Methodendeklaration

$MethodDeclaration$:
  $MethodHeader\ MethodBody$
$MethodHeader$:
  $Modifiers_{opt}\ Type\ MethodDeclarator\ Throws_{opt}$
  $Modifiers_{opt}$ `void` $MethodDeclarator\ Throws_{opt}$
$MethodDeclarator$:
  $Identifier$ ($FormalParameterList_{opt}$)
  $MethodDeclarator$ `[ ]`
$FormalParameterList$:
  $FormalParameter$
  $FormalParameterList$, $FormalParameter$
$FormalParameter$:
  $Type\ VariableDeclaratorId$
$Throws$:
  `throws` $ClassTypeList$
$ClassTypeList$:
  $ClassType$
  $ClassTypeList$, $ClassType$
$MethodBody$:
  $Block$
  ;

Polymorphie

Der Aufrufer einer Methode sagt, **was** er von dem Objekt will, das die Methode beherrscht. Das Objekt bestimmt, **wie** es den gewünschten Dienst erbringt.

Statische Polymorphie:

Eine Methode wird stets mit ihrem Namen und ihren Parametern bezeichnet. Folglich sind gleichnamige Methoden mit unterschiedlich vielen Parametern oder mit Parametern unterschiedlichen Typs verschiedene Methoden.

Beispiel

```
class Mensch {
    ...
    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);
    }
}
```

Dynamische Polymorphie

Eine Klasse erbt alle Eigenschaften und Methoden ihrer Oberklasse. Sie kann aber eine geerbte Methode auch re-definieren. Zur Laufzeit des Programms wird festgestellt, welche Methode tatsächlich ausgeführt wird. Es wird stets die Methode der speziellsten Klasse genommen.

Beispiel:

Hausrat ist eine Unterklasse von **Vector**, die eine Methode **toString** hat. Alle Objekte im Behälter müssen auch eine Methode **toString** haben, aber jedes Objekt kann eine andere solche Methode haben.

Besitz ist eine Klasse, deren Objekte in den Behälter **Hausrat** kommen. Sie hat eine Methode **toString**, die den Namen eines Besitzobjektes liefert. **Ball** ist eine Unterklasse von **Besitz** und kann eine Methode **toString** haben, die farbe + "Ball" liefert.

Wenn der **Hausrat** für das Drucken als String dargestellt werden soll, wird von jedem Objekt im Behälter seine Methode **toString** aufgerufen.

Block

Ein *Block* ist eine Folge von Anweisungen, die in geschweifte Klammern eingefaßt ist. Die Anweisungen verwenden die Parameter der Methode oder Variablen, die Eigenschaften von Objekten derjenigen Klasse bezeichnen, für die die Methode definiert wurde. Mit Methoden können wir Eigenschaften von Objekten verändern.

Methoden des Balls

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


public Ball (String _name,String _farbe, float _x, float _y) {
        super (_name);
        farbe = _farbe;
        x = _x;
        y = _y; }

    public void rolle (float dx,float dy) {
        x += dx; y += dy;
    }

    public void drucke () {
        System.out.println ("Ball,"+ farbe +"ist jetzt in
Position:"+ x + ""+y);
    }
}
```

Aufruf der Methode rolle

```
class Mensch {
    String name;
    String geschlecht;
    Hausrat hausrat;

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

Ein/Ausgabe Klassen

Die Bibliothek IO von Vornberger bietet Methoden zum Lesen vom Bildschirm und zum Anzeigen auf dem Bildschirm.

Beim Lesen wird zunächst eine Zeichenkette angezeigt und dann ein Wert eingelesen.

Jeder Typ des Wertes erfordert eine Methode: **readFloat**, **readString**...

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

Realisierung von 1 zu 1 Assoziationen

```
class Besitz {
    Mensch besitzer;
    String name;

    public Besitz (String _name) {
        name = _name;
    }
    public void gehoere (Mensch _besitzer) {
        besitzer = _besitzer;
    }
}
```

Realisierung von 1 zu m Assoziationen

```
class Mensch {
    String name;
    boolean geschlecht;
    Hausrat hausrat;

    public Mensch (String _name,boolean _geschlecht) {
        name = _name;
        geschlecht = _geschlecht;
        hausrat = new Hausrat ();
    }
    public void empfang (Besitz geschenk) {
        hausrat.aufnehmen (geschenk,this);
    }
}
```

Eine Behälterklasse

```
class Hausrat extends Vector {

    public void aufnehmen (Besitz geschenk, Mensch
mensch) {
        geschenk.gehoere (mensch);
        this.addElement(geschenk);
    }
}
```

Es unterstreicht den objektorientierten Charakter, daß das Objekt, dessen Methode ausgeführt werden soll, dem Methodennamen vorangestellt wird.

Parameterübergabe

kein Parameter: Die Eigenschaft (die Variable) ist bei jedem Objekt der betreffenden Klasse vorhanden und bekannt.

Parameter: Bei der Methodendeklaration wird für jeden Parameter der Typ und der Name der Variablen angegeben. Diese Variablen sind nur innerhalb der Methode bekannt. Ist die Methode abgearbeitet, sind die Variablen “vergessen”.

Beim Aufruf der Methode wird als Parameter ein bestimmter Wert angegeben, oder eine Variable, mit deren Wert die Methode arbeiten soll.

Referenzübergabe

Referenzübergabe Sei in der Methodendeklaration ein Parameter v angegeben, dessen Typ kein einfacher Datentyp ist, sei im Methodenaufruf an entsprechender Stelle der Parameterliste eine Variable w angegeben, so wird die Adresse, die als Wert von w bekannt ist, kopiert und als Wert von v innerhalb der Methode eingetragen.

Parameterdeklaration in **Besitz**:

public void **gehoere** (Mensch _besitzer) {
    besitzer = _besitzer;
}

Aufruf mit Referenzübergabe in **Hausrat**:

public void **aufnehmen** (Besitz geschenk, Mensch mensch) {

    geschenk.gehoere (mensch);...

}

Irgendwo hat $mensch$ einen Wert bekommen.

Beispiel Referenzübergabe

Adresse	32	Adresse	512	Adresse 640
name:	"ball1"	name:	"Uta"	[]
farbe:	"blau"	geschlecht:	true	
besitzer:		hausrat:	Adr. 640	

1. Uta ruft **aufnehmen(ball,this)** auf, d.h. $geschenk$ erhält als Wert Adresse 32, $mensch$ Adresse 512.

2. **gehoere(mensch)** wird aufgerufen als Methode von dem Objekt ab Adresse 32. $mensch$ (Adresse 512) wird der Wert von $\_besitzer$.

3. Die Zuweisung in der Methode **gehoere** übergibt die Kopie von Adresse 512 als Wert an die Eigenschaft $besitzer$.

Adresse	32	Adresse	512	Adresse 640
name:	"ball1"	name:	"Uta"	[Adr. 32]
farbe:	"blau"	geschlecht:	true	
besitzer:	Adr. 512	hausrat:	Adr. 640	

Wertübergabe

Wertübergabe Sei in der Methodendeklaration ein Parameter v angegeben, dessen Typ ein einfacher Datentyp ist, sei im Methodenaufruf an entsprechender Stelle der Parameterliste eine Variable w angegeben, so wird der Wert von w kopiert und als Wert von v innerhalb der Methode eingetragen.

Beispiel Wertübergabe

```
class Zaehler {
    public void erhoehe1 (int _x) {
        _x += 1;
        System.out.println ("waehrend "+ _x);
    }
}

class WertBeispiel {
    public static void main (String argv[]) {
        int y =3;
        Zaehler z;
        z = new Zaehler ();
        System.out.println ("vorher "+ y);
        z.erhoehe1 (y);
        System.out.println ("nachher "+ y);
    }
}
```

Ausgabe:

```
vorher 3

waehrend 4

nachher 3
```

```
class ZaehlerO {
    public void erhoehe1 (Geld _x) {
        _x.betrag += 1;
        _x.drucke("waehrend ");
    }
}

class Geld {
    int betrag; String waehrung;
    public Geld (int _betrag, String _waehrung) {
        betrag = _betrag;
        waehrung = _waehrung;
    }
    public void drucke (String txt) {
        System.out.println (txt + betrag + waehrung);
    }
}

class WertBeispielK {
    public static void main (String argv[]) {
        ZaehlerO z; Geld y;
        y = new Geld (3,"Euro");
        z = new ZaehlerO ();
        y.drucke ("vorher ");
        z.erhoehe1 (y);
        y.drucke ("nachher ");
    }
```

```
}
```

Ausgabe:

```
vorher 3Euro
waehrend 4Euro
nachher 4Euro
```