Informatik

Quelltexte

Kasten:

```
   public class KASTEN
{
   private int links;
   private int oben;
   private int hoehe;
   private int breite;
   public KASTEN()
   {
      links = 50;
      oben = 100;
      hoehe = 100;
      breite = 200;
   }
   public KASTEN(int linksStart, int obenStart, int hoeheStart, int
breiteStart)
   {
      links = linksStart;
      oben = obenStart;
      hoehe = hoeheStart;
      breite = breiteStart;
   }
   public void zeichne()
   {

ZEICHENFENSTER.gibFenster().zeichneRechteck(links,oben,hoehe,breite);
   }
   public void setzeLinks(int linksNeu)
   {
      links = linksNeu;
   }
   public void setzeHoehe(int hoeheNeu)
   {
      hoehe = hoeheNeu;
   }
```

```
}
```

Vollkreis:

```
public class VOLLKREIS
{
   private int radius;
   private int farbnr;
   private int xmitte;
   private int ymitte;
   public VOLLKREIS(int xmitteNeu, int ymitteNeu, int radiusNeu, int
farbnrNeu)
   {
      xmitte = xmitteNeu;
      ymitte = ymitteNeu;
      radius = radiusNeu;
      farbnr = farbnrNeu;
   }
   public void zeichne()
   {
       ZEICHENFENSTER.gibFenster().fuelleKreis(xmitte,ymitte,radius,farbnr);
   }
   public void setzeFarbe(int fNeu)
   {
      farbnr = fNeu;
   }
   public void setzeMitte(int xNeu, int yNeu)
   {
      xmitte = xNeu;
      ymitte = yNeu;
   }
   public void setzeRadius(int radiusNeu)
   {
      radius = radiusNeu;
   }
   public void verschiebe(int xplus, int yplus)
   {
      xmitte = xmitte + xplus;
      ymitte = ymitte + yplus;
   }
}
```

Spielwalze:

```
public class SPIELWALZE
{
```

```
    private KASTEN rahmen;
    private VOLLKREIS lampe;
    private int farbNr;
    private int hoehe;
    private int breite;
    private int links;
    private int oben;
    public SPIELWALZE (int farbNrStart, int breiteStart, int linksStart,
int obenStart)
    {
        rahmen = new KASTEN (linksStart,obenStart,breiteStart,breiteStart);
        lampe = new VOLLKREIS (linksStart + breiteStart/2,obenStart +
        breiteStart/2,breiteStart/3,farbNrStart);
        farbNr = farbNrStart
    }
    public void zeichne()
    {
        rahmen.zeichne();
        lampe.zeichne();
    }
    public void faerbeUm (int farbnrNeu)
    {
        farbNr = farbnrNeu;
        lampe.setzeFarbe(farbNr); //Methodenaufruf
        zeichne ();

    }

}
```

Spielautomat:

```
import java.util.Random; //Mit der Anweisung Import Paketname.Klassenname kann man eine
//bereits vorhandene Java-Klasse verwenden
public class SPIELAUTOMAT
{
    private SPIELWALZE walze1;
    private SPIELWALZE walze2;
    private SPIELWALZE walze3;
    private Random zufall;
    private int z1;
    private int z2;
    private int z3;

    public SPIELAUTOMAT()
    {
```

```
        zufall = new Random();
        z1 = 0;
        z2 = 0;
        z3 = 0;
        walze1 = new SPIELWALZE(z1,200,0,100);
        walze2 = new SPIELWALZE(z2,200,200,100);
        walze3 = new SPIELWALZE(z3,200,400,100);
    }
    public void zeichne()
    {
        walze1.zeichne();
        walze2.zeichne();
        walze3.zeichne();
    }
    public void setzeFarbmuster(int z1Neu, int z2Neu, int z3Neu)
    {
        z1 = z1Neu;
        z2 = z2Neu;
        z3 = z3Neu;
        walze1.faerbeUm(z1); //Methodenaufruf
        walze2.faerbeUm(z2);
        walze3.faerbeUm(z3);
    }
    public void spiele()
    {
setzeFarbmuster(zufall.nextInt(9),zufall.nextInt(9),zufall.nextInt(9));
    }
    public void schreibeWerte()
    {
        System.out.println("Farbnummer1:"+z1+" "+"Farbnummer2:"+z2+"
        "+"Farbnummer3:"+z3);
    }
}
```

Algorithmische Strukturen

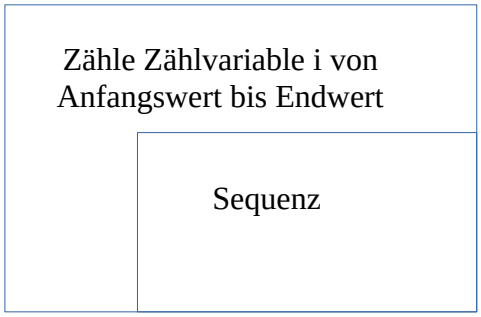
Bedingte Wiederholung mit Endbedingung

Bedingte Wiederholung mit Anfangsbedingung

21.01.2016

Bedingte Wiederholung mit Zähler

Struktogramm:



Zu Beginn hat die Zählvariable einen bestimmten Anfangswert. Nach jedem Durchlauf durch die Sequenz wird der Wert der Variablen um eins erhöht bzw. verringert. Beim letzten Durchlauf hat die Zählvariable den Endwert.

Java-Syntax:

```
for (int i = Anfangswert; i <= Endwert; i++) // i++ - Abkürzung für i = i + 1
        {Sequenz;}
for (int i = Anfangswert; i >= Endwert; i--) // i-- - Abkürzung für i = i + 1
        {Sequenz;}
```

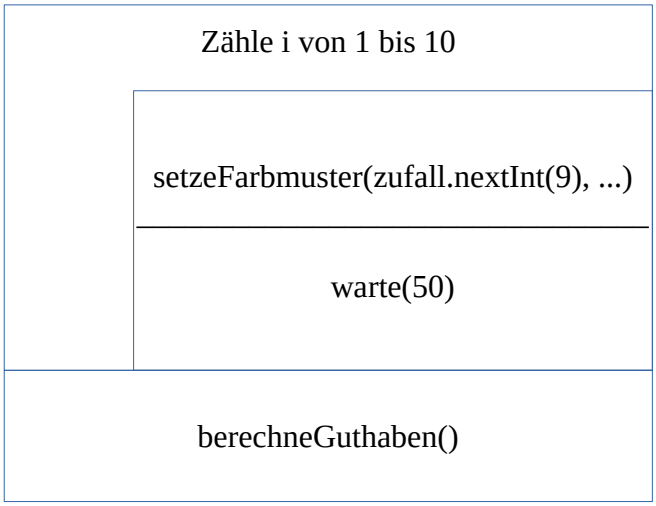
i++ i--

Der Ausdruck „i++“ sorgt dafür, dass, nach dem Durchlaufen der Sequenz, der Wert der Zählvariablen um eins erhöht wird. Bei „i--“ wird er um eins verringert.

Motivation:

Der Spielautomat soll zunächst mehrmals die Farben der Walzen wechseln und erst dann seinen Endzustand erreichen.

Konkretes Struktogramm:



Java-Syntax:

```
public void laufe()
{       for(int i = 1; i <= 10; i++)
            {       setzeFarbmuster(zufall.nextInt(9), …);
                    ZEICHENFENSTER.gibFenster().warte(50);
            }
        berechneGuthaben();
}
```

18.02.2016

Aufgaben

- Schreibe eine Methode, die die Summe aller natürlichen Zahlen bis n angibt.

```
Public int berechneSumme(int n)
{       int summe=0;
        for(int i=1; i<=n; i++)
        {       summe=summe+i;
        }
        return summe;
}
```

oder

```
public int berechneSumme(int n)
{       int a=1;
```

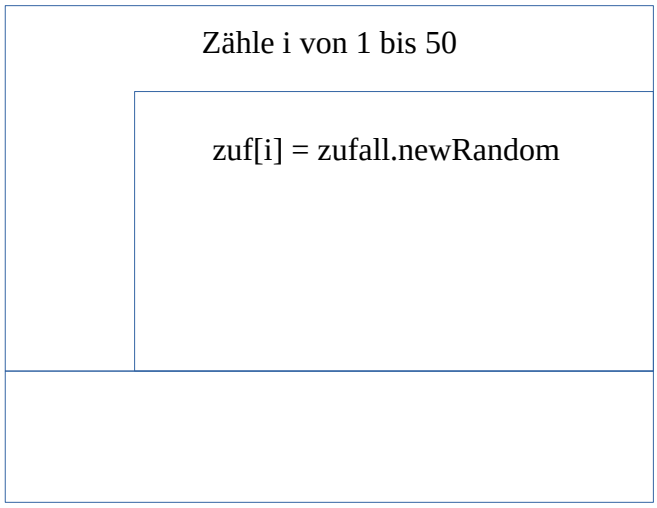
Oder:

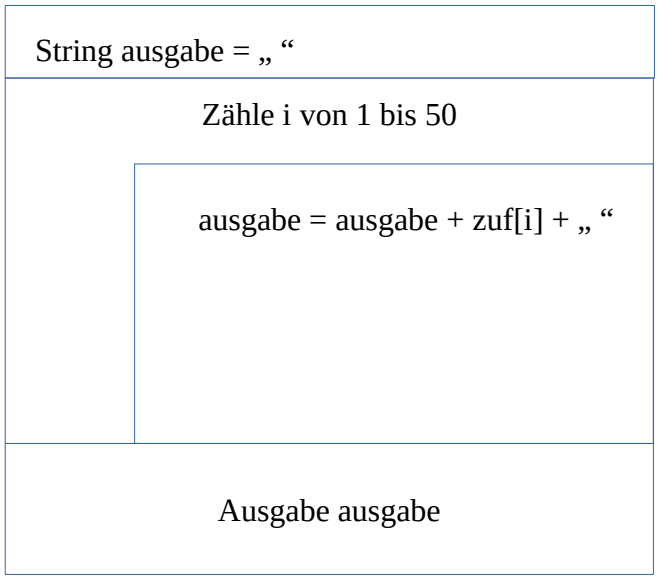
```
Public int berechneSumme(int n)
{       int summe = (n*n+1)/2;
        return summe;
}
```

```
        int summe=0;
        do
        {       summe=summe+a;
                a++;
        }
        while(a<n)
        return summe;
}
```

Aufgaben – AB 18.02.2016-01

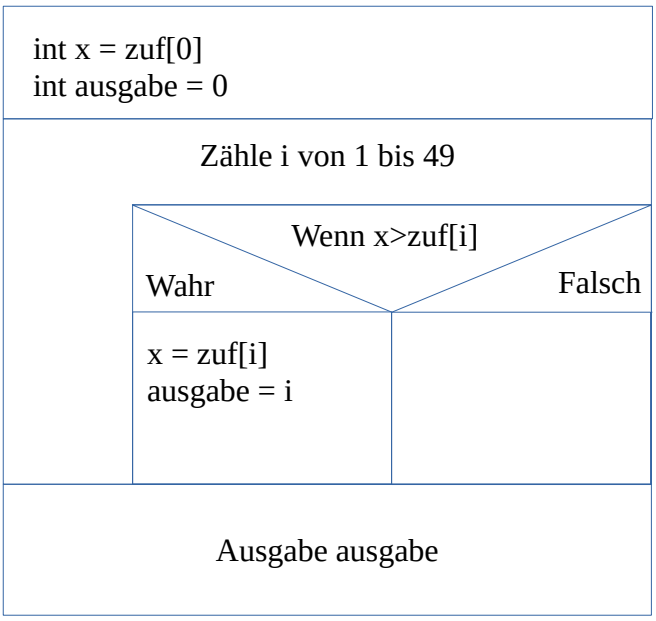
```
public class SORTIEREN
{private int[] zuf;
```

1. 

2. 

3. …
4. …

5.



gibMinStelle()

public int gibMinStelle()

{if …

Elemente Graphischer Oberflächen

Graphische Oberflächen sind aus Komponenten wie Schaltknopf, Textanzeige, Texteingabe aufgebaut. Alle diese Komponenten werden in das Zeichenfenster eingefügt. Zu jeder Komponente stellt das Paket javax.swing eine geeignete Klasse zur Verfügung.

Bsp.: JButton (Schaltknopf)

JLabel (Textanzeige)

Die Klasse Zeichenfenster hat eine Methode zum einfügen graphischer Komponenten:

komponenteHinzufuegen(JKomponent element, string Position)

… fügt die im Parameter übergebene graphische Komponente in einem Bereich rechts oder unter der eigentlichen Zeichenfläche ein. (Position = rechts bzw. unten)

Java-Syntax:

```
Import javax.swing.*; //“*“ importiert alle Klassen aus dem Paket
public class SPIELAUTOMAT
{…
    private JButton knopf;
    private JLabel textanzeige;
    public SPIELAUTOMAT
    {       …
```

```
                knopf = new JButton(„Spielen“);
                textanzeige = new JLabel(„ “);
                ZEICHENFENSTER.gibFenster().komponenteHinzufuegen(knopf, „unten“)
                ZEICHENFENSTER.gibFenster().komponenteHinzufuegen(textanzeige,
„unten“;
        }
    }
```

Projekt Petrus

Im neuen Projekt sollen unterschiedliche Niederschlagsarten am Bildschirm simuliert werden. Zunächst wird nur eine Niederschlagsart durch fallende Tropfen dargestellt.

PETRUS ^{1}– $^{\text{steuert}>}$ → ^{1} WOLKE ^{1}– $^{\text{verwaltet}>}$ → ^{n} NIEDERSCHLAG

Die Klasse Niederschlag

Die Regentropfen werden als Kreise dargestellt, die sich mit konstanter Geschwindigkeit nach unten bewegen.

NIEDERSCHLAG
private double x private double y private double vx private double vy
public NIEDERSCHLAG(double xStart, double yStart) public void bewege(double zeit) public void zeichne()

Java-Syntax:

```
Public class NIEDERSCHLAG
{       …
    public NIEDERSCHLAG(double xStart, double yStart)
    {       x = xStart;
            y = yStart;
            vx = 1;
            vy = 10;
    }
    public void zeichne()
    {       ZEICHENFENSTER.gibFenster().zeichneKreis((int)x, (int)y, 10)
```

```
            // „(int)“ = Typecast(Typumwandlung)
        }
}
```