**Informatik mit Java -- Komplettlösung Band I**

**Inhalt**

Danksagung

Keins meiner Lieblingsbücher und möchte man mit java anfangen sollte man vielleicht erstmal lernen mit den standartbibliotheken klarzukommen,da hilft der Deutsch/Englisch misch auch nichts sondern frustriert eher. Ich mag es einfach nicht mit sum.\*; zu arbeiten.

Vorwort an Schüler

Verständlich das ihr hier abgucken wollt das Buch kann sehr zäh sein und ich kann verstehen wenn man ihrgendwann frustriert kein Bock mehr hat.

Informatik ist nicht immer so langweilig, vielleicht gefällt euch ja datenanalyse oder

Gamedesign. Wenn ihr informatik haben müsst ,da ihr falsch gewählt habt oder so tut mir das sehr leid ,aber kopfhoch ihr schafft das!! :D

Vorwort an Lehrer

Wenn sie entscheidungsfreiheit haben bringen sie den Schülern bitte nicht mit diesm Buch informatik nahe.

Den nicht Informatik interssierten wird dieses Buch schnell garnicht mehr helfen und den Informatik intressierten wird es eher abwenden da die projekte eher langweilig sind und man nicht viel über Standart Java lernt.

**Kurze Erklärung**

Kapitel ohne Aufgaben werden übersprungen.

ROT markierte Kapitel und oder Aufgaben werden zu einem späteren Zeitpunkt realisiert.

**Kapitel 2 Klassen und Objekte**

**2.3 Dienste aufrufen**

Übung 2.1

Die Hintergrundfarbe des erzeugten Bildschirms färbt sich rot.

Übung 2.2

0 Schwarz

1 Dunkelblau

2 Türkis

3 Dunkelgrau

4 Hellgrau

5 Hellgrün

6 WeißGrau

7 Pink

8 Senfgelb

9 Beige

10 Rot

11 Weiß

12 Gelb

Übung 2.3

Ja,wenn man die größe des Bildschirms anpasst passt sich die größe des Bildschirms an.

**2.5 Anfragen und Aufträge**

Übung 2.4

Camel case ist eine Schreibweise, bei der mehrere Wörter zusammen geschrieben werden und dabei der erste Buchstabe jedes Wortes außer dem ersten groß geschrieben wird. Zum Beispiel: "MeinNameIstBernard". Diese Schreibweise wird häufig in der Informatik verwendet, um Variablen- oder Funktionsnamen lesbarer zu machen und gleichzeitig den Code kompakter zu gestalten.

Übung 2.5

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

s = new Stift();

this.fuehreAus();

}

>-Übung 2.6

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

//Dreieck

s.bewegeBis(100, 100);

s.runter();

s.bewegeUm(100);

s.dreheUm(120);

s.bewegeUm(100);

s.dreheUm(120);

s.bewegeUm(100);

s.hoch();

}

}

>-Übung 2.7

In unsere fuehreAus() Funktion:

//Rechteck

s.bewegeBis(200, 200);

s.runter();

s.zeichneRechteck(200, 200);

s.dreheBis(315);

s.bewegeUm(280);

s.bewegeUm(-280);

s.dreheBis(0);

s.bewegeUm(200);

s.dreheBis(225);

s.bewegeUm(280);

s.hoch();

**Kapitel 3 Aufbau eines SuM-Programms**

**3.1 Erzeugen einer Programm-Vorlage**

Die Programm-Vorlage in Question:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

s.bewegeBis(100, 100);

s.schreibeText("Hallo Welt");

s.gibFrei();

b.gibFrei();

}

}

**3.2 Programmaufbau**

In diesem Kapitel steht man solle nicht an kommentaren sparen. Bitte folgt dieser Anweisung nicht und schreibt lieber eine richtige Dokumentation :> lange Kommentare inmitten des Codes nerven nur alle die diesen editieren müssen.

Wenn ihr allerdings alleine Arbeitet schreibt ruhig Bücher in euren code :)

**3.4 Fehlersuche**

Keine Aufgaben bei denen ich helfen könnte alles im Editor.

**3.5 Programmgesteuertes Zeichnen**

Übung 3.5

public void fuehreAus()

{

s.bewegeBis(100, 10);

//Header

s.schreibeText("Zeichnung gestalted von Willi Wusel");

//Dreieck

s.bewegeBis(10, 100);

s.runter();

s.bewegeUm(100);

s.dreheUm(120);

s.bewegeUm(100);

s.dreheUm(120);

s.bewegeUm(100);

s.hoch();

//Rechteck

s.bewegeBis(120, 11);

s.runter();

s.zeichneRechteck(100, 100);

s.hoch();

//Sechseck

s.bewegeBis(280, 100);

s.runter();

s.dreheBis(0);

s.bewegeUm(50);

s.dreheBis(60);

s.bewegeUm(50);

s.dreheBis(120);

s.bewegeUm(50);

s.dreheBis(180);

s.bewegeUm(50);

s.dreheBis(240);

s.bewegeUm(50);

s.dreheBis(300);

s.bewegeUm(50);

s.hoch();

//Paragramm

s.bewegeBis(10, 200);

s.runter();

s.dreheBis(0);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.dreheUm(135);

s.bewegeUm(50);

s.hoch();

//Stern

s.bewegeBis(120, 200);

s.runter();

s.dreheBis(0);

s.bewegeUm(100);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.dreheUm(45);

s.bewegeUm(50);

s.bewegeUm(-50);

s.hoch();

//Kreis

s.bewegeBis(300, 200);

s.zeichneKreis(50);

s.gibFrei();

b.gibFrei();

}

**Kapitel 4 Kontrollstrukturen**

**4.1 Schleife mit Ausgangsbedingung**

Übung 4.2

import sum.kern.\*;

public class Freihand1

{

Bildschirm b;

Stift s;

Maus m;

public Freihand1()

{

b = new Bildschirm(500,500);

s = new Stift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

s.runter();

do{

s.bewegeBis(m.hPosition(), m.vPosition());

}while(!m.istGedrueckt());

}

}

**4.2 Einseitige Verzweigung**

Übung 4.3

import sum.kern.\*;

public class Freihand2

{

Bildschirm b;

Stift s;

Maus m;

public Freihand2()

{

b = new Bildschirm(500,500);

s = new Stift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.zeichneKreis(3);

}

}while(!m.doppelKlick());

}

}

**4.3 Schleife mit Eingangsbedingung**

Übung 4.4

import sum.kern.\*;

public class Freihand3

{

Bildschirm b;

Stift s;

Maus m;

public Freihand3()

{

b = new Bildschirm(500,500);

s = new Stift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

do{

while(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.zeichneKreis(3);

}

}while(!m.doppelKlick());

}

}

**4.4 Zweiseitige Verzweigung**

Übung 4.5

import sum.kern.\*;

public class Freihand4

{

Bildschirm b;

Stift s;

Maus m;

public Freihand4()

{

b = new Bildschirm(500,500);

s = new Stift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.runter();

}else{

s.hoch();

}

}while(!m.doppelKlick());

}

}

**4.5 Klasse Tastatur**

import sum.kern.\*;

public class Freihand5

{

Bildschirm b;

Stift s;

Maus m;

Tastatur t;

public Freihand5()

{

b = new Bildschirm(500,500);

s = new Stift();

m = new Maus();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.runter();

}else if(t.wurdeGedrueckt()){

s.radiere();

}else{

s.hoch();

}

}while(!m.doppelKlick());

}

}

Übung 4.7

Wurde in Übung 4.6 bereits umgesetzt.

**4.6 Linien zeichnen**

Übung 4.8

import sum.kern.\*;

public class Freihand6

{

Bildschirm b;

Stift s;

Maus m;

Tastatur t;

public Freihand6()

{

b = new Bildschirm(500,500);

s = new Stift();

m = new Maus();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

do{

while(m.istGedrueckt())

{

if(m.istGedrueckt())

{

s.zeichneKreis(3);

}else{

s.runter();

s.bewegeBis(m.hPosition(), m.vPosition());

}

}

}while(!m.doppelKlick());

}

}

**4.7 ist-Beziehung**

Übung 4.9

Diese Übung ist eine Übung zum Ausprobieren ,da kann ich erstmal nicht helfen.

Übung 4.10

import sum.kern.\*;

public class Freihand7

{

Bildschirm b;

Buntstift s;

Maus m;

Tastatur t;

public Freihand7()

{

b = new Bildschirm(500,500);

s = new Buntstift();

m = new Maus();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.runter();

}else if(t.wurdeGedrueckt()){

s.setzeFarbe(10);

}else{

s.hoch();

}

}while(!m.doppelKlick());

}

}

**4.8 Tastaturpuffer**

Übung 4.11

Hier könnte dein Strukturdiagramm sein.

Übung 4.12

import sum.kern.\*;

public class Freihand7

{

Bildschirm b;

Buntstift s;

Maus m;

Tastatur t;

public Freihand7()

{

b = new Bildschirm(500,500);

s = new Buntstift();

m = new Maus();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.runter();

}else if(t.wurdeGedrueckt()){

if(t.zeichen() == 'r')

{

s.setzeFarbe(Farbe.ROT);

}else{

s.setzeFarbe(Farbe.SCHWARZ);

t.weiter();

}

}else{

s.hoch();

}

}while(!m.doppelKlick());

}

}

**4.9 Mehrseitige Verzweigung**

import sum.kern.\*;

public class Freihand9

{

Bildschirm b;

Buntstift s;

Maus m;

Tastatur t;

public Freihand9()

{

b = new Bildschirm(500,500);

s = new Buntstift();

m = new Maus();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.runter();

}else{

s.hoch();

}

if (t.wurdeGedrueckt())

{ switch (t.zeichen())

{ case 'r': case 'R': s.setzeFarbe(Farbe.ROT); break;

case 'b': case 'B': s.setzeFarbe(Farbe.BLAU); break;

case 'g': case 'G': s.setzeFarbe(Farbe.GRUEN); break;

//WEITERE FARBEN HIER

default: s.setzeFarbe(Farbe.SCHWARZ); break;

}

t.weiter();

}

}while(!m.doppelKlick());

}

}

Übung 4.14

import sum.kern.\*;

public class Freihand9

{

Bildschirm b;

Buntstift s;

Maus m;

Tastatur t;

public Freihand9()

{

b = new Bildschirm(500,500);

s = new Buntstift();

m = new Maus();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

do{

if(m.istGedrueckt())

{

s.bewegeBis(m.hPosition(), m.vPosition());

s.runter();

}else{

s.hoch();

}

if (t.wurdeGedrueckt())

{ switch (t.zeichen())

{ case 'r': case 'R': s.setzeFarbe(Farbe.ROT); break;

case 'b': case 'B': s.setzeFarbe(Farbe.BLAU); break;

case 'g': case 'G': s.setzeFarbe(Farbe.GRUEN); break;

default: s.setzeFarbe(Farbe.SCHWARZ); break;

}

switch (t.zeichen())

{ case '1': s.setzeLinienBreite(10); break;

case '2': s.setzeLinienBreite(20); break;

case '3': s.setzeLinienBreite(30); break;

case '4': s.setzeLinienBreite(40); break;

case '5': s.setzeLinienBreite(50); break;

case '6': s.setzeLinienBreite(60); break;

case '7': s.setzeLinienBreite(70); break;

case '8': s.setzeLinienBreite(80); break;

case '9': s.setzeLinienBreite(90); break;

default: s.setzeLinienBreite(1); break;

}

t.weiter();

}

}while(!m.doppelKlick());

}

}

**Kapitel 5 Kontrollstrukturen II**

**5.1 Pfeil und Dartscheibe**

Übung 5.1

import sum.kern.\*;

public class Dart1

{

Bildschirm b;

Buntstift s;

Maus m;

public Dart1()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

this.fuehreAus();

}

}

Übung 5.2

import sum.kern.\*;

public class Dart1

{

Bildschirm b;

Buntstift s;

Maus m;

public Dart1()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

//Dartscheibe zeichnen

s.bewegeBis(500, 200);

s.zeichneKreis(10);

s.zeichneKreis(20);

s.zeichneKreis(30);

s.zeichneKreis(40);

//Pfeil zeichnen

s.bewegeBis(10, 300);

s.dreheUm(45);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

}

}

**5.2 Fliegender Pfeil**

Übung 5.3  
import sum.kern.\*;

public class Dart2

{

Bildschirm b;

Buntstift s;

Maus m;

public Dart2()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

//Dartscheibe zeichnen

s.bewegeBis(500, 200);

s.zeichneKreis(10);

s.zeichneKreis(20);

s.zeichneKreis(30);

s.zeichneKreis(40);

//Pfeil zeichnen

s.bewegeBis(10, 300);

s.dreheUm(12);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

//Pfeil fliegt

do{

s.radiere();

s.bewegeUm(-30);

s.zeichneKreis(3);

s.normal();

s.bewegeUm(0.01);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

}while(true);

}

}

**5.3 Pfeil fällt**

Übung 5.4

import sum.kern.\*;

public class Dart2

{

Bildschirm b;

Buntstift s;

Maus m;

public Dart2()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

//Dartscheibe zeichnen

s.bewegeBis(500, 200);

s.zeichneKreis(10);

s.zeichneKreis(20);

s.zeichneKreis(30);

s.zeichneKreis(40);

int i;

i = 0;

while(!m.istGedrueckt())

{

if( i <= 600)

{

s.radiere();

s.bewegeUm(-30);

s.zeichneKreis(3);

s.normal();

s.bewegeBis(10, i);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

i = i+1;

}else{

i = 0;

}

}

//Pfeil fliegt

do{

s.radiere();

s.bewegeUm(-30);

s.zeichneKreis(3);

s.normal();

s.bewegeUm(0.01);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

}while(s.hPosition()<= 500);

}

}

**5.4 Pfeil dreht**

Übung 5.5

import sum.kern.\*;

public class Dart2

{

Bildschirm b;

Buntstift s;

Maus m;

public Dart2()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

//Dartscheibe zeichnen

s.bewegeBis(500, 200);

s.zeichneKreis(10);

s.zeichneKreis(20);

s.zeichneKreis(30);

s.zeichneKreis(40);

int i;

i = 0;

while(!m.istGedrueckt())

{

if( i <= 600)

{

s.radiere();

s.bewegeUm(-30);

s.zeichneKreis(3);

s.normal();

s.bewegeBis(10, i);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

i = i+1;

}else{

i = 0;

}

}

while(m.istGedrueckt())

{

s.dreheUm(i);

s.radiere();

s.bewegeUm(-30);

s.zeichneKreis(3);

s.normal();

s.bewegeUm(0.01);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

i = i+1;

}

//Pfeil fliegt

do{

s.radiere();

s.bewegeUm(-30);

s.zeichneKreis(3);

s.normal();

s.bewegeUm(0.01);

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.hoch();

}while(s.hPosition()<= 500);

}

}

**5.5 Treffer!**

Übung 5.6

Am ende der fuehreAus() Funktion einfügen.

if(s.vPosition() == 200)

{

s.bewegeBis(300, 200);

s.schreibeText("GETROFFEN");

}else{

s.bewegeBis(300, 200);

s.schreibeText("DANNEBEN");

}

**5.6 Wieder hoch!**

>Übung 5.7

Diese Funktion wurde bereits implementiert.

**Das Dart Einspielerprojekt etwas verschönert**:

import sum.kern.\*;

import sum.werkzeuge.\*;

public class Dartspiel\_für\_eine\_Person

{

Bildschirm b;

Buntstift s;

Maus m;

Rechner r;

public Dartspiel\_für\_eine\_Person()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

r = new Rechner();

this.fuehreAus();

}

public void fuehreAus()

{

//Dartscheibe zeichnen

s.bewegeBis(500, r.ganzeZufallsZahl(50,b.hoehe()-50));

double ScheibeVPosition = s.vPosition();

for(int i=0;i<=50;i++)

{

if(i<10){

s.setzeFarbe(12);

}else if(i<20){

s.setzeFarbe(10);

}else if(i<30){

s.setzeFarbe(1);

}else if(i<40){

s.setzeFarbe(0);

}else{

i = 50;

}

s.zeichneKreis(i);

}

//Pfeil fällt

s.bewegeBis(10, -1);

while(!m.istGedrueckt()){

//Pfeil radieren

s.radiere();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.normal();

//Pfeil bewegen

s.dreheBis(270);

s.bewegeUm(1);

s.dreheBis(-270);

//Pfeil zeichnen

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.hoch();

if(s.vPosition()>b.hoehe()){

s.bewegeBis(10, -1);

}

}

//Pfeil drehen

double i = 0;

while(m.istGedrueckt()){

//Pfeil radieren

s.radiere();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.normal();

//Pfeil drehen

s.dreheUm(i);

//Pfeil zeichnen

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.hoch();

i = i+0.0001;

}

//Pfeil fliegt

i = 0;

while(s.hPosition()<500){

//Pfeil radieren

s.radiere();

s.bewegeUm(40);

s.zeichneKreis(3);

s.bewegeUm(-40);

s.normal();

//Pfeil drehen

s.bewegeUm(i);

//Pfeil zeichnen

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.hoch();

i = i+0.01;

}

//Treffer?

if(s.vPosition() == ScheibeVPosition)

{

s.bewegeBis(300, 200);

s.schreibeText("GETROFFEN");

}else{

s.bewegeBis(300, 200);

s.schreibeText("DANNEBEN");

}

}

}

**5.7 Zwei Spieler**

import sum.kern.\*;

import sum.werkzeuge.\*;

public class Dartspiel\_für\_zwei\_Personen

{

Bildschirm b;

Buntstift s;

Maus m;

Rechner r;

Tastatur t;

public Dartspiel\_für\_zwei\_Personen()

{

b = new Bildschirm(600,400);

s = new Buntstift();

m = new Maus();

r = new Rechner();

t = new Tastatur();

this.fuehreAus();

}

public void fuehreAus()

{

//Dartscheibe zeichnen

s.bewegeBis(500, r.ganzeZufallsZahl(50,b.hoehe()-50));

double ScheibeVPosition = s.vPosition();

for(int i=0;i<=50;i++)

{

if(i<10){

s.setzeFarbe(12);

}else if(i<20){

s.setzeFarbe(10);

}else if(i<30){

s.setzeFarbe(1);

}else if(i<40){

s.setzeFarbe(0);

}else{

i = 50;

}

s.zeichneKreis(i);

}

//Pfeil fällt

s.bewegeBis(10, -1);

while(!m.istGedrueckt()){

//Pfeil radieren

s.radiere();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.normal();

//Pfeil bewegen

s.dreheBis(270);

s.bewegeUm(1);

s.dreheBis(-270);

//Pfeil zeichnen

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.hoch();

if(s.vPosition()>b.hoehe()){

s.bewegeBis(10, -1);

}

}

//Pfeil drehen

double i = 0;

while(m.istGedrueckt()){

//Pfeil radieren

s.radiere();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.normal();

//Pfeil drehen

s.dreheUm(i);

//Pfeil zeichnen

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.hoch();

i = i+0.0001;

}

//Pfeil fliegt

i = 0;

while(s.hPosition()<500){

//Pfeil radieren

s.radiere();

s.bewegeUm(40);

s.zeichneKreis(3);

s.bewegeUm(-40);

s.normal();

//Pfeil drehen

if (t.wurdeGedrueckt())

{

switch (t.zeichen())

{ case 'r': case 'R': s.dreheUm(5); break;

case 'l': case 'L': s.dreheUm(-5); break;

default: s.dreheUm(-5); break;

}

}

s.bewegeUm(i);

//Pfeil zeichnen

s.runter();

s.bewegeUm(30);

s.zeichneKreis(3);

s.bewegeUm(-30);

s.hoch();

i = i+0.001;

}

//Treffer?

if(s.vPosition() == ScheibeVPosition)

{

s.bewegeBis(300, 200);

s.schreibeText("GETROFFEN");

}else{

s.bewegeBis(300, 200);

s.schreibeText("DANNEBEN");

}

}

}

**5.8 Jetzt wird's bunt**

Diese Funktion wurde bereits realisiert.

**5.9 Zufallszahlen**

Übung 5.11

Struktogram

Übung 5.12

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Himmel

{

// Objekte

Bildschirm b;

Stift s;

Rechner r;

Maus m;

// Konstruktor

public Himmel()

{

b = new Bildschirm(600,600);

s = new Stift();

m = new Maus();

r = new Rechner();

this.fuehreAus();

}

public void fuehreAus()

{

while(!m.istGedrueckt()){

s.bewegeBis(r.ganzeZufallszahl(1,b.breite()), r.ganzeZufallsZahl(1,b.hoehe()));

s.zeichneKreis(1);

}

}

}

Übung 5.13

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Himmel

{

// Objekte

Bildschirm b;

Buntstift s;

Rechner r;

Maus m;

// Konstruktor

public Himmel()

{

b = new Bildschirm(600,600);

s = new Buntstift();

m = new Maus();

r = new Rechner();

this.fuehreAus();

}

public void fuehreAus()

{

while(!m.istGedrueckt()){

s.bewegeBis(r.ganzeZufallszahl(1,b.breite()), r.ganzeZufallsZahl(1,b.hoehe()));

s.setzeFarbe(r.ganzeZufallsZahl());

s.zeichneKreis(1);

}

}

}

Übung 5.14

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Linien

{

// Objekte

Bildschirm b;

Buntstift s;

Rechner r;

Maus m;

// Konstruktor

public Linien()

{

b = new Bildschirm(600,600);

s = new Buntstift();

m = new Maus();

r = new Rechner();

this.fuehreAus();

}

public void fuehreAus()

{

while(!m.istGedrueckt()){

s.bewegeBis(r.ganzeZufallszahl(1,b.breite()), r.ganzeZufallsZahl(1,b.hoehe()));

s.setzeFarbe(r.ganzeZufallsZahl());

s.runter();

s.setzeLinienBreite(r.ganzeZufallsZahl(0,10));

s.bewegeUm(r.ganzeZufallsZahl());

s.hoch();

}

}

}

Übung 5.15

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Wanderer

{

// Objekte

Bildschirm b;

Buntstift s;

Rechner r;

Maus m;

// Konstruktor

public Wanderer()

{

b = new Bildschirm(600,600);

s = new Buntstift();

m = new Maus();

r = new Rechner();

this.fuehreAus();

}

public void fuehreAus()

{

s.bewegeBis(r.ganzeZufallsZahl(0, b.breite()), r.ganzeZufallsZahl(0, b.hoehe()));

do{

s.runter();

s.bewegeUm(1);

s.dreheUm(r.ganzeZufallsZahl(0, 360));

}while(true);

}

}

**5.10 Schatz verstecken**

Übung 5.16

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Schatz1

{

Bildschirm b;

Buntstift schatzstift;

Rechner r;

public Schatz1()

{

b = new Bildschirm(600,600);

r = new Rechner();

schatzstift = new Buntstift();

this.fuehreAus();

}

public void fuehreAus()

{

schatzstift.bewegeBis(r.ganzeZufallsZahl(50,b.breite()-50), r.ganzeZufallsZahl(50,b.hoehe()-50));

schatzstift.setzeFuellMuster(2);

schatzstift.setzeFarbe(12);

schatzstift.zeichneKreis(30);

}

}

**5.11 Abstand berechnen**

Übung 5.17

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Schatz1

{

Bildschirm b;

Buntstift schatzStift;

Stift schreibStift;

Rechner r;

Maus m;

public Schatz1()

{

b = new Bildschirm(600,600);

r = new Rechner();

schatzStift = new Buntstift();

schreibStift = new Stift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

schatzStift.bewegeBis(r.ganzeZufallsZahl(50,b.breite()-50), r.ganzeZufallsZahl(50,b.hoehe()-50));

schatzStift.setzeFuellMuster(2);

schatzStift.setzeFarbe(12);

schatzStift.zeichneKreis(30);

schreibStift.bewegeBis(10, 10);

do{

schreibStift.schreibeZahl(r.wurzel(

r.quadrat(m.hPosition()-

schatzStift.hPosition())) +

r.wurzel(r.quadrat(m.vPosition()-

schatzStift.vPosition())));

}while(!m.doppelKlick());

}

}

Übung 5.18

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Schatz1

{

Bildschirm b;

Buntstift schatzStift;

Stift schreibStift;

Rechner r;

Maus m;

public Schatz1()

{

b = new Bildschirm(600,600);

r = new Rechner();

schatzStift = new Buntstift();

schreibStift = new Stift();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

schatzStift.bewegeBis(r.ganzeZufallsZahl(50,b.breite()-50), r.ganzeZufallsZahl(50,b.hoehe()-50));

schatzStift.setzeFuellMuster(2);

schreibStift.bewegeBis(10, 10);

do{

double i =

r.wurzel(

r.quadrat(m.hPosition()-schatzStift.hPosition())) + r.wurzel(r.quadrat(m.vPosition()-

schatzStift.vPosition()));

if(i < 15){

schatzStift.setzeFarbe(12);

schatzStift.zeichneKreis(30);

}

}while(!m.doppelKlick());

}

}

**5.12 Schnell suchen!**

Übung 5.19

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Schatz\_mit\_Uhr

{

Bildschirm b;

Buntstift schatzStift;

Stift schreibStift;

Rechner r;

Maus m;

Uhr u;

public Schatz\_mit\_Uhr()

{

b = new Bildschirm(600,600);

r = new Rechner();

schatzStift = new Buntstift();

schreibStift = new Stift();

m = new Maus();

u = new Uhr();

this.fuehreAus();

}

public void fuehreAus()

{ schatzStift.bewegeBis(r.ganzeZufallsZahl(50,b.breite()-50), r.ganzeZufallsZahl(50,b.hoehe()-50));

schatzStift.setzeFuellMuster(2);

schreibStift.bewegeBis(10, 10);

do{

double i =

r.wurzel(

r.quadrat(m.hPosition()-schatzStift.hPosition())) + r.wurzel(r.quadrat(m.vPosition()-

schatzStift.vPosition()));

if(i < 15){

schatzStift.setzeFarbe(12);

schatzStift.zeichneKreis(30);

}else if (u.verstricheneZeit() <60){

this.fuehreAus();

}

}while(!m.doppelKlick());

}

}

Übung 5.20

Was soll ich hier machen Screenshots ach comon das könnt ihr selber (wenn v1 hiervon raus ist können wir über v2 reden)

Übung 5.21

Struktogramm kommt in v2 :)

**5.13 Zusammenfassung**

Super,alles super ,aber hohlt bitte deutsche Wörter aus Java . Java ist ohne sum Verständlicher und schöner.

**Kapitel 6 Eigene Klassen I**

6.1 Eigene Unterklassen erzeugen

Übung 6.1

Das genutzte Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Figurstift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

s = new Figurstift();

this.fuehreAus();

}

public void fuehreAus()

{

//Rechteck

s.bewegeBis(200, 100);

s.zeichneQuadrat(10);

//Ohne Fehler

s.bewegeBis(100, 100);

s.zeichneRing(5, 10);

//Für den Fehler

s.zeichneRing(10, 5);

}

}

Der Figurstift:

import sum.kern.\*;

public class Figurstift extends Stift

{

public Figurstift()

{

super();

}

public void zeichneQuadrat(double pGroesse)

{

this.zeichneRechteck(pGroesse,pGroesse);

}

public void zeichneRing(double pInnererRing, double pAeußererRing)

{

if(pAeußererRing > pInnererRing){

this.zeichneKreis(pInnererRing);

this.zeichneKreis(pAeußererRing);

}else

{

System.out.println("Fehler! Innerer Ring muss größer als aeußerer Ring sein.");

}

}

}

Übung 6.2

Das verwendete Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Planstift s;

Tastatur t;

Maus m;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

s = new Planstift();

t = new Tastatur();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

do{

s.bewegeBis(m.hPosition(), m.vPosition());

if (t.wurdeGedrueckt())

{ switch (t.zeichen())

{ case 'w': case 'W': s.zeichneWand(); break;

case 'f': case 'F': s.zeichneFenster(); break;

case 't': case 'T': s.zeichneTuer(); break;

case 'l': case 'L': s.zeichneEckeLinks(); break;

case 'r': case 'R': s.zeichneEckeRechts(); break;

default: s.zeichneWand(); break;

}

t.weiter();

}

}while (true);

}

}

Der Planstift:

import sum.kern.\*;

public class Planstift extends Stift

{

public Planstift()

{

super();

}

public void zeichneWand()

{

this.runter();

this.bewegeUm(30);

this.hoch();

this.dreheBis(90);

this.bewegeUm(6);

this.runter();

this.bewegeUm(-30);

this.hoch();

}

public void zeichneFenster()

{

this.dreheBis(-90);

this.runter();

this.bewegeUm(6);

this.bewegeUm(-3);

this.dreheBis(0);

this.bewegeUm(30);

this.dreheBis(-90);

this.bewegeUm(-3);

this.bewegeUm(6);

this.hoch();

}

public void zeichneTuer()

{

this.dreheBis(-90);

this.runter();

this.bewegeUm(6);

this.bewegeUm(-3);

this.dreheBis(30);

this.bewegeUm(30);

this.bewegeUm(-30);

this.dreheBis(0);

this.hoch();

this.bewegeUm(30);

this.runter();

this.dreheBis(-90);

this.bewegeUm(-3);

this.bewegeUm(6);

this.hoch();

}

public void zeichneEckeLinks()

{

this.runter();

this.bewegeUm(6);

this.dreheBis(90);

this.bewegeUm(6);

this.hoch();

}

public void zeichneEckeRechts()

{

this.runter();

this.bewegeUm(6);

this.dreheBis(-90);

this.bewegeUm(6);

this.hoch();

}

}

**6.2 Klassendiagramm**

Übung 6.3

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

//Attribute

//Konstruktor

public Kugel()

{

}

//Dienste

public void gibfrei()

{

}

public void zeichne()

{

}

public void bewege()

{

}

}

Übung 6.4

Ich entschuldige mich an der Stelle,allerdings habe ich keine Ahnung was diese Aufgabe von mir Verlangt.

**6.3 hat-Beziehung**

Übung 6.5

Kürzer,grade Objektorientiertes programmieren kann sehr groß werden deswegen alles verkürzen.

Und doppelter Code ist meist unnötig.

Übung 6.6

Das genutzte Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Kugel k;

Maus m;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

k = new Kugel();

m = new Maus();

this.fuehreAus();

}

public void fuehreAus()

{

while(!m.istGedrueckt())

{

k.bewege();

}

}

}

Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

//Konstruktor

public Kugel()

{

s = new Stift();

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(5);

}

public void bewege()

{

s.bewegeUm(0.1);

this.zeichne();

}

}

Ich entschuldige mich aber die Frage habe ich leider nicht verstanden :(

Übung 6.7

Hier die modifizierte Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

//Konstruktor

public Kugel()

{

s = new Stift();

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(5);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(5);

s.normal();

}

public void bewege()

{

this.loesche();

s.bewegeUm(0.1);

this.zeichne();

}

}

**6.4 Beziehungsdiagramm**

Übung 6.8

Das genuzte Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Kugel k;

Maus m;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

k = new Kugel(100,100);

m = new Maus();

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

s.bewegeBis(10, 10);

s.zeichneRechteck(570, 570);

while(!m.istGedrueckt())

{

k.bewege();

if(k.hPosition() > 570)

{

k.setzeRichtung(-180);

}

if(k.hPosition() < 16)

{

k.setzeRichtung(0);

}

if(k.vPosition() > 570)

{

k.setzeRichtung(90);

}

if(k.vPosition() < 16)

{

k.setzeRichtung(-90);

}

}

}

}

Die genutzte Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

//Konstruktor

public Kugel(double pH,double pV)

{

s = new Stift();

s.bewegeBis(pH, pV);

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(5);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(5);

s.normal();

}

public void bewege()

{

this.loesche();

s.bewegeUm(0.1);

this.zeichne();

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

Übung 6.9:

Das genuzte Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Kugel k;

Maus m;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

k = new Kugel(100,100);

m = new Maus();

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

s.bewegeBis(10, 10);

s.zeichneRechteck(570, 570);

while(!m.istGedrueckt())

{

k.bewege();

}

}

}

Die Verwendete Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

//Konstruktor

public Kugel(double pH,double pV)

{

s = new Stift();

s.bewegeBis(pH, pV);

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(5);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(5);

s.normal();

}

private boolean amLinkenRand()

{

return this.hPosition() < 16;

}

private boolean amRechtenRand()

{

return this.hPosition() > 570;

}

public void bewege()

{

this.loesche();

s.bewegeUm(0.1);

this.zeichne();

if(amRechtenRand()== true)

{

this.setzeRichtung(-180);

}

if(amLinkenRand() == true)

{

this.setzeRichtung(0);

}

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

**6.5 Attribute**

Übung 6.10

Die modifizierte Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

private int zGroesse;

private double zGeschwindigkeit;

//Konstruktor

public Kugel(double pH,double pV,int pGroesse,double pGeschwindigkeit)

{

s = new Stift();

s.bewegeBis(pH, pV);

zGroesse = pGroesse;

zGeschwindigkeit = pGeschwindigkeit;

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(zGroesse);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(5);

s.normal();

}

private boolean amLinkenRand()

{

return this.hPosition() < 16;

}

private boolean amRechtenRand()

{

return this.hPosition() > 570;

}

public void bewege()

{

this.loesche();

s.bewegeUm(zGeschwindigkeit);

this.zeichne();

if(amRechtenRand()== true)

{

this.setzeRichtung(-180);

}

if(amLinkenRand() == true)

{

this.setzeRichtung(0);

}

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

Übung 6.11

Das verwendete Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Kugel k1,k2,k3;

Maus m;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

k1 = new Kugel(100,100,5,0.1);

k2 = new Kugel(150,150,7,0.2);

k3 = new Kugel(170,170,10,0.3);

m = new Maus();

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

s.bewegeBis(10, 10);

s.zeichneRechteck(570, 570);

while(!m.istGedrueckt())

{

k1.bewege();

k2.bewege();

k3.bewege();

}

}

}

Die verbesserte Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

private int zGroesse;

private double zGeschwindigkeit;

//Konstruktor

public Kugel(double pH,double pV,int pGroesse,double pGeschwindigkeit)

{

s = new Stift();

s.bewegeBis(pH, pV);

zGroesse = pGroesse;

zGeschwindigkeit = pGeschwindigkeit;

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(zGroesse);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(zGroesse);

s.normal();

}

private boolean amLinkenRand()

{

return this.hPosition() < 11 + zGroesse;

}

private boolean amRechtenRand()

{

return this.hPosition() > 570 - zGroesse;

}

public void bewege()

{

this.loesche();

s.bewegeUm(zGeschwindigkeit);

this.zeichne();

if(amRechtenRand()== true)

{

this.setzeRichtung(-180);

}

if(amLinkenRand() == true)

{

this.setzeRichtung(0);

}

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

Übung 6.12

Das verwendete Hauptprogramm:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

//Attribute

private int zGroesse , zRichtung;

private double zGeschwindigkeit;

//Konstruktor

public Kugel(double pH,double pV,int pGroesse,double pGeschwindigkeit, int pRichtung)

{

s = new Stift();

s.bewegeBis(pH, pV);

zGroesse = pGroesse;

zGeschwindigkeit = pGeschwindigkeit;

zRichtung = pRichtung;

s.dreheBis(pRichtung);

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(zGroesse);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(zGroesse);

s.normal();

}

private boolean amLinkenRand()

{

return this.hPosition() < 11 + zGroesse;

}

private boolean amRechtenRand()

{

return this.hPosition() > 570 - zGroesse;

}

private boolean amOberenRand()

{

return this.vPosition() < 11 + zGroesse;

}

private boolean amUnterenRand()

{

return this.vPosition() > 570 - zGroesse;

}

public void bewege()

{

this.loesche();

s.bewegeUm(zGeschwindigkeit);

this.zeichne();

if(this.amLinkenRand()||this.amRechtenRand())

{

s.dreheBis(180 - s.winkel());

}else if(this.amOberenRand()||this.amUnterenRand())

{

s.dreheBis(360 - s.winkel());

}

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

Das verwendete Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Kugel k1,k2,k3;

Maus m;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

k1 = new Kugel(100,100,5,0.1,10);

k2 = new Kugel(150,150,7,0.2,30);

k3 = new Kugel(170,170,10,0.3,70);

m = new Maus();

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

s.bewegeBis(10, 10);

s.zeichneRechteck(570, 570);

while(!m.istGedrueckt())

{

k1.bewege();

k2.bewege();

k3.bewege();

}

}

}

**6.6 kennt-Beziehung**

Übung 6.14

Das genutzte Hauptprogramm:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

Bildschirm zBildschirm;

//Attribute

private int zGroesse , zRichtung;

private double zGeschwindigkeit;

//Konstruktor

public Kugel(double pH,double pV,int pGroesse,double pGeschwindigkeit, int pRichtung,Bildschirm kenntBildschirm)

{

s = new Stift();

s.bewegeBis(pH, pV);

zGroesse = pGroesse;

zGeschwindigkeit = pGeschwindigkeit;

zRichtung = pRichtung;

zBildschirm = kenntBildschirm;

s.dreheBis(pRichtung);

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(zGroesse);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(zGroesse);

s.normal();

}

private boolean amLinkenRand()

{

return this.hPosition() < 0 + zGroesse;

}

private boolean amRechtenRand()

{

return this.hPosition() > zBildschirm.breite() - zGroesse;

}

private boolean amOberenRand()

{

return this.vPosition() < 0 + zGroesse;

}

private boolean amUnterenRand()

{

return this.vPosition() > zBildschirm.hoehe() - zGroesse;

}

public void bewege()

{

this.loesche();

s.bewegeUm(zGeschwindigkeit);

this.zeichne();

if(this.amLinkenRand()||this.amRechtenRand())

{

s.dreheBis(180 - s.winkel());

}else if(this.amOberenRand()||this.amUnterenRand())

{

s.dreheBis(360 - s.winkel());

}

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

Die genutzte Kugel:

import sum.kern.\*;

public class Kugel

{

//Bezugsobjecte

Stift s;

Bildschirm zBildschirm;

//Attribute

private int zGroesse , zRichtung;

private double zGeschwindigkeit;

//Konstruktor

public Kugel(double pH,double pV,int pGroesse,double pGeschwindigkeit, int pRichtung,Bildschirm kenntBildschirm)

{

s = new Stift();

s.bewegeBis(pH, pV);

zGroesse = pGroesse;

zGeschwindigkeit = pGeschwindigkeit;

zRichtung = pRichtung;

zBildschirm = kenntBildschirm;

s.dreheBis(pRichtung);

}

//Dienste

public void gibfrei()

{

s.gibFrei();

}

public void zeichne()

{

s.zeichneKreis(zGroesse);

}

public void loesche()

{

s.radiere();

s.zeichneKreis(zGroesse);

s.normal();

}

private boolean amLinkenRand()

{

return this.hPosition() < 0 + zGroesse;

}

private boolean amRechtenRand()

{

return this.hPosition() > zBildschirm.breite() - zGroesse;

}

private boolean amOberenRand()

{

return this.vPosition() < 0 + zGroesse;

}

private boolean amUnterenRand()

{

return this.vPosition() > zBildschirm.hoehe() - zGroesse;

}

public void bewege()

{

this.loesche();

s.bewegeUm(zGeschwindigkeit);

this.zeichne();

if(this.amLinkenRand()||this.amRechtenRand())

{

s.dreheBis(180 - s.winkel());

}else if(this.amOberenRand()||this.amUnterenRand())

{

s.dreheBis(360 - s.winkel());

}

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public void setzeRichtung(double pRichtung)

{

s.dreheBis(pRichtung);

}

}

**6.7 ist-Beziehung**

Übung 6.15

Die Reibungskugel:

import sum.kern.\*;

public class Reibungskugel extends Kugel

{

//Bezugsobjecte

//Attribute

private double zGeschwindigkeit;

//Konstruktor

public Reibungskugel(int pH, int pV, int pGroesse,double pGeschwindigkeit, int pRichtung, Bildschirm kenntBildschirm)

{

super(pH, pV, pGroesse, pGeschwindigkeit, pRichtung,kenntBildschirm);

}

public void setzeGeschwindigkeit(double i)

{

zGeschwindigkeit = zGeschwindigkeit \* i;

}

private void bremse()

{

this.setzeGeschwindigkeit(0.99);

}

public void bewege()

{

this.bremse();

super.bewege();

}

}

Übung 6.16

Das Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

Bildschirm b;

Farbkugel k1;

Maus m;

Stift s;

public Hauptprogramm()

{

b = new Bildschirm(600,600);

k1 = new Farbkugel(100,100,5,0.2,30,b,10);

m = new Maus();

s = new Stift();

this.fuehreAus();

}

public void fuehreAus()

{

while(!m.istGedrueckt())

{

k1.bewege();

}

}

}

Die Farbkugel:

import sum.kern.\*;

public class Farbkugel extends Kugel

{

// Bezugsobjekte

private Buntstift b;

// Attribute

private int zFarbe;

double zGroesse;

//Konstruktor

public Farbkugel(int pH,int pV,int pGroesse,double pGeschwindigkeit,int pRichtung,Bildschirm pBildschirm, int pFarbe)

{

super(pH,pV,pGroesse,pGeschwindigkeit, pRichtung,pBildschirm);

b = new Buntstift();

b.bewegeBis(pH, pV);

b.setzeFuellmuster(Muster.GEFUELLT);

b.setzeFarbe(pFarbe);

zGroesse = pGroesse;

}

public void zeichne()

{

b.bewegeBis(this.hPosition(), this.vPosition());

b.zeichneKreis(zGroesse);

}

public void loesche()

{

b.radiere();

this.zeichne();

b.normal();

}

}

Übung 6.17

Spurkugel:

import sum.kern.\*;

public class Spurkugel extends Kugel

{

//Bezugsobjecte

//Attribute

private double zGeschwindigkeit;

private int zGroesse;

//Konstruktor

public Spurkugel(int pH,int pV,int pGroesse,double pGeschwindigkeit,int pRichtung,Bildschirm kenntBildschirm)

{

super(pH, pV, pGroesse, pGeschwindigkeit, pRichtung,kenntBildschirm);

zGroesse = pGroesse;

}

public void loesche()

{

}

public void bewege()

{

super.bewege();

}

}

Zahlkugel:

Pulsierkugel:

Ich habs versucht ich hab aber noch keine Ahnung warums grad nicht funktioniert ich werds später nochmal versuchen.

Übung 6.18

Nein

Übung 6.19

Gleiches Problem ka bin anscheinend kein Leistungstarker schüler mehr

na im sorry ich muss mir die sum.kern library nochmal angucken ich bin damit noch nicht so gut :(

**Kapitel 7 Eigene Klassen II**

**7.1 Zufallsweg**

Übung 7.1

Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

// Objekte

Bildschirm b;

Wolf w;

// Konstruktor

public Hauptprogramm()

{

b = new Bildschirm(600,600);

w = new Wolf(b);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

do{

w.bewege();

}while(true);

}

}

Normales Programm:

import sum.werkzeuge.\*;

import sum.kern.\*;

public class Wolf

{

// Objekte

Stift s;

Bildschirm zBildschirm;

Rechner r;

// Konstruktor

public Wolf(Bildschirm pBildschirm)

{

s = new Stift();

r = new Rechner();

zBildschirm = pBildschirm;

s.bewegeBis(100, 100);

s.runter();

}

// Dienste

private boolean amLinkenRand()

{

return s.hPosition() < 0;

}

private boolean amRechtenRand()

{

return s.hPosition() > zBildschirm.breite();

}

private boolean amOberenRand()

{

return s.vPosition() < 0;

}

private boolean amUnterenRand()

{

return s.vPosition() > zBildschirm.hoehe();

}

public void bewege()

{

s.bewegeUm(1);

if(this.amLinkenRand()||this.amRechtenRand())

{

s.dreheBis(180 - s.winkel());

}else if(this.amOberenRand()||this.amUnterenRand())

{

s.dreheBis(360 - s.winkel());

}else{

s.dreheUm(r.ganzeZufallsZahl(-90, 90));

}

}

}

Verändrete Schrittlänge und Winkelintervall:

import sum.kern.\*;

import sum.werkzeuge.\*;

public class Wolf

{

// Objekte

Stift s;

Bildschirm zBildschirm;

Rechner r;

// Konstruktor

public Wolf(Bildschirm pBildschirm)

{

s = new Stift();

r = new Rechner();

zBildschirm = pBildschirm;

s.bewegeBis(100, 100);

s.runter();

}

// Dienste

private boolean amLinkenRand()

{

return s.hPosition() < 0;

}

private boolean amRechtenRand()

{

return s.hPosition() > zBildschirm.breite();

}

private boolean amOberenRand()

{

return s.vPosition() < 0;

}

private boolean amUnterenRand()

{

return s.vPosition() > zBildschirm.hoehe();

}

public void bewege()

{

s.bewegeUm(0.1);

if(this.amLinkenRand()||this.amRechtenRand())

{

s.dreheBis(180 - s.winkel());

}else if(this.amOberenRand()||this.amUnterenRand())

{

s.dreheBis(360 - s.winkel());

}else{

s.dreheUm(r.ganzeZufallsZahl(0, 360));

}

}

}

7.2 Weg zum Ziel

Übung 7.2

Übung 7.3

ICH WERD VERRÜCKT, Sorry ergänzung ich bin.

Im Folgenden finden sich fehler ich weiß net warum wird aber verbessert.

Das falsche Hauptprogramm:

import sum.kern.\*;

public class Hauptprogramm

{

// Objekte

Bildschirm b;

Wolf w;

Rotkaepchen r;

// Konstruktor

public Hauptprogramm()

{

b = new Bildschirm(600,600);

r = new Rotkaepchen(b);

w = new Wolf(b,r);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

do{

w.bewege();

}while(true);

}

}

Der fehlerhafte Wolf:

import sum.kern.\*;

import sum.werkzeuge.\*;

public class Wolf

{

// Objekte

Stift s;

Bildschirm zBildschirm;

Rotkaepchen zRotkaepchen;

Rechner r;

public double zAbstand = 0;

// Konstruktor

public Wolf(Bildschirm pBildschirm,Rotkaepchen pRotkaepchen)

{

s = new Stift();

r = new Rechner();

zBildschirm = pBildschirm;

zRotkaepchen = pRotkaepchen;

zAbstand = this.abstand();

s.bewegeBis(100, 100);

s.runter();

}

// Dienste

private double abstand()

{

double xR = zRotkaepchen.hPosition();

double xW = s.hPosition();

double yR = zRotkaepchen.vPosition();

double yW = s.vPosition();

return xR\*yR + xW\*yW;

}

private boolean amLinkenRand()

{

return s.hPosition() < 0;

}

private boolean amRechtenRand()

{

return s.hPosition() > zBildschirm.breite();

}

private boolean amOberenRand()

{

return s.vPosition() < 0;

}

private boolean amUnterenRand()

{

return s.vPosition() > zBildschirm.hoehe();

}

public void bewege()

{

s.bewegeUm(1);

if(this.amLinkenRand()||this.amRechtenRand())

{

s.dreheBis(180 - s.winkel());

}else if(this.amOberenRand()||this.amUnterenRand())

{

s.dreheBis(360 - s.winkel());

}else{

s.dreheUm(r.ganzeZufallsZahl( 360,0));

}

if(this.abstand() > zAbstand)

{

s.dreheUm(90);

zAbstand = this.abstand();

}

}

}

Das fehlerhafte Rotkäpchen:

import sum.kern.\*;

import sum.werkzeuge.\*;

public class Rotkaepchen

{

// Objekte

Bildschirm zBildschirm;

Buntstift s;

Rechner r;

// Konstruktor

public Rotkaepchen(Bildschirm pBildschirm)

{

zBildschirm = pBildschirm;

s = new Buntstift();

r = new Rechner();

s.bewegeBis(r.ganzeZufallsZahl(0, zBildschirm.breite()),r.ganzeZufallsZahl(0, zBildschirm.hoehe()));

s.zeichneKreis(10);

s.setzeFuellMuster(2);

s.zeichneKreis(10);

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

}

**7.3 Intelligenz?**

Übung 7.4

Übung 7.5

Übung 7.6

DAS HIER FIXEN WENN WOLF RICHTIG IST !!!!!!

**7.4 Minigolf**

**7.5 Treffen!**

**7.6 Zusammenfassung**

**Kapitel 8 Abstrakte Klassen I**

**8.1 Substantiv-Verb-Methode**

**8.2 Dynamische Referenz**

**8.3 Generalisierung**

**8.4 Späte Bindung**

**8.5 Sichtbarkeit**

**8.6 Erweiterungen**

**8.7 Zusammenfassung**

**Kapitel 9 Abstrakte Klassen II**

**9.1 Entwurf**

Übung 9.1

Benötigte Dienste:

-zeichnen (in den Unterklassen genauer Definiert)

-löschen (in den Unterklassen genauer Definiert)

-bewegen (Oberklasse)

-bewegeUm (Oberklasse)

-bewegeBis (Oberklasse)

-hPosition (Oberklasse)

-vPosition (Oberklasse)

Steht auf der nächsten Seite im Struktogramm

Übung 9.2

-länge (Oberklasse)

-istAktiv (Oberklasse)

**9.2 Lokale Variable**

Übung 9.3

Lokomotieve:

import sum.kern.\*;

public class Lokomotive extends Waggon{

// Bezugsobjekte

Stift s;

// Attribute

int zH,zV;

// Konstruktor

public Lokomotive(int pH, int pV)

{

super(pH,pV);

zH = pH;

zV = pV;

s = new Stift();

super.setzeLaenge(90);

}

protected void zeichne()

{

s.bewegeBis(zH, zV);

s.runter();

s.dreheBis(90);

s.bewegeUm(3);

s.dreheUm(180);

s.bewegeUm(6);

s.dreheUm(180);

s.bewegeUm(3);

s.dreheUm(270);

s.bewegeUm(5);

s.hoch();

s.dreheUm(90);

s.bewegeUm(20);

s.zeichneRechteck(50,25);

s.bewegeUm(15);

s.dreheUm(-90);

s.bewegeUm(10);

s.zeichneRechteck(5,15);

s.bewegeUm(40);

s.dreheUm(90);

s.bewegeUm(10);

s.zeichneRechteck(30,50);

s.bewegeUm(-5);

s.dreheUm(-90);

s.bewegeUm(5);

s.zeichneRechteck(20,20);

s.dreheUm(-90);

s.bewegeUm(50);

s.dreheUm(90);

s.bewegeUm(15);

s.zeichneKreis(5);

s.bewegeUm(-60);

s.zeichneKreis(5);

s.bewegeUm(70);

s.dreheUm(90);

s.bewegeUm(10);

s.runter();

s.dreheUm(270);

s.bewegeUm(5);

s.dreheUm(90);

s.bewegeUm(3);

s.dreheUm(180);

s.bewegeUm(6);

s.dreheUm(180);

s.bewegeUm(3);

s.hoch();

}

}

Personenwagen:

import sum.kern.\*;

public class Personenwagen extends Waggon

{

// Bezugsobjekte

Stift s;

// Attribute

int zH,zV;

// Konstruktor

public Personenwagen(int pH, int pV)

{

super(pH,pV);

zH = pH;

zV = pV;

s = new Stift();

super.setzeLaenge(90);

}

protected void zeichne()

{

s.bewegeBis(zH, zV);

s.runter();

s.dreheBis(90);

s.bewegeUm(3);

s.dreheUm(180);

s.bewegeUm(6);

s.dreheUm(180);

s.bewegeUm(3);

s.dreheUm(270);

s.bewegeUm(5);

s.dreheUm(90);

s.bewegeUm(45);

s.zeichneRechteck(80,50);

s.hoch();

s.dreheUm(270);

s.bewegeUm(4);

s.dreheUm(270);

s.bewegeUm(10);

s.runter();

s.zeichneRechteck(15,35);

s.hoch();

s.dreheUm(90);

s.bewegeUm(19);

s.runter();

s.zeichneRechteck(15,10);

s.hoch();

s.bewegeUm(19);

s.runter();

s.zeichneRechteck(15,10);

s.hoch();

s.bewegeUm(19);

s.runter();

s.zeichneRechteck(15,10);

s.hoch();

s.dreheUm(270);

s.bewegeUm(10);

s.runter();

s.zeichneRechteck(15,10);

s.hoch();

s.dreheUm(270);

s.bewegeUm(19);

s.runter();

s.zeichneRechteck(15,10);

s.hoch();

s.bewegeUm(19);

s.runter();

s.zeichneRechteck(15,10);

s.hoch();

s.bewegeUm(19);

s.runter();

s.dreheUm(90);

s.hoch();

s.bewegeUm(35);

s.dreheUm(90);

s.bewegeUm(11);

s.runter();

s.zeichneKreis(5);

s.hoch();

s.bewegeUm(50);

s.runter();

s.zeichneKreis(5);

s.hoch();

s.bewegeUm(15);

s.dreheUm(90);

s.bewegeUm(10);

s.runter();

s.dreheUm(270);

s.bewegeUm(5);

s.dreheUm(90);

s.bewegeUm(3);

s.dreheUm(180);

s.bewegeUm(6);

s.dreheUm(180);

s.bewegeUm(3);

s.hoch();

}

}

Güterwagen:

import sum.kern.\*;

public class Gueterwagen extends Waggon

{

// Bezugsobjekte

Stift s;

// Attribute

int zH,zV;

// Konstruktor

public Gueterwagen(int pH, int pV)

{

super(pH,pV);

zH = pH;

zV = pV;

s = new Stift();

super.setzeLaenge(110);

}

protected void zeichne()

{

s.bewegeBis(zH, zV);

s.runter();

s.dreheBis(90);

s.bewegeUm(3);

s.dreheUm(180);

s.bewegeUm(6);

s.dreheUm(180);

s.bewegeUm(3);

s.dreheUm(270);

s.bewegeUm(5);

s.hoch();

s.dreheUm(90);

s.bewegeUm(10);

s.zeichneRechteck(100,15);

s.bewegeUm(30);

s.dreheUm(-90);

s.bewegeUm(1);

s.zeichneRechteck(5,30);

s.bewegeUm(93);

s.zeichneRechteck(5,30);

s.bewegeUm(-4);

s.dreheUm(270);

s.bewegeUm(50);

s.zeichneKreis(5);

s.dreheUm(90);

s.bewegeUm(-80);

s.zeichneKreis(5);

s.bewegeUm(90);

s.dreheUm(90);

s.bewegeUm(10);

s.runter();

s.dreheUm(270);

s.bewegeUm(5);

s.dreheUm(90);

s.bewegeUm(3);

s.dreheUm(180);

s.bewegeUm(6);

s.dreheUm(180);

s.bewegeUm(3);

s.hoch();

}

}

Übung 9.4

Der Waggon

import sum.kern.\*;

public abstract class Waggon

{

//Bezugsobjekte

Stift hatStift;

//Attribute

int zLaenge;

public Waggon(int pH, int pV)

{

}

protected abstract void zeichne();

protected Stift stift()

{

Stift stift;

return stift = new Stift();

}

public int laenge()

{

return zLaenge;

}

protected void setzeLaenge(int pLaenge)

{

}

public void gibFrei()

{

}

}

Übung 9.5

Das Zug/Hauptprogramm

import sum.kern.\*;

public class Zugprogramm

{

// Objekte

Bildschirm b;

Lokomotive l;

Gueterwagen g;

Personenwagen p;

// Konstruktor

public Zugprogramm()

{

b = new Bildschirm(600,600);

l = new Lokomotive(100,50);

g = new Gueterwagen(190,50);

p = new Personenwagen (300,50);

this.fuehreAus();

}

public void fuehreAus()

{

// Aktionsteil

l.zeichne();

g.zeichne();

p.zeichne();

}

}

**9.3 Verkettung**

Übung 9.6

Die Modifizierten classen

Das Zugprogramm:

import sum.kern.\*;

public class Zugprogramm

{

// Objekte

Bildschirm b;

Maus m;

Lokomotive l;

Personenwagen p;

Gueterwagen g;

// Konstruktor

public Zugprogramm()

{

b = new Bildschirm(600,600);

m = new Maus();

g = new Gueterwagen(510,60, null);

p = new Personenwagen(420,60, g);

l = new Lokomotive(330, 60, p);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

// Aktionsteil

l.zeichne();

p.zeichne();

g.zeichne();

do{

l.bewegeUm();

}while(true);

}

}

Der Waggon:

import sum.kern.\*;

public abstract class Waggon

{

//Bezugsobjekte

Stift s;

Waggon kenntWaggon;

//Attribute

int zLaenge;

public Waggon(int pH, int pV, Waggon pWaggon)

{

kenntWaggon = pWaggon;

s = new Stift();

s.bewegeBis(pH,pV);

}

protected abstract void zeichne();

protected Stift stift()

{

return s;

}

public int laenge()

{

return zLaenge;

}

protected void setzeLaenge(int pLaenge)

{

zLaenge = pLaenge;

}

public void gibFrei()

{

s.gibFrei();

}

public void loesche()

{

s.radiere();

this.zeichne();

s.normal();

}

public void bewegeUm()

{

this.loesche();

s.dreheBis(180);

s.bewegeUm(0.1);

this.zeichne();

if(kenntWaggon!=null)

kenntWaggon.bewegeUm();

}

}

Die Lokomotieve:

import sum.kern.\*;

public class Lokomotive extends Waggon

{

// Bezugsobjekte

// Attribute

// Konstruktor

public Lokomotive(int pH, int pV, Waggon pWaggon)

{

super(pH, pV, pWaggon);

super.setzeLaenge(90);

}

// Dienste

protected void zeichne()

{

//Startposition merken

double lH = this.stift().hPosition();

double lV = this.stift().vPosition();

// Puffer links

this.stift().runter();

this.stift().dreheBis(90); //oben

this.stift().bewegeUm(3);

this.stift().dreheUm(180); //unten

this.stift().bewegeUm(6);

this.stift().dreheUm(180); //oben

this.stift().bewegeUm(3);

this.stift().dreheUm(270); //rechts

this.stift().bewegeUm(5);

// Vorderteil

this.stift().hoch();

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(20);

this.stift().zeichneRechteck(50,25);

// Schornstein

this.stift().bewegeUm(15);

this.stift().dreheUm(-90); //rechts

this.stift().bewegeUm(10);

this.stift().zeichneRechteck(5,15);

// Fuehrerhaus

this.stift().bewegeUm(40);

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(10);

this.stift().zeichneRechteck(30,50);

// Fenster

this.stift().bewegeUm(-5);

this.stift().dreheUm(-90); //rechts

this.stift().bewegeUm(5);

this.stift().zeichneRechteck(20,20);

// Raeder

this.stift().dreheUm(-90); //unten

this.stift().bewegeUm(50);

this.stift().dreheUm(90); //rechts

this.stift().bewegeUm(15);

this.stift().zeichneKreis(5);

this.stift().bewegeUm(-60);

this.stift().zeichneKreis(5);

// Puffer rechts

this.stift().bewegeUm(70);

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(10);

this.stift().runter();

this.stift().dreheUm(270); //rechts

this.stift().bewegeUm(5);

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(3);

this.stift().dreheUm(180); //unten

this.stift().bewegeUm(6);

this.stift().dreheUm(180); //oben

this.stift().bewegeUm(3);

// Zurueck zur Ausgangsposition

this.stift().hoch();

this.stift().bewegeBis(lH,lV);

this.stift().dreheBis(0);

}

}

Der Güterwagen:

import sum.kern.\*;

public class Gueterwagen extends Waggon

{

// Bezugsobjekte

// Attribute

// Konstruktor

public Gueterwagen(int pH, int pV, Waggon pWaggon)

{

super(pH, pV, pWaggon);

super.setzeLaenge(110);

}

// Dienste

protected void zeichne()

{

//Startposition merken

double lH = this.stift().hPosition();

double lV = this.stift().vPosition();

//Puffer links

this.stift().dreheBis(90); //oben

this.stift().bewegeUm(3);

this.stift().dreheUm(180); //unten

this.stift().bewegeUm(6);

this.stift().dreheUm(180); //oben

this.stift().bewegeUm(3);

this.stift().dreheUm(270); //rechts

this.stift().bewegeUm(5);

//Boden

this.stift().hoch();

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(10);

this.stift().zeichneRechteck(100,15);

//Wand links

this.stift().bewegeUm(30);

this.stift().dreheUm(-90); //rechts

this.stift().bewegeUm(1);

this.stift().zeichneRechteck(5,30);

//Wand rechts

this.stift().bewegeUm(93);

this.stift().zeichneRechteck(5,30);

//Raeder

this.stift().bewegeUm(-4);

this.stift().dreheUm(270); //unten

this.stift().bewegeUm(50);

this.stift().zeichneKreis(5);

this.stift().dreheUm(90); //rechts

this.stift().bewegeUm(-80);

this.stift().zeichneKreis(5);

this.stift().bewegeUm(90);

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(10);

//Puffer rechts

this.stift().runter();

this.stift().dreheUm(270); //rechts

this.stift().bewegeUm(5);

this.stift().dreheUm(90); //oben

this.stift().bewegeUm(3);

this.stift().dreheUm(180); //unten

this.stift().bewegeUm(6);

this.stift().dreheUm(180); //oben

this.stift().bewegeUm(3);

// Zurueck zur Ausgangsposition

this.stift().hoch();

this.stift().bewegeBis(lH,lV);

this.stift().dreheBis(0);

}

}

Der Personenwagen

import sum.kern.\*;

public class Personenwagen extends Waggon

{

// Bezugsobjekte

// Attribute

// Konstruktor

public Personenwagen(int pH, int pV, Waggon pWaggon)

{

super(pH, pV, pWaggon);

super.setzeLaenge(90);

}

// Dienste

protected void zeichne()

{

//Startposition merken

double lH = this.stift().hPosition();

double lV = this.stift().vPosition();

// Schuelerloesung

this.stift().runter();

this.stift().dreheBis(90);

this.stift().bewegeUm(3);

this.stift().dreheUm(180);

this.stift().bewegeUm(6);

this.stift().dreheUm(180);

this.stift().bewegeUm(3);

this.stift().dreheUm(270);

this.stift().bewegeUm(5);

this.stift().dreheUm(90);

this.stift().bewegeUm(45);

this.stift().zeichneRechteck(80,50);

this.stift().hoch();

this.stift().dreheUm(270);

this.stift().bewegeUm(4);

this.stift().dreheUm(270);

this.stift().bewegeUm(10);

this.stift().runter();

this.stift().zeichneRechteck(15,35);

this.stift().hoch();

this.stift().dreheUm(90);

this.stift().bewegeUm(19);

this.stift().runter();

this.stift().zeichneRechteck(15,10);

this.stift().hoch();

this.stift().bewegeUm(19);

this.stift().runter();

this.stift().zeichneRechteck(15,10);

this.stift().hoch();

this.stift().bewegeUm(19);

this.stift().runter();

this.stift().zeichneRechteck(15,10);

this.stift().hoch();

this.stift().dreheUm(270);

this.stift().bewegeUm(10);

this.stift().runter();

this.stift().zeichneRechteck(15,10);

this.stift().hoch();

this.stift().dreheUm(270);

this.stift().bewegeUm(19);

this.stift().runter();

this.stift().zeichneRechteck(15,10);

this.stift().hoch();

this.stift().bewegeUm(19);

this.stift().runter();

this.stift().zeichneRechteck(15,10);

this.stift().hoch();

this.stift().bewegeUm(19);

this.stift().runter();

this.stift().dreheUm(90);

this.stift().hoch();

this.stift().bewegeUm(35);

this.stift().dreheUm(90);

this.stift().bewegeUm(11);

this.stift().runter();

this.stift().zeichneKreis(5);

this.stift().hoch();

this.stift().bewegeUm(50);

this.stift().runter();

this.stift().zeichneKreis(5);

this.stift().hoch();

this.stift().bewegeUm(15);

this.stift().dreheUm(90);

this.stift().bewegeUm(10);

this.stift().runter();

this.stift().dreheUm(270);

this.stift().bewegeUm(5);

this.stift().dreheUm(90);

this.stift().bewegeUm(3);

this.stift().dreheUm(180);

this.stift().bewegeUm(6);

this.stift().dreheUm(180);

this.stift().bewegeUm(3);

this.stift().hoch();

// Zurueck zur Ausgangsposition

this.stift().hoch();

this.stift().bewegeBis(lH,lV);

this.stift().dreheBis(0);

}

}

Übung 9.7

(1)Variante FEHLER SIND EXISTENT (Variante 2 ist fehlerfrei)

Das Zugprogramm:  
import sum.kern.\*;

public class Zugprogramm

{

// Objekte

Bildschirm b;

Maus m;

Lokomotive l;

Personenwagen p,nP;

Gueterwagen g;

// Konstruktor

public Zugprogramm()

{

b = new Bildschirm(600,600);

m = new Maus();

g = new Gueterwagen(510,60, null);

p = new Personenwagen(420,60, g);

l = new Lokomotive(330, 60, p);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

// Aktionsteil

nP = new Personenwagen(-100,-100, null);

do{

if(m.istGedrueckt())

{

l.koppleAn(nP);

}

l.bewegeUm(0.1);

}while(true);

}

}

Der Waggon;

import sum.kern.\*;

public abstract class Waggon

{

//Bezugsobjekte

Stift s;

Waggon kenntWaggon;

//Attribute

int zLaenge;

public Waggon(int pH, int pV, Waggon pWaggon)

{

kenntWaggon = pWaggon;

s = new Stift();

s.bewegeBis(pH,pV);

}

public Waggon nachfolger()

{

return kenntWaggon;

}

protected abstract void zeichne();

protected Stift stift()

{

return s;

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public int laenge()

{

return zLaenge;

}

protected void setzeLaenge(int pLaenge)

{

zLaenge = pLaenge;

}

public void gibFrei()

{

s.gibFrei();

}

public void loesche()

{

s.radiere();

this.zeichne();

s.normal();

}

public void bewegeUm(double Geschwindigkeit)

{

this.loesche();

s.bewegeUm(- Geschwindigkeit);

this.zeichne();

if(kenntWaggon!=null)

{

kenntWaggon.bewegeUm(Geschwindigkeit);

}

}

public void bewegeBis(double H,double V)

{

s.bewegeBis(H, V);

this.zeichne();

}

public void koppleAn(Waggon pNeuerWaggon)

{

if(kenntWaggon != null)

{

kenntWaggon.koppleAn(pNeuerWaggon);

}else

{

pNeuerWaggon.bewegeBis(s.hPosition()+zLaenge,s.vPosition());

kenntWaggon = pNeuerWaggon;

}

}

}

(2)Variante --> Ohne Fehler ╰(\*°▽°\*)╯

Das Zugprogramm:  
import sum.kern.\*;

public class Zugprogramm

{

// Objekte

Bildschirm b;

Maus m;

Tastatur t;

Lokomotive l;

Personenwagen p,lNeuerWaggon;

Gueterwagen g;

Waggon kenntLetzterWaggon;

// Konstruktor

public Zugprogramm()

{

b = new Bildschirm(600,600);

m = new Maus();

t = new Tastatur();

g = new Gueterwagen(510,60, null);

p = new Personenwagen(420,60, g);

l = new Lokomotive(330, 60, p);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

// Aktionsteil

lNeuerWaggon = new Personenwagen(-100,-100, null);

do{

if(t.wurdeGedrueckt())

{

lNeuerWaggon.bewegeBis(g.hPosition() + g.laenge(), g.vPosition());

g.setzeNachfolger(lNeuerWaggon);

}

while(!m.istGedrueckt())

{

l.bewegeUm(0.1);

}

}while(true);

}

}

Der Waggon:

import sum.kern.\*;

public abstract class Waggon

{

//Bezugsobjekte

Stift s;

Waggon kenntWaggon;

//Attribute

int zLaenge;

public Waggon(int pH, int pV, Waggon pWaggon)

{

kenntWaggon = pWaggon;

s = new Stift();

s.bewegeBis(pH,pV);

}

public Waggon nachfolger()

{

return kenntWaggon;

}

protected abstract void zeichne();

protected Stift stift()

{

return s;

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public int laenge()

{

return zLaenge;

}

protected void setzeLaenge(int pLaenge)

{

zLaenge = pLaenge;

}

public void gibFrei()

{

s.gibFrei();

}

public void loesche()

{

s.radiere();

this.zeichne();

s.normal();

}

public void bewegeUm(double Geschwindigkeit)

{

this.loesche();

s.bewegeUm(- Geschwindigkeit);

this.zeichne();

if(kenntWaggon!=null)

{

kenntWaggon.bewegeUm(Geschwindigkeit);

}

}

public void bewegeBis(double H,double V)

{

s.bewegeBis(H, V);

this.zeichne();

}

public void setzeNachfolger(Waggon pWaggon)

{

kenntWaggon = pWaggon;

}

}

(3)Variante FEHLER SIND EXISTENT (Variante 2 ist fehlerfrei)

Das Zugprogramm:

import sum.kern.\*;

public class Zugprogramm

{

// Objekte

Bildschirm b;

Maus m;

Tastatur t;

Lokomotive l;

Personenwagen p;

Gueterwagen g;

Waggon kenntLetzterWaggon,lWaggon;

// Konstruktor

public Zugprogramm()

{

b = new Bildschirm(600,600);

m = new Maus();

t = new Tastatur();

g = new Gueterwagen(510,60, null);

p = new Personenwagen(420,60, g);

l = new Lokomotive(330, 60, p);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

// Aktionsteil

lWaggon = new Personenwagen(-100,-100, null);

do{

if(t.wurdeGedrueckt())

{

lWaggon = p;

while (lWaggon.nachfolger() != null)

{

lWaggon = lWaggon.nachfolger();

kenntLetzterWaggon.bewegeBis(lWaggon.hPosition() + lWaggon.laenge(),

lWaggon.vPosition());

lWaggon.setzeNachfolger(lWaggon);

}

}

while(!m.istGedrueckt())

{

l.bewegeUm(0.1);

}

}while(true);

}

}

Der Waggon:

import sum.kern.\*;

public abstract class Waggon

{

//Bezugsobjekte

Stift s;

Waggon kenntWaggon;

//Attribute

int zLaenge;

public Waggon(int pH, int pV, Waggon pWaggon)

{

kenntWaggon = pWaggon;

s = new Stift();

s.bewegeBis(pH,pV);

}

public Waggon nachfolger()

{

return kenntWaggon;

}

protected abstract void zeichne();

protected Stift stift()

{

return s;

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public int laenge()

{

return zLaenge;

}

protected void setzeLaenge(int pLaenge)

{

zLaenge = pLaenge;

}

public void gibFrei()

{

s.gibFrei();

}

public void loesche()

{

s.radiere();

this.zeichne();

s.normal();

}

public void bewegeUm(double Geschwindigkeit)

{

this.loesche();

s.bewegeUm(- Geschwindigkeit);

this.zeichne();

if(kenntWaggon!=null)

{

kenntWaggon.bewegeUm(Geschwindigkeit);

}

}

public void bewegeBis(double H,double V)

{

s.bewegeBis(H, V);

this.zeichne();

}

public void setzeNachfolger(Waggon pWaggon)

{

kenntWaggon = pWaggon;

}

}

Übung 9.8

Basierend auf v2:

Das Zugprogramm:

import sum.kern.\*;

public class Zugprogramm

{

// Objekte

Bildschirm b;

Maus m;

Tastatur t;

Lokomotive l;

Personenwagen p,lNeuerWaggon;

Gueterwagen g;

Waggon kenntLetzterWaggon;

// Konstruktor

public Zugprogramm()

{

b = new Bildschirm(600,600);

m = new Maus();

t = new Tastatur();

g = new Gueterwagen(510,60, null);

p = new Personenwagen(420,60, g);

l = new Lokomotive(330, 60, p);

this.fuehreAus();

}

// Dienste

public void fuehreAus()

{

// Aktionsteil

lNeuerWaggon = new Personenwagen(-100,-100, null);

double Geschwindigkeit = 0.1;

do{

if(t.wurdeGedrueckt())

{

lNeuerWaggon.bewegeBis(g.hPosition() + g.laenge(), g.vPosition());

g.setzeNachfolger(lNeuerWaggon);

}

while(!m.istGedrueckt())

{

if (t.wurdeGedrueckt())

{ switch (t.zeichen())

{ case '+':Geschwindigkeit = Geschwindigkeit + 0.1 ; break;

case '-':Geschwindigkeit = Geschwindigkeit - 0.1 ; break;

//WEITERE FARBEN HIER

default: Geschwindigkeit = 0.1 ; break;

}

t.weiter();

}

l.bewegeUm(Geschwindigkeit);

this.testeRand();

}

}while(true);

}

public void testeRand()

{

//Lok

if(l.hPosition() <= - l.laenge() )

{

l.bewegeBis(b.breite(), l.vPosition());

}

if(l.hPosition() >= b.breite() + l.laenge() )

{

l.bewegeBis(-l.laenge(), l.vPosition());

}

//gue

if(g.hPosition() <= - g.laenge() )

{

g.bewegeBis(b.breite(), g.vPosition());

}

if(g.hPosition() >= b.breite() + g.laenge() )

{

g.bewegeBis(-g.laenge(), g.vPosition());

}

//per

if(p.hPosition() <= - p.laenge() )

{

p.bewegeBis(b.breite(), p.vPosition());

}

if(p.hPosition() >= b.breite() + p.laenge() )

{

p.bewegeBis(-p.laenge(), p.vPosition());

}

}

}

Der Waggon:

import sum.kern.\*;

public abstract class Waggon

{

//Bezugsobjekte

Stift s;

Waggon kenntWaggon;

//Attribute

int zLaenge;

public Waggon(int pH, int pV, Waggon pWaggon)

{

kenntWaggon = pWaggon;

s = new Stift();

s.bewegeBis(pH,pV);

}

public Waggon nachfolger()

{

return kenntWaggon;

}

protected abstract void zeichne();

protected Stift stift()

{

return s;

}

public double hPosition()

{

return s.hPosition();

}

public double vPosition()

{

return s.vPosition();

}

public int laenge()

{

return zLaenge;

}

protected void setzeLaenge(int pLaenge)

{

zLaenge = pLaenge;

}

public void gibFrei()

{

s.gibFrei();

}

public void loesche()

{

s.radiere();

this.zeichne();

s.normal();

}

public void bewegeUm(double Geschwindigkeit)

{

this.loesche();

s.bewegeUm(- Geschwindigkeit);

this.zeichne();

if(kenntWaggon!=null)

{

kenntWaggon.bewegeUm(Geschwindigkeit);

}

}

public void bewegeBis(double H,double V)

{

s.bewegeBis(H, V);

this.zeichne();

}

public void setzeNachfolger(Waggon pWaggon)

{

kenntWaggon = pWaggon;

}

}

Übung 9.9

Vielleicht in v2 hier nicht :) ist auch erstmal nicht wichtig.

Übung 9.10

**9.4 Zusammenfassung**

**Kapitel 10 Ereignisorientierung**

**10.1 Anwendung**

**10.2 Problemzerlegung**

**10.3 Ereignisanwendung**

**10.4 Leerlaufereignis**

**10.5 Freihandzeichnen ereignisorientiert**

**10.6 Hauptprogramm**

**10.7 Zusammenfassung**

**Kapitel 11 Ereignisbearbeiter**

**11.1 Knöpfe**

**11.2 Aktionen**

**11.3 Ereignisbearbeiter**

**11.4 Ereignisverteiler**

**11.5 Debugger**

**11.6 Sichtbarkeit von Komponenten**

**11.7 Zusammenfassung**

**Kapitel 12 Phasen eines Softwareprojekts**

**12.3 Entwurfsphase**

Übung 12.1



Die Taskleiste beinhaltet ein Icon Image die Funktion minimize maximise und Exit on Close.

Oben links haben wir zum einen das Guthaben des Nutzers als auch den Gewinn des Nutzers.

In der mitte haben wir drei Slots und einen Hebel. Dieser Hebel könnte als Textured JButton konstruiert werden und auf den ersten druck die slots starten und auf den zweiten druck die slots stoppen. Wir könnten allerdings auch realistischer werden und auf den ersten druck einen timer starten und dann warten bis die slots ausgerollt haben.

Ganz unten haben wir noch einen Münzeinwurf.

Übung 12.2

**12.5 Modultest**

Übung 12.3

**12.6 Zusammenbau der Module**

Übung 12.4

**12.7 Ereignisbibliothek**

Übung 12.5

Übung 12.6

**Kapitel 13 Model View Controller**

**13.2 SuM-Programmgenerator**

Übung 13.1

**13.3 Datenmodellierung**

Übung 13.2

Übung 13.3

Übung 13.4

Übung 13.5

Übung 13.6

Übung 13.7

**13.4 Programmsteuerung**

Übung 13.8

**13.6 MVC-Modell**

Übung 13.9

Übung 13.10

Übung 13.11

**13.7 Zusammenfassung**

Du kannst dich so glücklich schätzen das du hiermit durch bist dieses Buch ist echt nicht sonderlich gut ,desweiteren haben sowohl der code editor als auch das gui programm viel zu wünschen über. Kann man nicht mittlerweile erwarten das man ein programm maximiren kann?! Selbst der windows standart text editor kann das.

Wenn du Spaß haben willst setze dir ein Ziel installiere VS Code von Microsoft das Java Ide und alle Java extensions,dann schnapp dir Stackoverflow und Youtube und du wirst überascht sein wischnell du mehr lernst als in diesem Buch. :)

**Anhang**

Die besten Sachen die man wärend des Informatikunterricht machen kann wenn alle Sachen erledigt sind:

1.Anderen helfen

2.Sein eigenes Projekt erweitern

3.Nette gespräche mit anderen fertigen Schülern führen

4.Auf den folgenen Seiten spiele spielen:

- <https://neal.fun/>

- <https://www.newgrounds.com/>

oder für Leute mit weniger Anspruch:

- <https://www.spiele-kostenlos-online.de/>

- <https://www.spielaffe.de/>