
Aufgabe 21: Objektorientierung

Zur Demonstration der Wiederverwendbarkeit und Erweiterbarkeit objektorientierter Programme soll im Rahmen dieser Übung dasselbe Programm sowohl nicht-objektorientiert (Programmiersprache C) als auch objektorientiert (Programmiersprache Java) gelöst werden:

a) Schreiben Sie in C ein Modul zum Zeichnen von *Dreiecken, Quadraten* und *Rechtecken.* Das Modul soll einen geeigneten Datentyp `Drawable` sowie Funktionen zum Erzeugen von Dreiecken, Quadraten und Rechtecken und ausserdem eine Funktion `draw ( Drawable )` zur Verfügung stellen.

   Gehen Sie davon aus, dass es eine globale Funktion `drawLine ( float x1, float y2, float x2, float y2)` gibt zum Zeichnen einer Linie zwischen den Punkten $(x_1, y_1)^{\mathrm{T}}$ und $(x_2, y_2)^{\mathrm{T}}$ gibt.

b) Lösen Sie dasselbe Problem in Java, indem Sie eine Klasse `Drawable` und entsprechende Unterklassen `Dreieck`, `Quadrat` und `Rechteck` definieren. `Drawable` soll eine Methode `void draw()` zum Zeichnen eines Zeichenobjektes zur Verfügung stellen.

   Gehen Sie auch hier von einer (global verwendbaren) Methode `Graphics.drawLine(float x1, float y1, float x2, float y2)` zum Zeichnen einer Linie zwischen den Punkten $(x_1, y_1)^{\mathrm{T}}$ und $(x_2, y_2)^{\mathrm{T}}$ aus.

c) Ergänzen Sie nun Ihre beiden Lösungen jeweils um das Zeichnen einfacher *Linien.* Welche Lösung ist einfacher zu erweitern? Welche Probleme ergeben sich, wenn Sie zur Lösung dieser Teilaufgabe keinen Zugriff auf den Quellcode der beiden vorherigen Teilaufgaben haben?

Lösung:

a) Lösung in C: Dateien `drawable.h, drawable.c`

```
#include <stdio.h>

typedef struct Koord Koord;
struct Koord {
  float x;
  float y;
  Koord *nextpoint;
};

typedef struct Drawable Drawable;

struct Drawable {
  enum { Triangle, Square, Rectangle } shape_kind;
  Koord origin;
};

void draw( Drawable *drw );
void drawLine( float x1, float y1, float x2, float y2 );

Drawable *new_Dreieck(float ox, float oy, float px, \
                      float py, float qx, float qy);
Drawable *new_Quadrat(float ox, float oy, float sl);
Drawable * new_Rechteck(float ox, float oy, float sl1, float sl2);
```

```
#include "drawable.h"

Drawable *new_Dreieck(float ox, float oy, float px, float py, \
                      float qx, float qy) {
  Drawable *n_tria;
  Koord *nextkoord;
  if((n_tria = (Drawable *)malloc(sizeof(struct Drawable))) == NULL)
    fprintf(stderr, "new_Dreieck: Out of memory!\n");
  n_tria->shape_kind = Triangle;
  n_tria->origin.x = ox; /* Ursprungskoordinatenwerte eintragen */
  n_tria->origin.y = oy;

  if ((nextkoord = (Koord *)malloc(sizeof(struct Koord))) == NULL)
    fprintf(stderr, "Out of memory!\n");
  nextkoord->x = px; /* 2. Eckpunkt */
  nextkoord->y = py;
  n_tria->origin.nextpoint = nextkoord; /* einhaengen */
```

```
  if ((nextkoord = (Koord *)malloc(sizeof(struct Koord))) == NULL)
    fprintf(stderr, "Out of memory!\n");
  nextkoord->x = qx; /* 3. Eckpunkt */
  nextkoord->y = qy;
  nextkoord->nextpoint = NULL;
  n_tria->origin.nextpoint->nextpoint=nextkoord; /* einhaengen */
  return n_tria;
}

Drawable *new_Quadrat(float ox, float oy, float sl) {
  Drawable *n_squa;
  Koord *nextkoord;
  if((n_squa = (Drawable *)malloc(sizeof(struct Drawable))) == NULL) {
    fprintf(stderr, "new_Quadrat: Out of memory!\n"); exit(1); }
  n_squa->shape_kind = Square;
  n_squa->origin.x = ox; /* Ursprungskoordinatenwerte eintragen */
  n_squa->origin.y = oy;

  if ((nextkoord = (Koord *)malloc(sizeof(struct Koord))) == NULL) {
    fprintf(stderr, "Out of memory!\n"); exit(1); }
  nextkoord->x = sl; /* Seitenlaenge speichern */
  nextkoord->y = 0.0;
  nextkoord->nextpoint = NULL;
  return n_squa;
}

Drawable *new_Rechteck(float ox, float oy, float sl1, float sl2) {
  Drawable *n_rect;
  Koord *nextkoord;
  if((n_rect = (Drawable *)malloc(sizeof(struct Drawable))) == NULL) {
    fprintf(stderr, "new_Rechteck: Out of memory!\n"); exit(1); }
  n_rect->shape_kind = Rectangle;
  n_rect->origin.x = ox; /* Ursprungskoordinatenwerte eintragen */
  n_rect->origin.y = oy;

  if ((nextkoord = (Koord *)malloc(sizeof(struct Koord))) == NULL) {
    fprintf(stderr, "Out of memory!\n"); exit(1); }
  nextkoord->x = sl1; /* Seitenlaenge 1 speichern */
  nextkoord->y = sl2; /* Seitenlaenge 2 speichern */
  nextkoord->nextpoint = NULL;
  return n_rect;
}
```

```
void draw(Drawable *drw) {
  switch (drw->shape_kind) {
    case 0:
      printf("zeichne Dreieck: \n");
      drawLine(drw->origin.x,drw->origin.y, \
               drw->origin.nextpoint->x,drw->origin.nextpoint->y);
      drawLine(drw->origin.nextpoint->x,drw->origin.nextpoint->y,
               drw->origin.nextpoint->nextpoint->x, \
       drw->origin.nextpoint->nextpoint->y);
      drawLine(drw->origin.nextpoint->nextpoint->x, \
       drw->origin.nextpoint->nextpoint->y, \
       drw->origin.x,drw->origin.y);
      break;
    case 1:
      printf("zeichne Quadrat: \n");
      drawLine(drw->origin.x,drw->origin.y, \
               drw->origin.x+drw->origin.nextpoint->x,drw->origin.y);
      drawLine(drw->origin.x+drw->origin.nextpoint->x,drw->origin.y, \
               drw->origin.x+drw->origin.nextpoint->x, \
       drw->origin.y+drw->origin.nextpoint->x);
      drawLine(drw->origin.x+drw->origin.nextpoint->x, \
               drw->origin.y+drw->origin.nextpoint->x, \
               drw->origin.x,drw->origin.y+drw->origin.nextpoint->x);
      drawLine(drw->origin.x,drw->origin.y+drw->origin.nextpoint->x, \
               drw->origin.x,drw->origin.y);
      break;
    case 2:
      printf("zeichne Rechteck: \n");
      drawLine(drw->origin.x,drw->origin.y, \
               drw->origin.x+drw->origin.nextpoint->x,drw->origin.y);
      drawLine(drw->origin.x+drw->origin.nextpoint->x,drw->origin.y, \
               drw->origin.x+drw->origin.nextpoint->x, \
       drw->origin.y+drw->origin.nextpoint->y);
      drawLine(drw->origin.x+drw->origin.nextpoint->x, \
               drw->origin.y+drw->origin.nextpoint->y, \
               drw->origin.x,drw->origin.y+drw->origin.nextpoint->y);
      drawLine(drw->origin.x,drw->origin.y+drw->origin.nextpoint->y, \
               drw->origin.x,drw->origin.y);
      break;
    default:
      printf("Unbekannte Figur!!!\n");
  }

}
```

```
void drawLine( float x1, float y1, float x2, float y2) {
  printf("ziehe Linie von (%f,%f) nach (%f,%f)\n", x1, y1, x2, y2);
}

int main(int argc, char* argv[]) {
  Drawable *figure;
  printf("Start:\n");
  figure = new_Dreieck(1.0,1.0,1.0,2.0,2.0,2.0);
  draw(figure);
  free(figure);
  figure = new_Quadrat(1.0,1.0,2.0);
  draw(figure);
  free(figure);
  figure = new_Rechteck(1.0,1.0,3.0,2.0);
  draw(figure);
  free(figure);

  exit(0);
}
```

Start des Programms zum Testen des `drawable`-Moduls:

```
faui45e> ./drawable
Start:
zeichne Dreieck:
ziehe Linie von (1.000000,1.000000) nach (1.000000,2.000000)
ziehe Linie von (1.000000,2.000000) nach (2.000000,2.000000)
ziehe Linie von (2.000000,2.000000) nach (1.000000,1.000000)
zeichne Quadrat:
ziehe Linie von (1.000000,1.000000) nach (2.000000,1.000000)
ziehe Linie von (2.000000,1.000000) nach (2.000000,2.000000)
ziehe Linie von (2.000000,2.000000) nach (1.000000,2.000000)
ziehe Linie von (1.000000,2.000000) nach (1.000000,1.000000)
zeichne Rechteck:
ziehe Linie von (1.000000,1.000000) nach (2.000000,1.000000)
ziehe Linie von (2.000000,1.000000) nach (2.000000,3.000000)
ziehe Linie von (2.000000,3.000000) nach (1.000000,3.000000)
ziehe Linie von (1.000000,3.000000) nach (1.000000,1.000000)
```

b) Lösung in `Java`: Dateien `Drawable.java, Dreieck.java, Quadrat.java, Rechteck.java` Jede Klasse, sowohl die Oberklasse `Drawable` als auch die Unterklassen `Dreieck, Quadrat, Rechteck` ist in einer separaten Quellcode-Datei.

```
public class Drawable {
  public float ox, oy;

  public Drawable (float x, float y) {
    this.ox = x;
    this.oy = y;
  }

  public void draw() {
    Graphics.drawLine(this.ox, this.oy, 0.0, 0.0);
  }
}
```

```
public class Dreieck extends Drawable {
  public float px, py, qx, qy;

  public Dreieck(float x1, float y1,
                float x2, float y2,
float x3, float y3) {
    super(x1,y1);
    this.px = x2;
    this.py = y2;
    this.qx = x3;
    this.qy = y3;
  }

  public void draw() {
    Graphics.drawLine(this.ox,this.oy,this.px,this.py);
    Graphics.drawLine(this.px,this.py,this.qx,this.qy);
    Graphics.drawLine(this.qx,this.qy,this.ox,this.oy);
  }
}
```

```
public class Quadrat extends Drawable {
  public float sideLength;

  public Quadrat(float x1, float y1, float sl) {
    super(x1,y1);
    this.sideLength = sl;
  }
```

```
  public void draw() {
    Graphics.drawLine(this.ox,this.oy,this.ox+this.sideLength,this.oy);
    Graphics.drawLine(this.ox+this.sideLength,this.oy,this.ox+this.sideLength, \
                      this.oy+this.sideLength);
    Graphics.drawLine(this.ox+this.sideLength,this.oy+this.sideLength,this.ox, \
                      this.oy+this.sideLength);
    Graphics.drawLine(this.ox,this.oy+this.sideLength,this.ox,this.oy);
  }
}
```

```
public class Rechteck extends Drawable {
  public float sideLength1;
  public float sideLength2;

  public Rechteck(float x1, float y1,
                  float sl1, float sl2) {
    super(x1,y1);
    this.sideLength1 = sl1;
    this.sideLength2 = sl2;
  }

  public void draw() {
    Graphics.drawLine(this.ox,this.oy,this.ox+this.sideLength1,this.oy);
    Graphics.drawLine(this.ox+this.sideLength1,this.oy, \
                      this.ox+this.sideLength1,this.oy+this.sideLength2);
    Graphics.drawLine(this.ox+this.sideLength1,this.oy+this.sideLength2, \
                      this.ox,this.oy+this.sideLength2);
    Graphics.drawLine(this.ox,this.oy+this.sideLength2,this.ox,this.oy);
  }
}
```

c) Ergänzungen:

In der `C`-Lösung müssen in der Quellcode-Datei `drawable.c` sowie in der Header-Datei drawable.h Erweiterungen für das neue Bildelement `Linie` eingetragen werden:

```
In drawable.h

 ...

struct Drawable {
  enum { Triangle, Square, Rectangle, Line } shape_kind;
  Koord origin;
};

 ...

 Drawable * new_Linie(float ox, float oy, float x2, float y2);
```

```
In drawable.c

 ...

Drawable *new_Linie(float ox, float oy, float x1, float y2) {
   Drawable *n_line;
   Koord *nextkoord;
   if((n_line = (Drawable *)malloc(sizeof(struct Drawable))) == NULL) {
       fprintf(stderr, "new_Linie: Out of memory!\n"); exit(1); }
   n_line->shape_kind = Linie;
   n_line->origin.x = ox; /* Ursprungskoordinatenwerte eintragen */
   n_line->origin.y = oy;
   if ((nextkoord = (Koord *)malloc(sizeof(struct Koord))) == NULL) {
       fprintf(stderr, "Out of memory!\n"); exit(1); }
   nextkoord->x = x2; /* Koordinate Linienendpunkt speichern */
   nextkoord->y = y2;
   nextkoord->nextpoint = NULL;
   return n_line;
}
```

Ausserdem muss die Funktion `draw( Drawable *drw )` angepasst werden (ein weiteres `case` in dem `switch`-Statement mit dem Aufruf von `drawLine()` zum Zeichnen der Linie.

In `Java`: Es wird eine weitere Unterklasse `Linie` von der Oberklasse `Drawable` abgeleitet. Der Quellcode steht in einer eigenen Datei `Linie.java`, der restliche Quellcode muss nicht angepasst werden!

```
public class Linie extends Drawable {
  public float x2, y2;

  public Quadrat(float x1, float y1, float x2, float y2) {
    super(x1,y1);
    this.x2 = x2;
    this.y2 = y2;
  }

  public void draw() {
    Graphics.drawLine(this.ox,this.oy,this.x2,this.y2);
  }
}
```