Pakete

Jede Klasse ist Teil eines Pakets. Jede Variable und jede Methode ist in einer Klasse deklariert. Ihr vollständiger Name ist

Paketname.Klassenname.Variablenname bzw.

Paketname.Klassenname.Methodenname

david.games.tetris.SoundEffects.play()

Das Paket gibt an, wo der Übersetzer nach der Klasse suchen soll. Deshalb entspricht der Paketname einem Pfad in dem Verzeichnisbaum.

david/games/tetris

Klassenpfad

Ausgangspunkt des Pfades, der durch den Paketnamen angegeben wird, sind die Verzeichnisse, die in der Umgebungsvariable CLASSPATH angegeben sind. In UNIX:

`setenv` CLASSPATH .:de/unido/cs/ls8

Der Übersetzer sucht nach der Methode **play()** in der Datei

de/unido/cs/ls8/david/games/tetris/SoundEffects.class

oder (:) im aktuellen Verzeichnis (.)

Auch beim Aufruf wird in dieser Weise gesucht (und gefunden).

Übersetzungseinheit

besteht aus mindestens einer der Deklarationen:

die Paketdeklaration , die einen Namen für eine Menge von Klassendeklarationen festlegt,

die Importdeklaration , die Deklarationen aus einem anderen Paket verfügbar macht, und

die Klassen- und Schnittstellendeklarationen , die Klassen oder Schnittstellen angibt.

- Deklarationen des Pakets `java.lang` gelten in jedem JAVA-Code.
- Importierte Deklarationen gelten in der Übersetzungseinheit, in der die *import*-Anweisung steht.

BallBeispiel

```
package ballbeispiel;                                         // 1
import java.util.*;                                           // 2
import AlgoTools.IO;                                          // 3
class Mensch {                                                // 4
    String name;                                              // 5
    String geschlecht;                                        // 6
    Hausrat hausrat;                                          // 7

    public Mensch () {                                        // 8
        name = IO.readString ("Bitte einen Vornamen eingeben:");   // 9
        geschlecht = IO.readString ("Geschlecht? (w, m) ");   // 10
        hausrat = new Hausrat ();                             // 11
    }                                                         // 12
```

```
public void tritt (Ball ball) {
    float dx, dy;
    dx = IO.readFloat ("Wie soll "+this.name+"den Ball treten? DX");
    dy = IO.readFloat ("Wie soll "+this.name+"den Ball treten? DY");
    ball.rolle (dx,dy);
}
public void empfang () {
    Besitz geschenk;
    String geschenkN;
    geschenkN = IO.readString ("Was bekommt "+this.name+"jetzt geschenkt?");
    geschenk = new Besitz (geschenkN);
    hausrat.aufnehmen (geschenk,this);
}
```

```
    public void empfang (Besitz besitz) {
        hausrat.aufnehmen (besitz,this);
    }
    public void drucke () {
        System.out.println (name + "hat "+ hausrat.toString ());
    }
}
```

```
class Hausrat extends Vector {                                    // 33

    public void aufnehmen (Besitz geschenk, Mensch
mensch) {                                                         // 34
        geschenk.gehoere (mensch);                                // 35
        this .addElement(geschenk);                               // 36
    }                                                             // 37
}                                                                 // 38

class Besitz  {                                                   // 39
    Mensch besitzer;                                              // 40
    String name;                                                  // 41

    public Besitz (String _name) {                                // 42
        name = _name;                                             // 43
    }                                                             // 44
    public void gehoere (Mensch _besitzer)  {                     // 45
        besitzer = _besitzer;                                     // 46
    }                                                             // 47
    public String toString () {                                   // 48
        return (name);                                            // 49
    }                                                             // 50
}                                                                 // 51
```

```
class Ball extends Besitz {
    float x,y;
    String farbe;

    public Ball (String _name) {
        super (_name);
        farbe= IO.readString ("Welche Farbe soll der Ball haben? ");
        x = IO.readFloat ("Wo ist er auf der X-Achse? ");
        y = IO.readFloat ("Wo ist er auf der Y-Achse? ");
    }
```

```
public void rolle (float dx,float dy) {                                        // 61
    x += dx; y += dy;                                                          // 62
}                                                                              // 63

public void drucke ()  {                                                       // 64
    System.out.println (name+“, “+farbe+“ist jetzt in Position:“+ x + ““+y);  // 65
}                                                                              // 66
}                                                                              // 67
```

```
class BallBeispiel {                                                          // 68
    public static void main (String argv[]) {                                 // 69
        Mensch mensch;                                                        // 70
        Ball ball;                                                            // 71
        mensch = new Mensch ();                                               // 72
        System.out.println (“Und jetzt bekommt “+mensch.name+“einen Ball!“);  // 73
        ball = new Ball (“ball“);                                             // 74
        mensch.empfang (ball);                                                // 75
        while (IO.readString (“Soll “+mensch.name+“den Ball treten?“).equals(“ja“)) { // 76
            mensch.tritt (ball);                                              // 77
            ball.drucke ();                                                   // 78
        }                                                                     // 79
```

```
while (IO.readString (“Soll “+mensch.name+
                    “ ein Geschenk bekommen? (ja, nein) “).equals(“ja“)) {
            mensch.empfang ();
            mensch.drucke ();
        }
    }
}
```

Wir rufen unser Programm auf mit

`java ballbeispiel.Ballbeispiel`

und sehen auf dem Bildschirm

`Bitte einen Vornamen eingeben:`

Sagen wir ruhig: *Uta*

`Geschlecht: (w,m)`

w

`Und jetzt bekommt Uta einen Ball!`

`Welche Farbe soll der Ball haben?`

blau

`Wo ist er auf der X-Achse?`

1.0

`Wo ist er auf der Y-Achse?`

1.2

`Soll Uta den Ball treten?`

ja

```
Wie soll Uta den Ball treten? DX
```

3.0

```
Wie soll Uta den Ball treten? DY
```

0.8

```
ball, blau ist jetzt in Position:4.0 2.0
Soll Uta den Ball treten?
```

nein

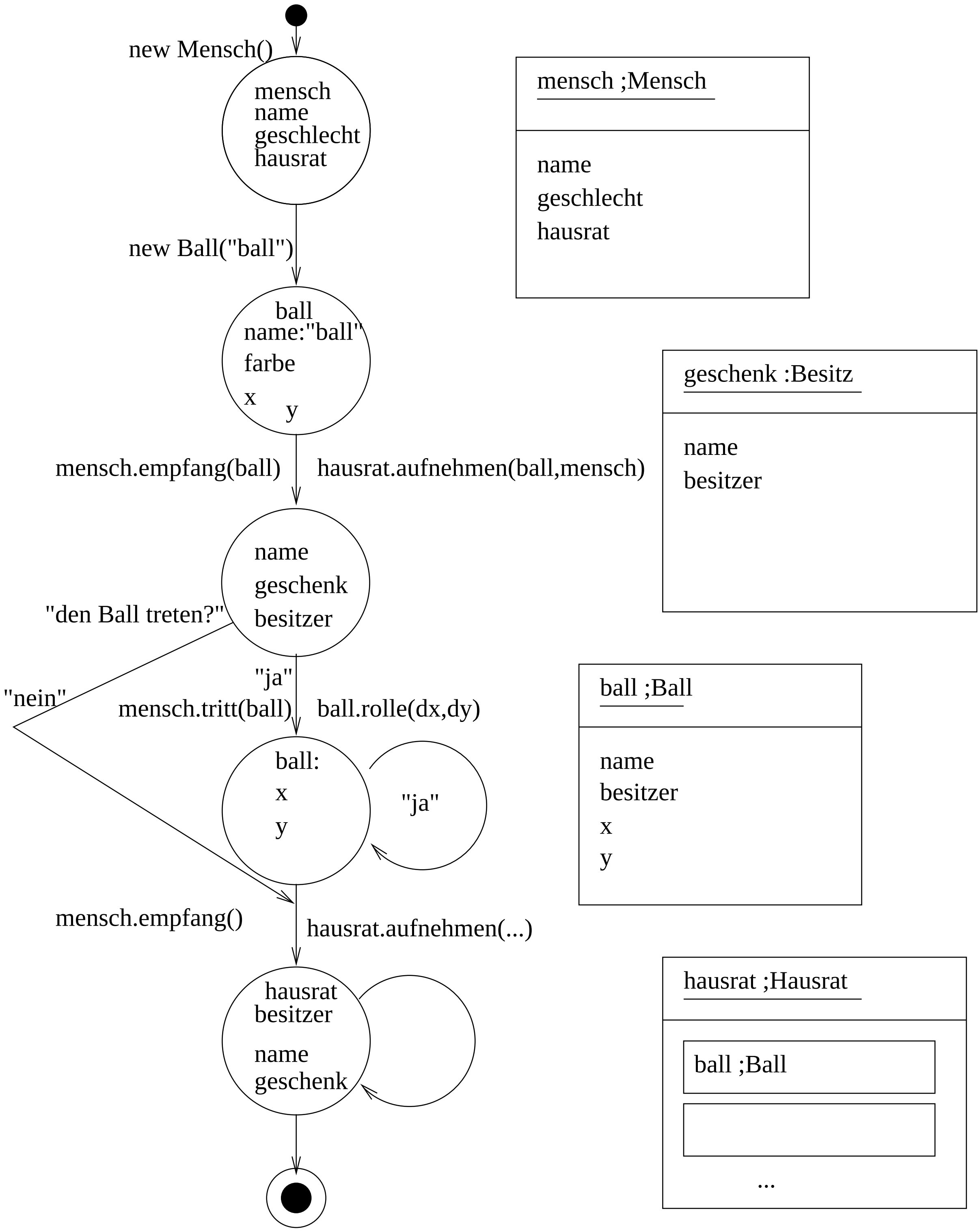
```
Soll Uta ein Geschenk bekommen? (ja, nein)
```

nein

Programmzustand

Ein Programmzustand besteht aus der Belegung aller Variablen mit einem Wert.

Wertverändernde Operationen (Methoden, Zuweisungen) überführen einen Programmzustand in den nächsten.


new Mensch()
mensch
name
geschlecht
hausrat
mensch ;Mensch
name
geschlecht
hausrat
new Ball("ball")
ball
name:"ball"
farbe
x
y
geschenk :Besitz
name
besitzer
mensch.empfang(ball)
hausrat.aufnehmen(ball,mensch)
name
geschenk
besitzer
"den Ball treten?"
"ja"
"nein"
mensch.tritt(ball)
ball.rolle(dx,dy)
ball ;Ball
name
besitzer
x
y
ball:
x
y
"ja"
mensch.empfang()
hausrat.aufnehmen(...)
hausrat
besitzer
name
geschenk
hausrat ;Hausrat
ball ;Ball
...

Zusicherungen

Aussagen über Programmzustände heißen Zusicherungen. Sie werden als logische Formeln mit dem Zustand (den Variablen) als Argument geschrieben:

$P(z)$.

Verschiedene Zusicherungen können in logischen Beziehungen stehen.

$P(j) = j > 5$ impliziert $Q(j) = j > 4$

$P \rightarrow Q$.

Wir können für wertverändernde Operationen die Zusicherungen vor und nach Ausführung der Operation angeben.

Beispiel

$k = 7;$

Vorbedingung $P(k)$: $k\ beliebig$.

Nachbedingung $Q(k)$: $k = 7$.

Warum interessieren uns Zusicherungen?

Zustandsverfolgung: Welche Zusicherungen $Q(z_n)$ gelten über den Zustand z_n, wenn wir wissen, daß $P(z_0)$ gilt?

Wie verändert also unser Programm den Ausgangszustand z_0 in n Schritten?

Bei welchem Anfangszustand $P(z_0)$ ist garantiert, daß nach n Schritten $Q(z_n)$ gilt?

Verifikation: Haben wir $P(z_0)$ als Charakterisierung des Anfangszustands und $Q(z_n)$ als Charakterisierung des Zielzustands, dann ist P, Q eine *Spezifikation*.

Hat das Programm, um dessen Zustände es geht, eine Folge von n Schritten, so daß $P(z_0)$ und $Q(z_n)$ gelten, und das Programm terminiert im Zustand z_n, dann ist das Programm spezifikationstreu oder *korrekt*.

Die Nachprüfung der Korrektheit heißt *Verifikation*.