

*Methode, griech. der Weg zu etwas,  
planmäßiges Verfahren  
- Brockhaus*

# Parameterübergabemechanismen für den Methodenaufruf

---

Martin Wirsing

in Zusammenarbeit mit  
Michael Barth, Philipp Meier und Gefei Zhang

11/04

## Ziele

- Wiederholung der wichtigsten Begriffe bei Klassen
- Verstehen des Parameterübergabebegriffs von Java
- Verstehen der Unterschiede zwischen  
Call-by-Value und Call-by-Reference

## Klasse Point

```

public class Point
{
    private int x,y;

    public Point(int x0, int y0)
    {
        this.x = x0;
        this.y = y0;
    }

    public void move(int dx, int dy)
    {
        this.x = this.x + dx;
        this.y = this.y + dy;
    }

    public int getX()
    {
        return this.x;
    }

    public int getY()
    {
        return this.y;
    }
}

```

Attribute  
(engl. field)

Konstruktor  
(dient zur  
Erzeugung von  
Objekten der  
Klasse Point)

Methode

---

M. Wirsing: Parameterübergabe

## Klasse Point

```

public class Point
{
    private int x,y;

    public Point(int x0, int y0)
    {
        this.x = x0;
        this.y = y0;
    }

    public void move(int dx, int dy)
    {
        this.x = this.x + dx;
        this.y = this.y + dy;
    }

    public int getX()
    {
        return this.x;
    }

    public int getY()
    {
        return this.y;
    }
}

```

vordefinierte lokale  
Variable `this`  
bezeichnet das gerade  
betrachtete Objekt

Instanz-  
variable

---

M. Wirsing: Parameterübergabe

## Modellierung in UML

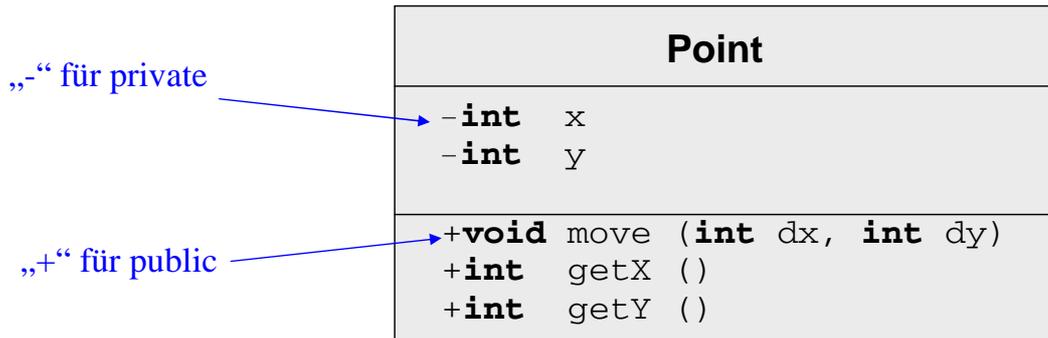
- UML ist eine graphische Darstellung zur Modellierung objekt-orientierter Systeme.

Für Klassen werden angegeben

der Klassenname, die Attribute und Methoden.

Da Konstruktoren Standardnamen besitzen, werden sie meist NICHT explizit angegeben.

- Beispiel: **Point**



M. Wirsing: Parameterübergabe

## Verwendung von Konstruktoren

- Ein neues Objekt der Klasse `Point` mit den Anfangswerten `a`, `b` wird erzeugt durch den Ausdruck

```
new Point(a, b);
```

- Beispiel:

```
Point p = new Point(5, 7);
```

legt im Keller die lokale Variable `p` an und

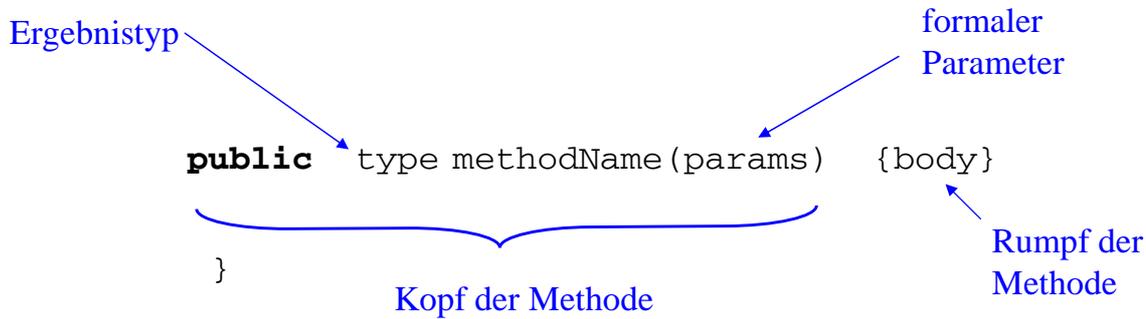
erzeugt auf der Halde ein neues `Point`-Objekt mit den Koordinaten `5`, `7`.



M. Wirsing: Parameterübergabe

## Methodendeklaration

Eine Methodendeklaration hat die Form:



**Beispiel:**

```
public void move(int dx, int dy)
{
    this.x = this.x + dx;
    this.y = this.y + dy;
}
```

## Methodenaufruf: Verwendung von Methoden

- Ein Methodenaufruf hat die Form

`o.m(a1, a2);`

**aktuelle Parameter**

- Beispiel:**

```
p.move(10, 10);
```

- Es gibt verschiedene Techniken zur Ausführung des Methodenaufruf:
  - Wertübergabe (Call by Value)
  - Adressübergabe (Call by Reference)
- Java verwendet Call by Value.

## Call-by-Value-Parameterübergabe

**Call-by-Value** Sei gegeben

Methodendeklaration: `void m (D x1, int x2) {Rumpf}`

Aufruf: `o.m(a1, a2);`

formale Parameter

aktuelle Parameter

1. **Schritt:** Berechne die Werte  $o_1, v_1, v_2$  der aktuellen Parameter  $o, a_1, a_2$  und weise diese Werte dem **impliziten** Parameter `this` und den **formalen** Parametern  $x_1, x_2$  zu, die als lokale Variablen des Rumpfs verwendet werden.
2. **Schritt:** Werte den Rumpf von `m` aus.
3. **Schritt:** Bei Beendigung der Auswertung des Rumpfs werden die lokalen Variablen **this**,  $x_1, x_2$  gelöscht (durch Zurücksetzung des „Top-Zeigers“ des Laufzeitkellerspeichers).

M. Wirsing: Parameterübergabe

## Call-by-Value-Parameterübergabe

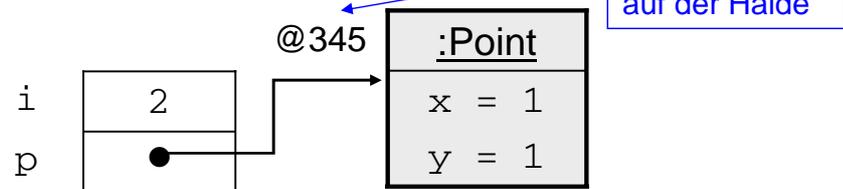
**Beispiel:**

```
int i = 2;
Point p = new Point(1,1); // (1)
p.move(i, 2+2); // (2)
```

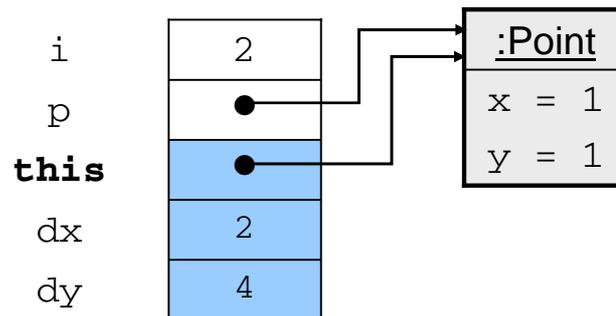
M. Wirsing: Parameterübergabe

## Call-by-Value-Parameterübergabe: Beispiel

Zum Zeitpunkt (1) habe der Speicher die Form



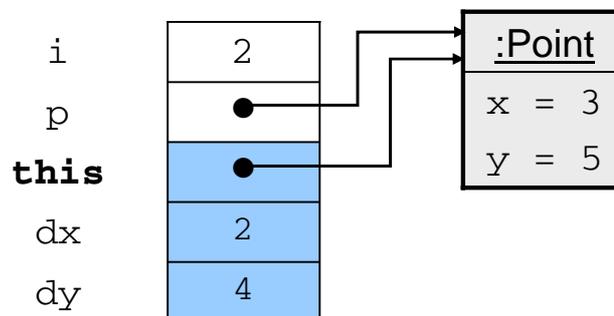
Beim Aufruf werden 3 lokale Variablen angelegt, die Werte von `p` (im Bsp. @345), `dx` (im Bsp. 2) und `dy` (im Bsp. 4) berechnet und die Initialisierungen **this** = @345; `dx` = 2; `dy` = 4; ausgeführt:



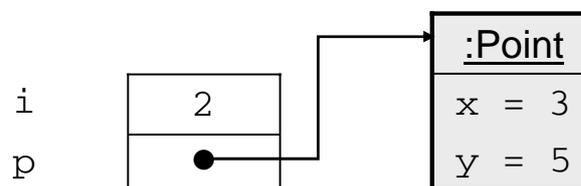
M. Wirsing: Parameterübergabe

## Call-by-Value-Parameterübergabe: Beispiel

Dann wird der Rumpf **this.x** = **this.x** + `dx`; **this.y** = **this.y** + `dy`; ausgeführt:



Zum Zeitpunkt (2) sind die lokalen Variablen **this**, `dx` und `dy` des Blocks wieder gelöscht:



M. Wirsing: Parameterübergabe

## Call-by-Value mit Objektparameter

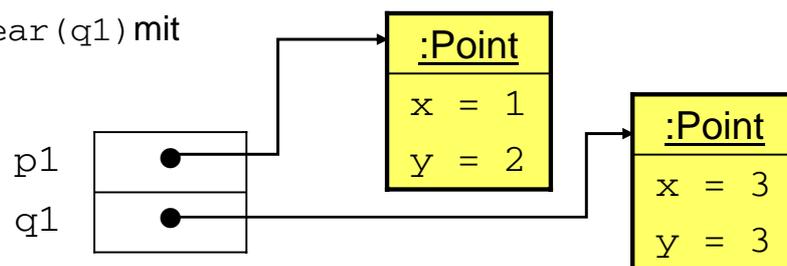
### Beispiel:

Erweitere die Klasse Point (zu einer Klasse `PointClr` mit Test `PointClrMain`) um die Methode `moveNClear`, die das aktuelle Objekt um die Koordinaten von `q` verschiebt und dann `q` auf den Ursprung setzt.

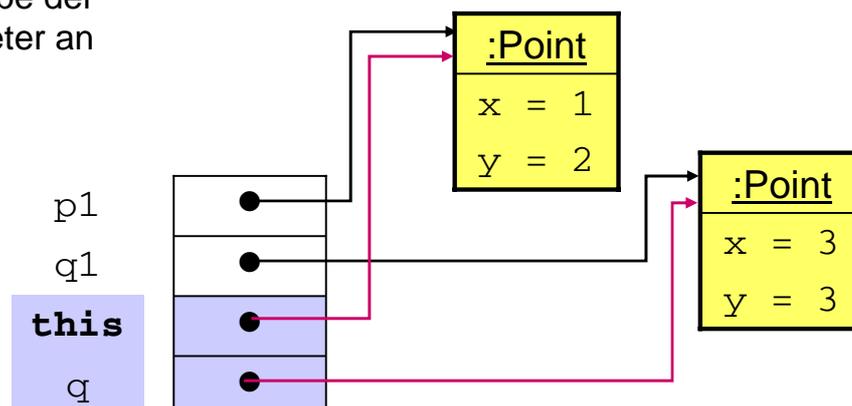
```
public void moveNClear (Point q)
{
    int dx = q.getX();
    int dy = q.getY();
    this.move(dx, dy);
    q.move(-dx, -dy);
}
```

## Call-by-Value mit Objektparameter

Aufruf `p1.moveNClear(q1)` mit

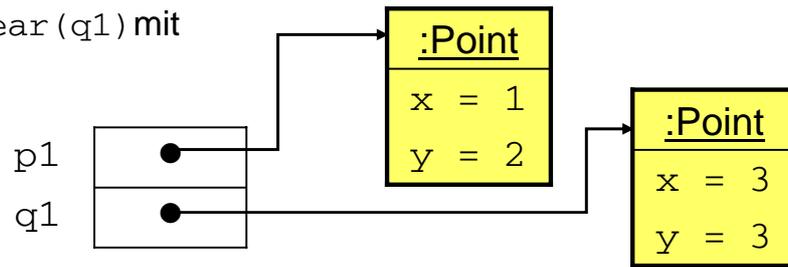


Schritt 1: Übergabe der aktuellen Parameter an `this`, `q`

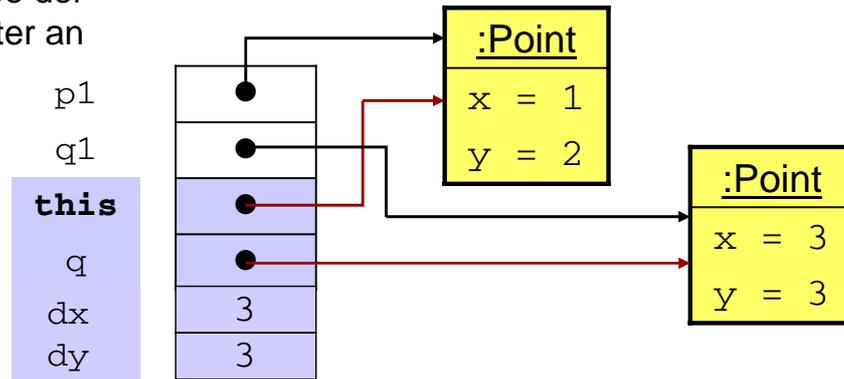


## Call-by-Value mit Objektparameter

Aufruf `p1.moveNClear(q1)` mit

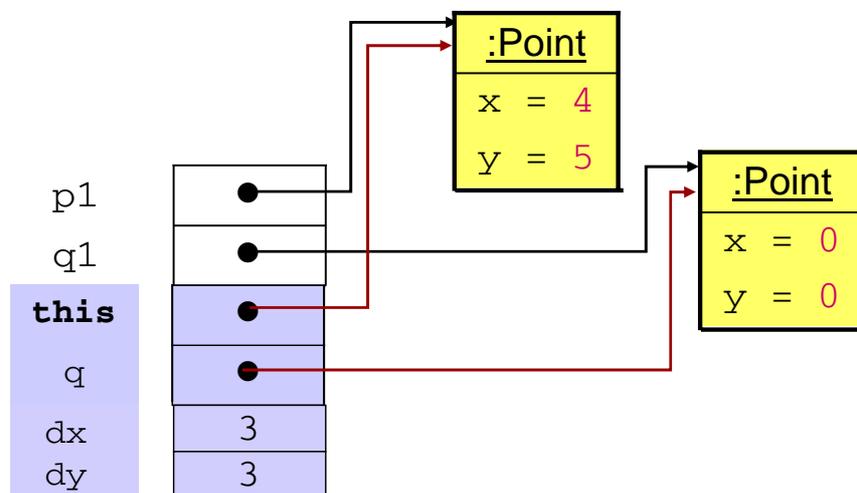


Schritt 1: Übergabe der aktuellen Parameter an `this`, `q`



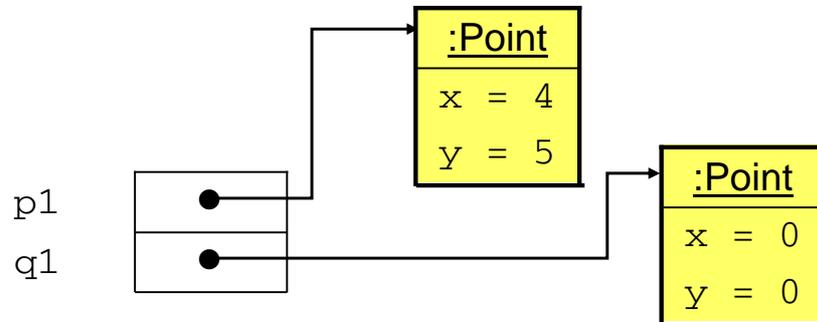
## Call-by-Value mit Objektparameter

Schritt 2: Ausführung des Rumpfs ergibt



## Call-by-Value mit Objektparameter

Schritt 3: Löschen der lokalen Variablen `this`, `q`



## Call-by-Value-Parameterübergabe

### Folgerung

- Da bei Call-by-Value die Werte der aktuellen Parameter an lokale Variablen (des Rumpfs) übergeben werden, die aktuellen Parameter aber selbst unangetastet bleiben, ändern sich die Werte der aktuellen Parameter **nicht**.
- Es können aber die Werte der Instanzvariablen eines aktuellen Parameters (vom Objekttyp) verändert werden.

## Call-by-Value und Call-by-Reference

In anderen Programmiersprachen (wie C++, Ada, C,...) gibt es neben Call-by-Value auch den Parameterübergabemechanismus Call-by-Reference (Adressübergabe).

Gegeben sei eine Methodendeklaration

```
type m(T &x) {body} // nicht in Java!
```

und ein Aufruf

```
o.m(p);
```

Referenz auf einen Wert von T

nicht in Java

## Call-by-Reference

**Schritt 1:** Übergabe des Werts von `o` an `this` und der **Adresse von `p` an `x`**

**Schritt 2:** Ausführung von `body`. Änderungen von `x` werden unter der Adresse von `p` gespeichert, d.h. direkt am aktuellen Parameter ausgeführt.

**Schritt 3:** Am Ende werden `this` und `x` gelöscht.

⇒ Änderung des Werts von `p` möglich

nicht in Java

## Call-by-Reference

### Beispiel: choose

„Wähle Punkt mit der größeren y-Koordinate“

Call-by-Value für  
this und q.

Bei der  
Ausführung  
werden  
Änderungen  
von  
higherPoint  
unter der  
Adresse des  
aktuellen  
Parameters  
gespeichert  
.

```
public void choose(Point q, Point &higherPoint)
{
    if (this.getY() >= q.getY())
        higherPoint = this;
    else
        higherPoint = q; //(2)
}

public static void main(String[] args)
{
    Point p1 = new Point(1,1);
    Point q1 = new Point(2,2);
    Point r; // (1)
    p1.choose(q1, r); // (3)
    ...
}
```

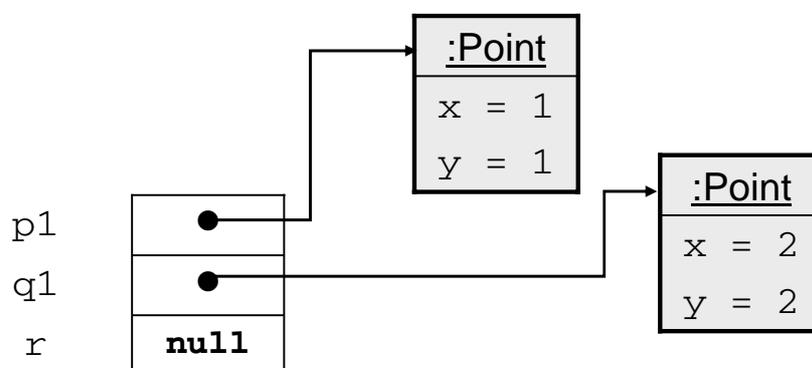
Call-by-Reference:  
higherPoint wird  
als Referenz (auf  
einen Zeiger auf  
Point) übergeben.

*nicht in Java*

M. Wirsing: Parameterübergabe

## Call-by-Reference

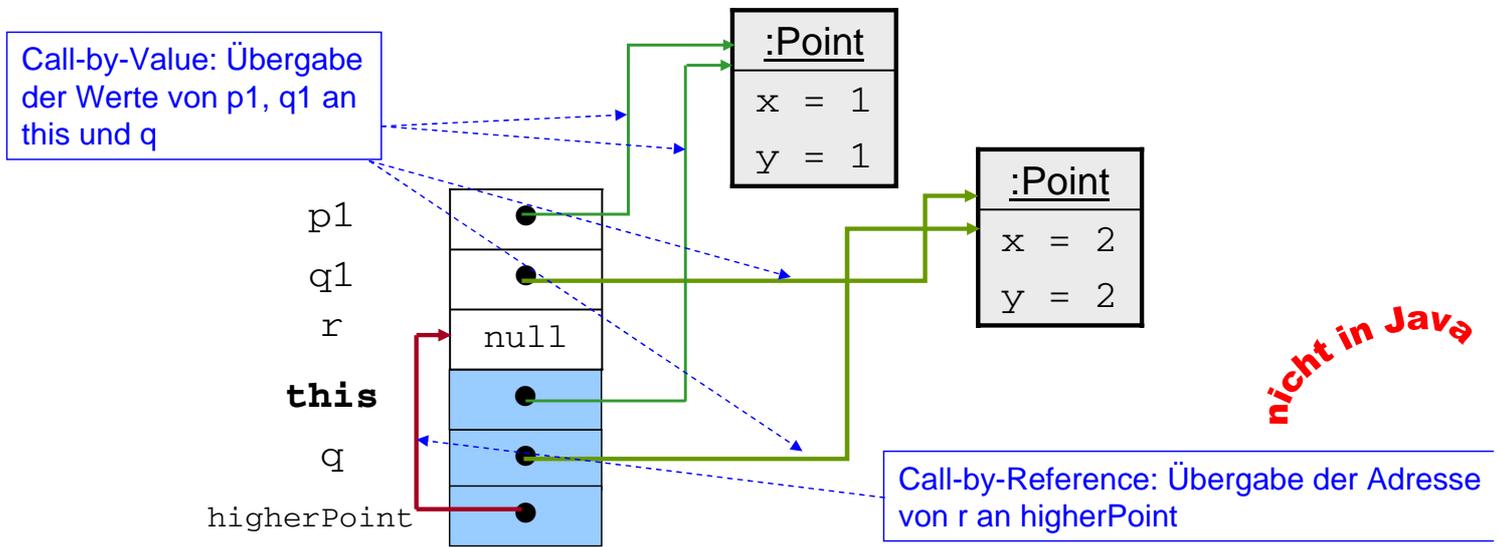
Im Zeitpunkt (1) gilt folgender Speicherzustand:



M. Wirsing: Parameterübergabe

## Call-by-Reference

Bei der Parameterübergabe wird für den Call-by-Reference-Parameter higherPoint die Adresse des aktuellen Parameters r übergeben:

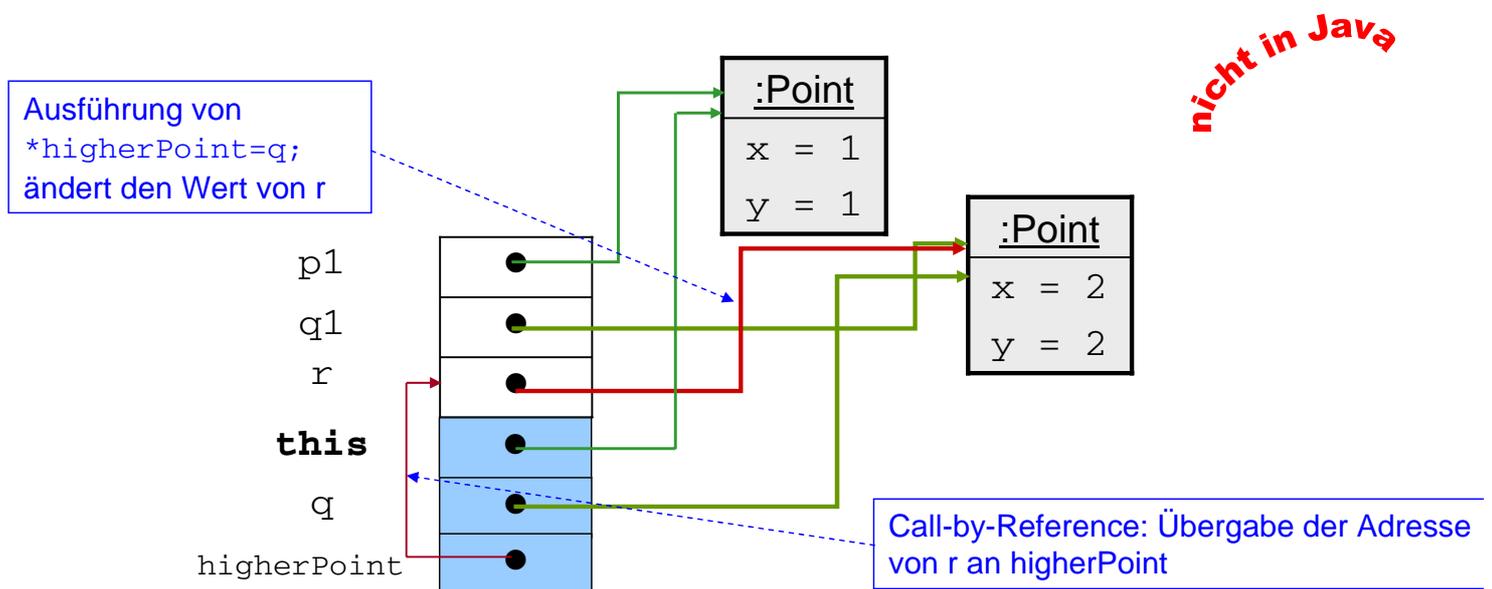


M. Wirsing: Parameterübergabe

## Call-by-Reference

Während der Ausführung des Rumpfs (Zeitpunkt 2) wird bei der Zuweisung an higherPoint die Änderung an der Adresse des aktuellen Parameters r durchgeführt:

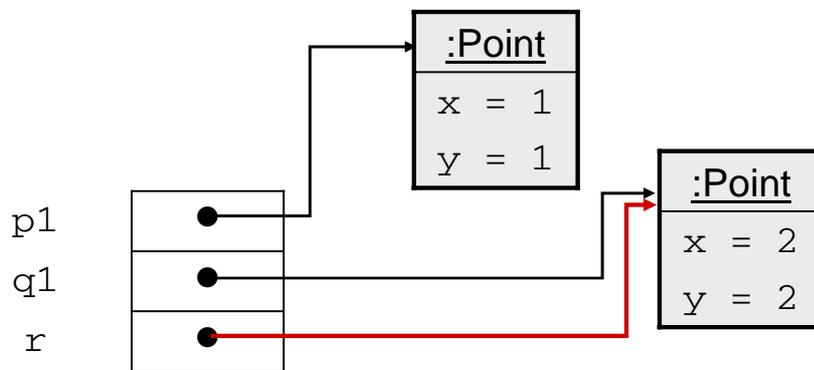
:



M. Wirsing: Parameterübergabe

## Call-by-Reference

Im Zeitpunkt (3) werden die lokalen Variablen gelöscht und man erhält man wegen Call-by-Reference eine Änderung des aktuellen Parameters: `r` ist nicht mehr `null`, sondern zeigt auf ein anderes Objekt.



*nicht in Java*

- `r` zeigt auf den Punkt mit der größeren `y`-Koordinate!

## Call-by-Value (Java)

**Beispiel:** Das (fast) gleiche [Programm](#) in Java mit Call-by-Value für `higherPoint`

```

public void choose(Point q, Point higherPoint)
{
    if (this.getY() >= q.getY())
        higherPoint = this;
    else
        higherPoint = q; // (2)
}

public static void main(String[] args) //siehe PointChoose0Main
{
    Point p1 = new Point(1,1);
    Point q1 = new Point(2,2);
    Point r = new Point(); // (1)
    p1.choose(q1, r); // (3)
    ...
}

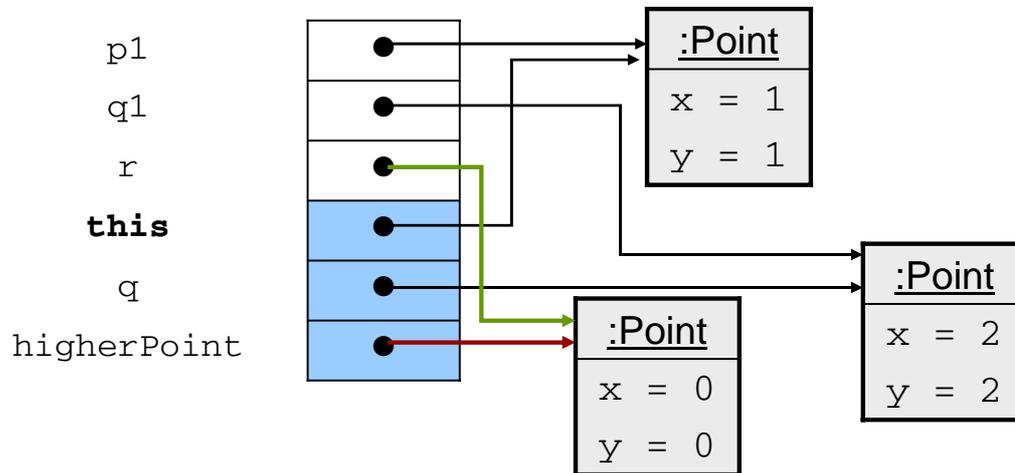
```

Call-by-Value

Initialisierung von r mit Standardkonstruktor

## Call-by-Value (Java)

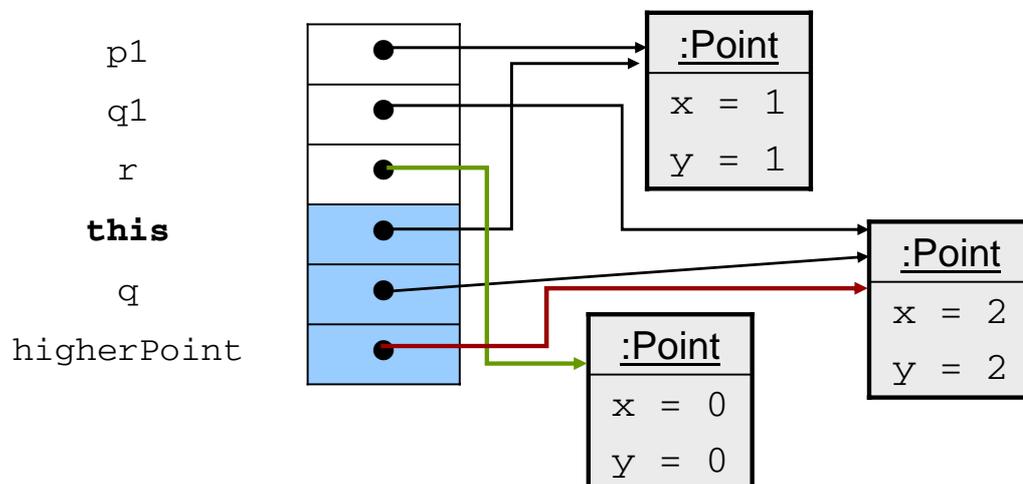
Zu Beginn der Ausführung des Rumpfs von `p1.choose(q1, r)`:



M. Wirsing: Parameterübergabe

## Call-by-Value (Java)

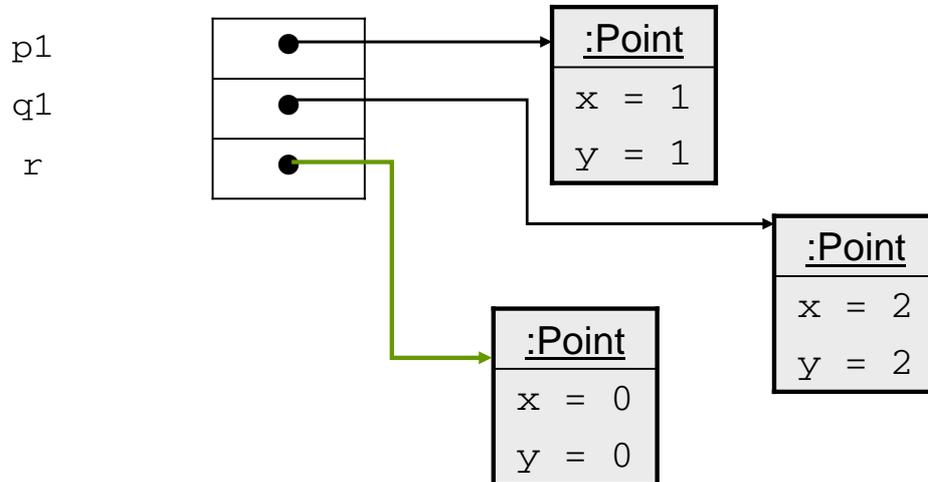
**Zeitpunkt (2):** Fallunterscheidung bewirkt eine Änderung der lokalen Variablen `higherPoint`.



M. Wirsing: Parameterübergabe

## Call-by-Value (Java)

Zum Zeitpunkt (3) haben sich also bei Call-by-Value die aktuellen Parameter **nicht** geändert:



## Call-by-Value (Java)

Um den gleichen Effekt wie bei Call-by-Reference zu erzielen, führt man in Java ein Ergebnis ein (siehe [PointChoose1](#) und [PointChoose1Main](#)):

```
public Point chooseJava(Point q)
{
    if (this.getY() >= q.getY())
        return this;
    else
        return q; // (2)
}

public static void main(String[] args)
{
    Point p1 = new Point(1,1);
    Point q1 = new Point(2,2); // (1)
    Point r = p1.chooseJava(q1); // (3)
}
```

Zum Zeitpunkt (3) ergibt sich jetzt der gewünschte Speicherzustand, bei dem r auf das gleiche Objekt zeigt, wie q1.

## Zusammenfassung

- Eine Methode berechnet ihr Resultat abhängig vom Zustand des aktuellen Objekts und der aktuellen expliziten Parameter.
- Der Parameterübergabemechanismus von Java ist Call-by-Value. Dabei werden die Werte der aktuellen Parameter an die formalen Parameter übergeben. Die Werte der aktuellen Parameter werden durch Call-by-Value nicht verändert; es können aber die Attributwerte der aktuellen Parameter verändert werden.
- Bei Call-by-Reference (wie in C, C++, Modula möglich) können die Werte der aktuellen Parameter verändert werden, da ihre Adressen (die L-Werte) übergeben werden. Java hat kein Call-by-Reference; es lassen sich aber durch Call-by-Value bei Objekten ähnliche Effekte und Speicherplatzersparnis erzielen.