Special Care Set for Java Begineers/ Ex 08

・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: 抽象的なクラス - 実装はあと回し -
Abstract Class - Impremention is later -

クラスを設計していると、『どうしても今厳密に定義出来ない』ようなメソッドが存在したりします。
例えば、大学の授業の採点基準は科目や教員によって違いますから、もし科目全体に統一した型を使って各科目を作成したとしても、点数計算の部分は各科目内でしか設定できません等というようなことになります。

今回は、オブジェクト指向で重要な『抽象化』について勉強します。
抽象化とは、ざっくり言えば『細かいことは抜きにして』という概念です。
どうしてもプログラミングする際には、細かい所をしっかりと書いていかなければいけませんが、ことソフトウエアを設計する段階においては、細かい所にばかり手を煩わせていると、大きい枠で全体を統括するようなシステムに手を付けられなくなってしまいます。
そのため、Javaでは『abstract』というキーワードを用いてクラスを定義し、『継承した後で厳密に定義してください』と次の人にお仕事を投げることが出来る方法を使うことが出来ます。

1. “抽象的な” Shape クラス

前回のShapeクラスとその派生クラスの設計では、Shapeクラスに様々な機能が入っていました。
しかしながら、Shapeクラスはどんな図形を定義しているのか分からないため、『面積』を計算することが出来ません。
他の図形では、例えばRectangleはその図形の面積は左上点と右下点が分かっていれば面積を計算することが出来ます。
こういった風に、『上位クラスでは意味は持っていても実装が出来ない』時に抽象クラスとして扱うことが出来ます。

下のコードは、以前のコードに面積を計算する getArea() メソッドを 追加したものです。
しかし、Shapeクラス中では実装出来ないので、Shapeクラスはabstractになっています。
Quadrangleクラス中では頑張れば出来るかもしれませんが、実装していないのでやはりabstractとなっています。
実際に getArea() メソッドを追加したのはRectangleクラス中です。

```
public class ShapeTest {

        /**
         * @param args
         */
        public static void main(String[] args) {
```

```
                // TODO Auto-generated method stub
                Object obj3 = new Rectangle(20,20,15,10);
                Object obj4 = new Square(30,30,15);

                print(obj3.toString());
                print(obj4.toString());
        }

        public static void print(Object obj){
                System.out.println(obj.toString());
        }

}
interface Drawable{
        void draw();
}
/* Shapeクラス */
abstract 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;
    }

    abstract int getArea();
}

/* Quadrangleクラス */
abstract 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] area: " + getArea() + ", "+ super.toString();
    }

    @Override
    public void draw() {
            // TODO Auto-generated method stub
```

```
                System.out.println("Draw Rectangle:" + toString());
        }

        @Override
        int getArea() {
                // TODO Auto-generated method stub
                return width * height;
        }

}

/* 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());
        }

}
```

出力結果は次の通りになります。

```
[Rectangle] area: 150, 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] area: 225, x: 30, y: 30, Width: 15, Height: 15
 (x1,y1)=(30,30), (x2,y2)=(45,30), (x3,y3)=(45,45), (x4,y4)=(30,45)
```

分からない所を『分からない！』と適当に過ごしてしまう、割と些細な技術なabstractですが、
使い方によっては非常に強力なツールとなるので、みなさんもしっかり勉強してください。