CARL VON OSSIETZKY universität OLDENBURG

Teil

Objektorientierte Programmierung

Unterrichtseinheit 22

Nutzung von Klassen

Dr. Dietrich Boles

Gliederung

- Klasse
- Objektvariablen
- Objekterzeugung
- Objektvariablen und Objekterzeugung
- Aufruf von Instanz-Methoden
- Beispiel Graphik
- Beispiel GUI-Programmierung
- Zusammenfassung

Klasse

```
<klasse>       ::= ["public"] "class" <bezeichner> "{"
                   { <konstruktor-def> |
                     <methoden-def> | ...
                   }
                   "}"

<methoden-def>    ::= <funktionen-def> (ohne static)

<konstruktor-def> ::= <methoden-def> (ohne Funktionstyp)
```

Klasse / Protokoll

```
class Hamster {     // Klasse (Typ)

  // Konstruktoren
  Hamster(int r, int s, int b, int k) { . . . }

  // (Instanz-) Methoden
  void vor() { . . . }
  void linksUm() { . . . }
  void gib() { . . . }
  void nimm() { . . . }

  boolean vornFrei() { . . . }
  boolean maulLeer() { . . . }
  boolean kornDa() { . . . }

  . . .
}
```

Objektvariablen

```
<objectvar-def> ::= <klassen-bezeichner>
                    <bezeichner>
                    { "," <bezeichner } 
                    ";"
```

Anmerkungen:

- Klassenbezeichner muss Name einer gültigen Klasse sein
- Bezeichner sind Objektvariablen ("Namen für Objekte")
- Objektvariablen sind Referenzvariablen
- Objektvariablen speichern Referenzen auf Objekte

Beispiele:

```
Hamster paul;
Hamster willi, maria, heike;
Hamster franz = null;  // expl. Initialisierung
```

Objekterzeugung

```
<object-erz> ::= "new" <bezeichner>
                 "(" { <parameter-liste> } ")"
```

Anmerkungen:

- Bezeichner muss Name einer gültigen Klasse sein
- es muss ein "passender" Konstruktor existieren

Semantik:

- Es wird ein Objekt der entsprechenden Klasse "erzeugt" und ggfl. ein Konstruktor ausgeführt
- reserviert Speicherplatz für Objektattribute auf dem Heap
- Konstruktor: initialisiert Objektattribute
- liefert Adresse (Referenz) auf den Speicherbereich
- Objektzerstörung: -> Garbage Collector

Beispiele:

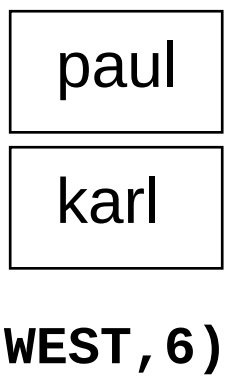
```
new Hamster(2,2,Hamster.NORD,2)
new Hamster(1,2,Hamster.OST,3)
new Hase()                  // Fehler (unbekannte Klasse)
new Hamster(0,0)            // Fehler (ungültiger Konstruktor)
```

Objektvariablen und Objekterzeugung

- häufig erfolgt die Objektvariablendefinition und die Objekterzeugung in einer Anweisung
- Zuweisungen möglich
- Typ der Objektvariablen muss gleich Typ des Objektes sein (Ausnahme: Polymorphie!)

Beispiele:

```
Hamster paul = new Hamster(1,1,Hamster.NORD,1);
Hamster karl = new Hamster(2,2,Hamster.OST,2);
paul = new Hamster(3,4,Hamster.SUED,5);
karl = paul;
```




```
class Tier { . . . }
Tier heike = new Hamster(3,6,Hamster.WEST,6);
  // Fehler!
```

Instanz-Methoden / Aufruf


```
<meth-aufruf> ::= <bezeichner> "." <funktionsaufruf>
```

Anmerkungen:

- Bezeichner muss gültige Objektvariable sein
- Objektvariable muss auf ein Objekt verweisen
- "passende" Funktion muss in der Klasse des Objektes vorhanden sein

Semantik:

- die Funktion ("für das Objekt") wird aufgerufen
- sie "operiert" auf den Attributen des Objektes

Beispiele:

```
Hamster paul = new Hamster(1,2,Hamster.OST,4);
if (paul.vornFrei()) paul.vor();
Hamster karl;
karl.linksUm();    // Laufzeitfehler (kein Objekt)
paul.rechtsUm();   // Fehler (fehlende Funktion/Methode)
paul.vor(3);       // Fehler (fehlende Funktion/Methode)
```

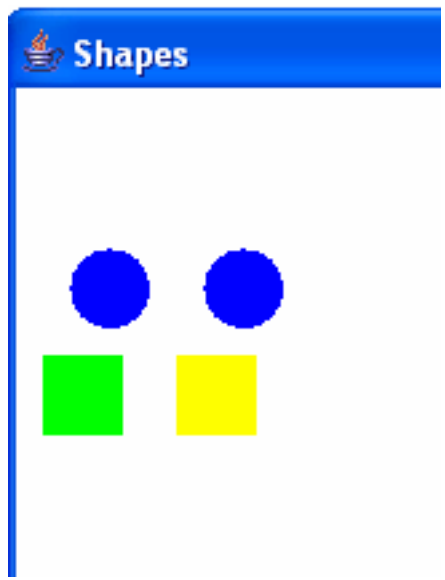
Beispiel Graphik / Protokoll

```
class Circle {
  Circle() { . . . }
  void moveHorizontal(int distance) { . . . }
  void moveVertical(int distance) { . . . }
  void changeSize(int newDiameter) { . . . }
  void changeColor(String newColor) { . . . }
}

class Square {
  Square() { . . . }
  void moveHorizontal(int distance) { . . . }
  void moveVertical(int distance) { . . . }
  void changeSize(int newSize) { . . . }
  void changeColor(String newColor) { . . . }
}
```

Beispiel Graphik / Beispielprogramm

```
class Shapes {
   public static void main(String[] args) {
      Circle c1 = new Circle();
      Circle c2 = new Circle();
      c2.moveHorizontal(50);
      Square s1 = new Square();
      s1.changeColor("yellow");
      s1.moveVertical(50);
      Square s2 = new Square();
      s2.changeColor("green");
      s2.moveVertical(50);
      s2.moveHorizontal(-50);
   }
}
```



Demo

Beispiel GUI-Programmierung / Protokoll (1)

```
class Frame {
  Frame()
  Frame(String title)

  void setTitle(String title)
  String getTitle()
  void setResizable(boolean resizable)
  void setSize(int width, int height)
  void setLocation(int x, int y)
  void setLayout(LayoutManager layoutManager)
  void add(Component comp)
  void setVisible(boolean visible)
}

class GridLayout { // LayoutManager
    GridLayout(int rows, int cols)
}
```

Beispiel GUI-Programmierung / Protokoll (2)

```
class Panel { // Component
  Panel()
  void add(Component comp)
  void setLayout(LayoutManager layoutManager)
}

class Label { // Component
  Label()
  Label(String label)
  void setLabel(String label)
  String getLabel()
}

class Button { // Component
  Button()
  Button(String label)
  void setLabel(String label)
  String getLabel()
}
```

Beispiel GUI-Programmierung / Protokoll (3)

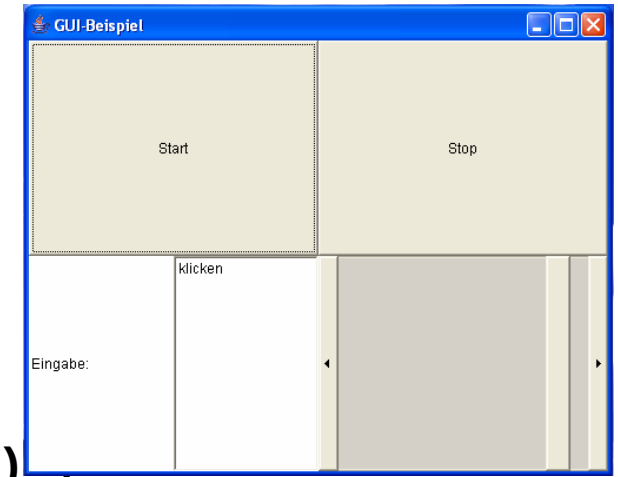
```
class Checkbox { // Component
  Checkbox(String label)
  Checkbox(String label, boolean checked)
  void setLabel(String label)
  boolean getState()
}

class TextField { // Component
  TextField()
  TextField(String text)
  void setText(String text)
  String getText()
}

class Scrollbar { // Component
  Scrollbar()
  void setOrientation(int orientation)
  void setValue(int value)
  int getValue()
}
```

Beispiel GUI-Programmierung / Programm (1)

Demo




```
import java.awt.*;

class GUIExample {

   public static void main(String[] args) {
      Frame fenster = new Frame("GUI-Beispiel");

      fenster.setLocation(10, 20);
      fenster.setSize(500, 400);
      GridLayout fensterLayout = new GridLayout(2, 2);
      fenster.setLayout(fensterLayout);

      Button start = new Button("Start");
      Button stop = new Button("Stop");
      fenster.add(start);
      fenster.add(stop);
```

Beispiel GUI-Programmierung / Programm (2)

```
        Panel p = new Panel();
        GridLayout pLayout = new GridLayout(1, 2);
        p.setLayout(pLayout);
        fenster.add(p);

        Label l = new Label("Eingabe:");
        p.add(l);
        TextField eingabe = new TextField("klicken");
        p.add(eingabe);

        Scrollbar bar = new Scrollbar();
        bar.setOrientation(0);
        bar.setValue(83);
        fenster.add(bar);

        fenster.setVisible(true);
    }

}
```

Zusammenfassung

- Klasse: Bauplan für gleichartige Dinge (Objekte)
- Eine Klasse definiert einen neuen Typ
- Methoden: Eine Klasse definiert u. a. Funktionen (Methoden), die für Objekte der Klasse aufgerufen werden können
- Objekte: Von einer Klasse können Objekte erzeugt werden
- Objektvariablen: Der Methodenaufruf erfolgt via der Punktnotation über Objektvariablen, die Referenzen auf Objekte speichern können
- OO-Programm: Aufruf von Methoden von/für Objekte