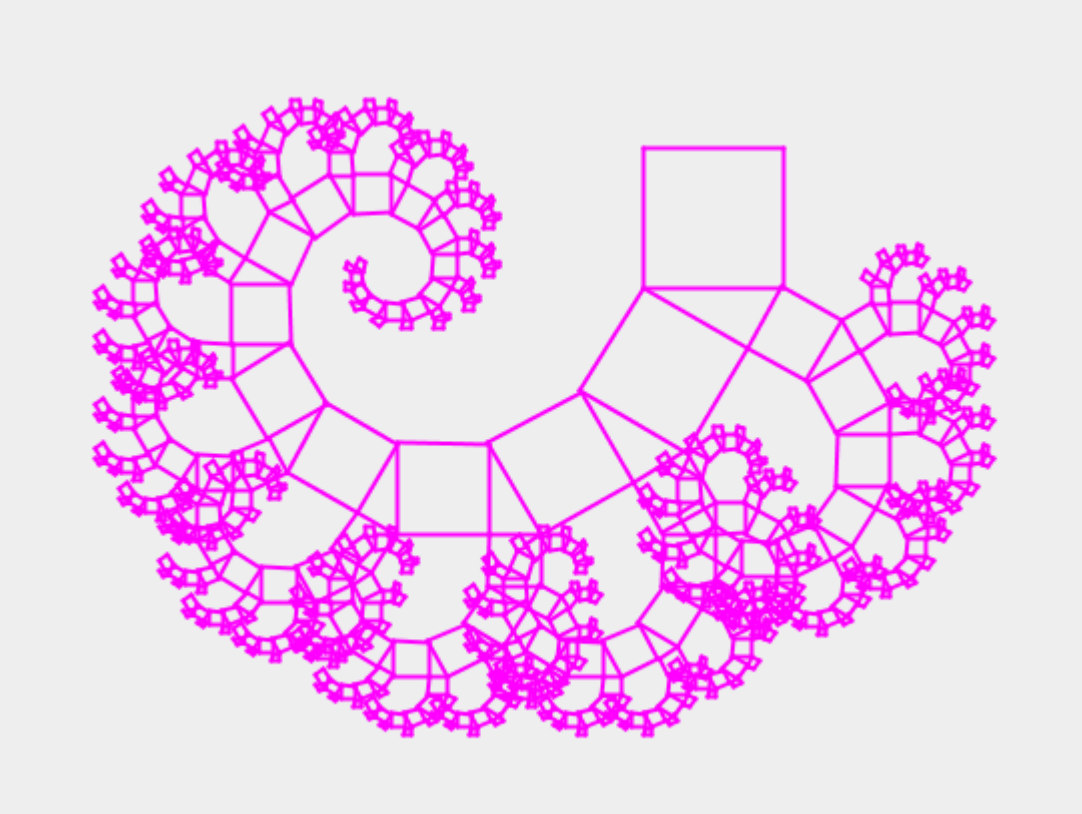
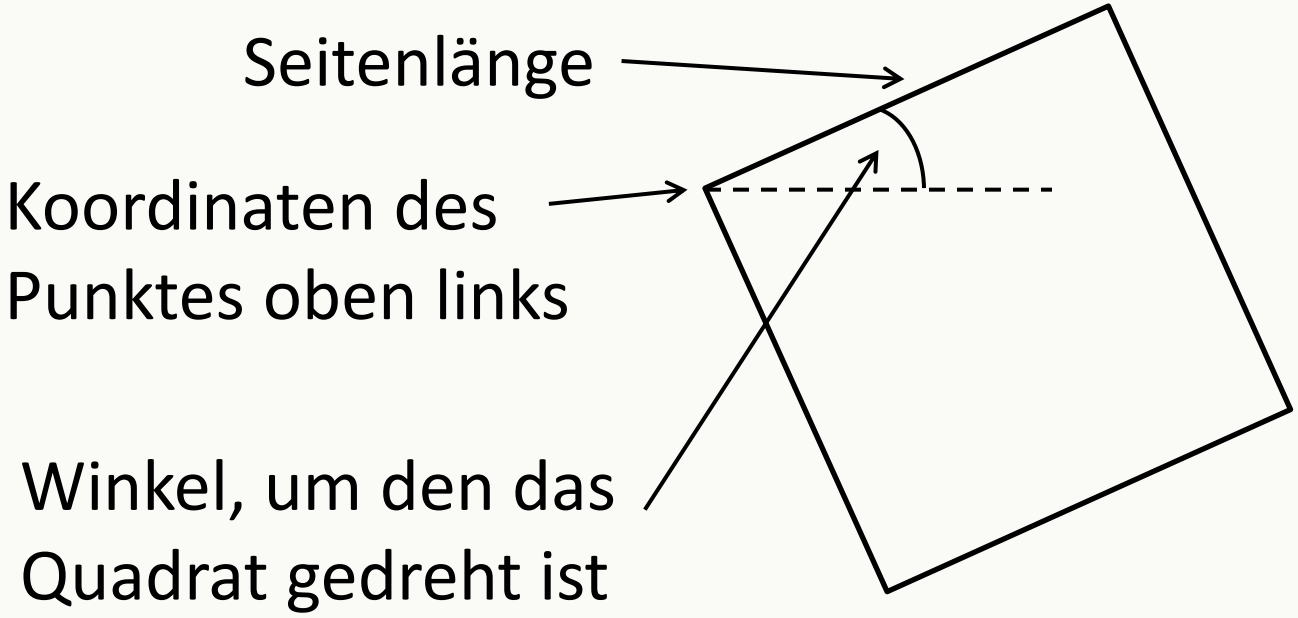
S.61/6 Die Bäume des Pythagoras

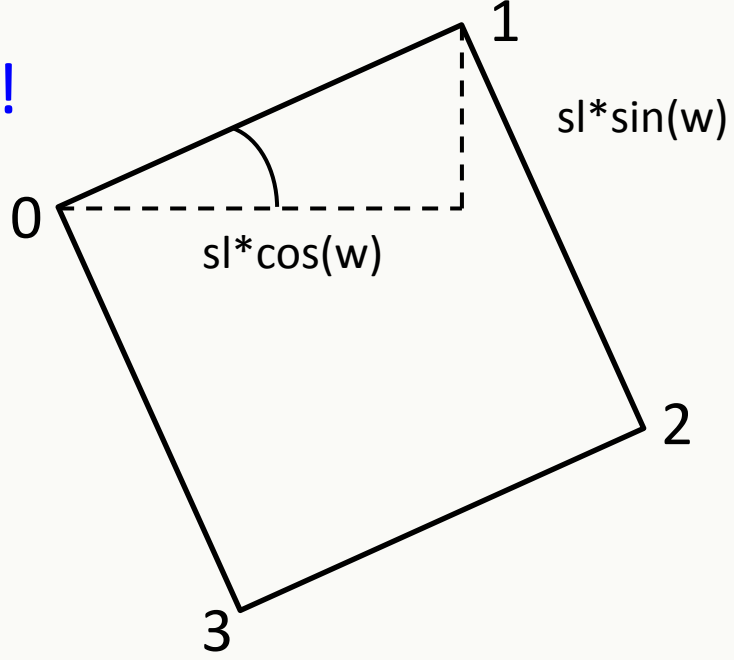


Hierbei ist es notwendig, (unausgefüllte) Quadrate mit beliebiger Seitenlänge, die um einen bestimmten Winkel w gegenüber der x-Achse gedreht sind, zeichnen zu können.



Öffne S61-A6-Vorlage und vervollständige die Klasse Quadratrahmen.

Ursprung ist links oben!

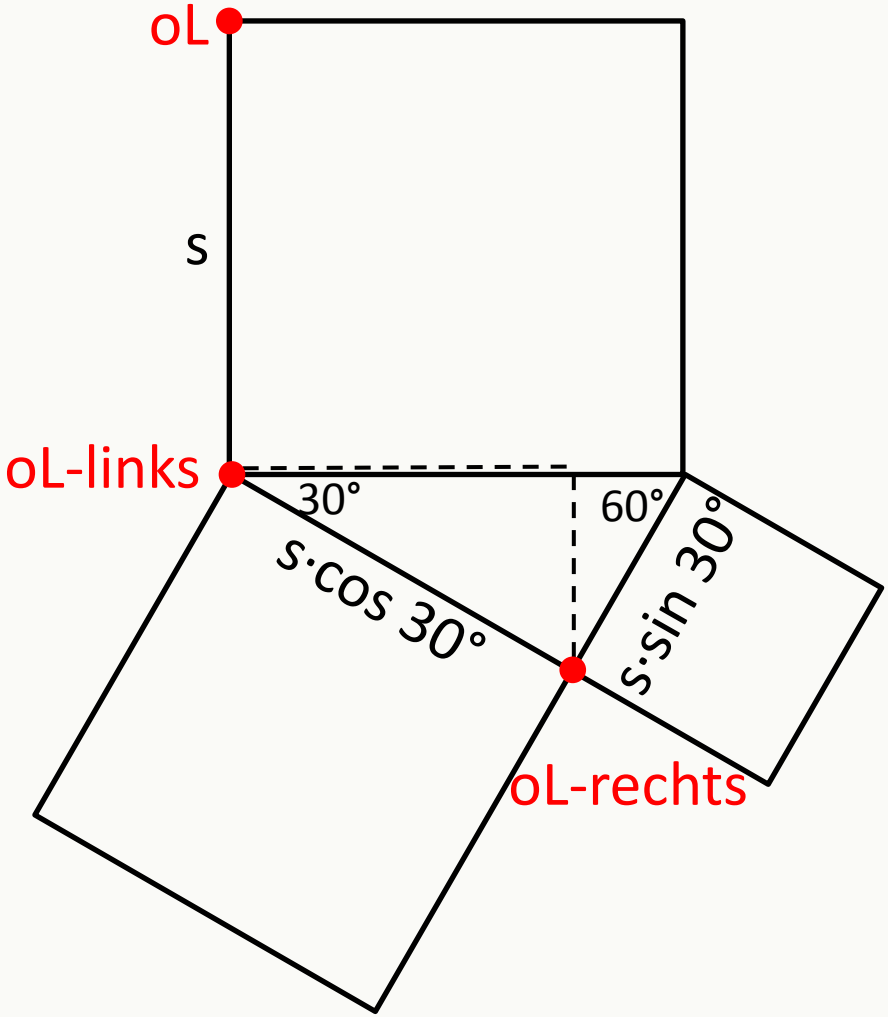



```
double sin_w_b = Math.sin(winkel_b);
double cos_w_b = Math.cos(winkel_b);
x[1] = (int) (x[0] + seitenlaenge * cos_w_b);
y[1] = (int) (y[0] - seitenlaenge * sin_w_b);
x[2] = (int) (x[1] + seitenlaenge * sin_w_b);
y[2] = (int) (y[1] + seitenlaenge * cos_w_b);
x[3] = (int) (x[2] - seitenlaenge * cos_w_b);
y[3] = (int) (y[2] + seitenlaenge * sin_w_b);
```

```
for (int i=0; i<4;i++){
        quadrat[i] = new Linie(x[i], y[i], x[(i+1)%4], y[(i+1)%4]);
        quadrat[i].setzeFarbe("magenta");
        quadrat[i].setzeLinienDicke(2);
    }
```

Der Konstruktor der Klasse Datenknoten ist hier rekursiv.
Mach dir den Ablauf klar, ergänze entsprechende Kommentare und veranschauliche es anhand einer Skizze in deinem Heft.
Vervollständige den else-Teil.
Setze dann in der Klasse Baum die Konstruktordaten passend.

```
public Datenknoten(int x_o_l, int y_o_l, double sl, double w){
    if (sl>5){
        inhalt = new Quadratrahmen(x_o_l, y_o_l, sl, w);
        double alpha =30.0;
        double alpha_b = alpha/180.0*Math.PI;
        double w_b = w/180.0*Math.PI;
        double laenge_g = sl*Math.cos(alpha_b);
        double laenge_k = sl*Math.sin(alpha_b);
        double x_links = x_o_l+sl*Math.sin(w/180.0*Math.PI);
        double y_links = y_o_l+sl*Math.cos(w/180.0*Math.PI);
        naechsterLinks = new Datenknoten((int) (x_links),(int) (y_links),
                                                    laenge_g, w-alpha);
        naechsterRechts = new Datenknoten
                        ((int) (x_links+laenge_g*Math.cos(alpha_b-w_b)),
                         (int) (y_links+laenge_g*Math.sin(alpha_b-w_b)),
                                laenge_k, w+90-alpha);
    }
```


oL
s
oL-links
30°
60°
s·cos 30°
s·sin 30°
oL-rechts

```
else {
        naechsterLinks = new Abschluss();
        naechsterRechts = new Abschluss();
        inhalt = new Quadratrahmen(x_o_l, y_o_l, sl, w);
        }
   }
```

Wann stoppt die Rekursion?
Welche andere Möglichkeit gäbe es?

```
        public Baum(){
              wurzel = new Datenknoten(350, 100, 70, 0);
           }
```

Teste das Programm. Werden die Quadrate preorder, inorder oder postorder gezeichnet? Teste jeweils die anderen Möglichkeiten.

Implementiere eine Methode, die alle Knoten zählt!

In Baum:

```
public int knotenZaehlen(){
    return wurzel.knotenZaehlen();
}
```

In Baumelement:

```
public abstract int knotenZaehlen();
```

In Datenknoten:

```
public int knotenZaehlen(){
    return 1+naechsterLinks.knotenZaehlen()
           +naechsterRechts.knotenZaehlen();
}
```

In Abschluss:

```
public int knotenZaehlen(){
    return 0;
}
```

Implementiere eine Methode, die den Flächeninhalt aller Quadrate berechnet!

In Baum:
```
public double flaecheBerechnen(){
    return wurzel.flaecheBerechnen();
}
```

In Baumelement:
```
public abstract double flaecheBerechnen();
```

In Datenknoten:
```
public double flaecheBerechnen(){
    return inhalt.seitenlaengeGeben()*inhalt.seitenlaengeGeben()
        +naechsterLinks.flaecheBerechnen()
        +naechsterRechts.flaecheBerechnen();
}
```

In Abschluss:
```
public double flaecheBerechnen(){
    return 0.0;
}
```