ですが、原則として全て足すと360度となることが言われています。
この角が全て90度となる時、これを『長方形』(Rectangle)と呼びます。すべての角が90度なので、向かい合う2つの辺は平行です。しかしながら、向かい合う辺同志を対として、2対の辺の長さは自由です。
2対の辺の長さが同じ、すなわち、全ての辺の長さが同じ長方形を『正方形』(Square)と呼びます。
この様に、四角形→長方形→正方形の順に、厳密に定義され、また、形⊇四角形⊇長方形⊇正方形のように、包含関係（Inclusive relation)があります。

Javaのクラス設計は、包含関係があることを用いて行われます。この例では、右図のような形で定義していくようにしましょう。

2. 上位クラスと下位クラスの関係

次に、上位クラスと下位クラスの関係について考えてみましょう。
ここでは、『形』を定義していますから、形すべてに関して共通するものを考えましょう。
キャンバスに形を書き込む場合は、各形の『アンカーポイント』が必要になります。その形の絶対位置を示すものです。
また、各形には『幅』と『高さ』が必要になります。
全ての形は、『アンカーポイント』『幅』『高さ』によって囲われる枠の中に書き込まれるようになります。(右図2-1)



さて、四角形はどうでしょうか？四角形で必要なのは4つの点ですから、アンカーポイントとアンカーポイントからの4つの相対座標があればいいことになります。(右図2-2)

四角形では、所持する点が自由なので、アンカーポイントと4つの点は相対的に扱いましたが、長方形は『角度が変化しない』限り、必ずアンカーポイントに1つの角が来て、幅と高さによってほかの頂点が決定することになります。つまり、長方形に必要な値は『アンカーポイント』『幅』『高さ』となります。（右図2-3)

最後に、正方形ですが、正方形は『幅』＝『高さ』になりますね。なので、正方形に必要な値は『アンカーポイント』『幅』のみで決定できます。（右図2-4)

さて、それぞれの要素を確認してみましょう。
- Shape: x, y, width, height
- Quadrangle: x, y, x1, y1, x2, y2, x3, y3, x4, y4, (width, height)
- Rectangle: x, y, width, height, (x1, y1, x2, y2, x3, y3, x4, y4)
- Square: x, y, width, (height, x1, y1, x2, y2, x3, y3, x4, y4)

括弧に含まれない値が、直接指定してその形を作成するのに必要な値ですが、括弧に含まれる値を追加すると、上位の形としても同様に定義できることが分かります。
このように、Classの継承を用いて定義する場合、上位の値との整合性を図ることで、上位にある形として同様に扱うことが出来るというわけです。

3. クラスの実装

それでは、実際にこの4つのクラスを実装したソースコードを見てみましょう。
なお、この4つの関係はこればかりではなく、また、これよりも良い設計の仕方、構築の仕方があると思います。
このソースコードは一例ですので、是非皆さんでも考えてみてください。

```
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();
    }

}
```

実行すると次のような出力になります。

```
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)
```

いかがでしょうか？
Shapeクラスではかなりの行数を書いていましたが、最後のSquareクラスではかなり簡単に書くことが出来ました。
是非、他の形状（三角形、五角形、平行四辺形、ひし形、台形）や、他の要素（対角線、外周長、面積）なども実装してみてください。
また、この上位クラスには当てはまりませんが、PointクラスやLineクラスなどもあると、より分かりやすく、より便利になると思います。