Bindungen

KW 22, Zentralübung Informatik II

2006-05-30

Call by Reference

Methode swap() in Java:

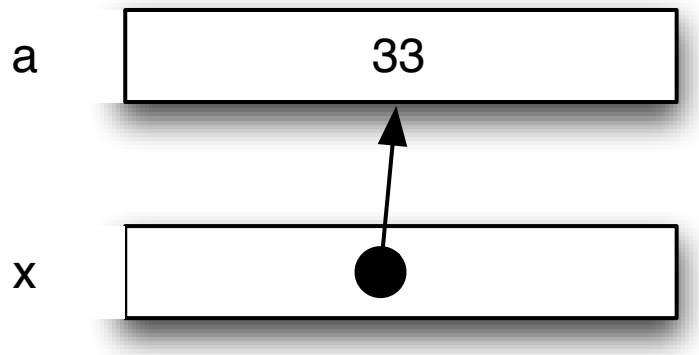
```
public static void swap(int x, int y) {
    int tmp = x;
    x = y;
    y = tmp;
}

int a = 33; int b = 2;
swap(a, b);
```

Funktioniert **nicht** wie geplant. . .
Da man keinen Zugriff mehr auf die Quelle der Daten hat

Call by Reference

Methode swap() in Pseudo-Java:

```
public static void swap(int& x, int &y) {
    int tmp = x;
    x = y;
    y = tmp;
}
```



Simuliere Referenzparameter

```
public static void swap(int[] x, int[] y) {
    assert x.length == 1 && y.length == 1;
    int tmp = x[0];
    x[0] = y[0];
    y[0] = tmp;
}
public static void main(String[] args) {
    int[] x = {23}; int[] y = {1};
    System.out.format("x=%s, y=%s\n", x[0], y[0]);
    swap(x,y);
    System.out.format("x=%s, y=%s\n", x[0], y[0]);
}
```

Java-Phänomene (1/2)

- Es gibt Teile der Java-API, die sind funktionaler Natur. Beispiel?

  String: Nicht-destruktives Konkatenieren etc.

- Java: Call-by-value mit Objekt-Referenzen

- Array-Programmierung in Java ist blöd, warum machen wir das?

  Um die Grundlagen kennenzulernen, sehr ähnlich zu Maschinensprache.

Java-Phänomene (2/2)

Warum kann ich manchmal auf die privaten Daten eines Parameters zugreifen?

```
public class Foo {
    private int counter;
    public Foo(Foo other) {
        this.counter = other.counter;
    }
}
```

Unsichtbarer Parameter „this“

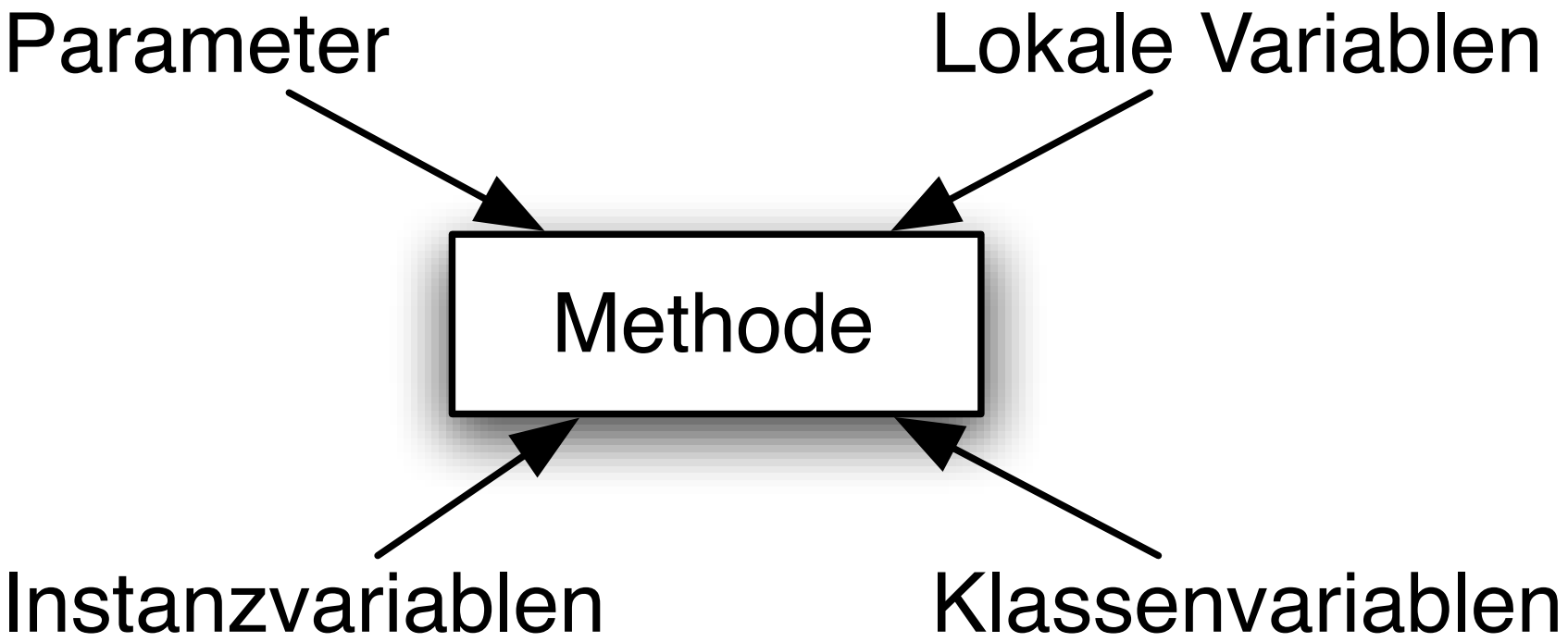
```
public class SimpleContainer1 {
    private int content;
    public SimpleContainer1(int myContent) {
        this.content = myContent;
    }
    /** Instanzmethode */
    public void printContent() {
        System.out.println("Content: "+this.content);
    }
    public static void main(String[] args) {
        SimpleContainer1 instance = new SimpleContainer1(333);
        instance.printContent(); // this ist implizit, vorangestellt
    }
}
```

Unsichtbarer Parameter „this“

Äquivalent zu SimpleContainer1, erst Vererbung bringt wesentliche Änderungen:

```
public class SimpleContainer2 {
    private int content;
    public SimpleContainer2(int myContent) {
        this.content = myContent;
    }
    /** Klassenmethode */
    public static void printContent(SimpleContainer2 myThis) {
        System.out.println("Content: "+myThis.content);
    }
    public static void main(String[] args) {
        SimpleContainer2 instance = new SimpleContainer2(333);
        printContent(instance); // this ist explizit, normales Argument
    }
}
```

Woher bekommt eine Methode ihre Daten?



Woher bekommt eine Methode ihre Daten?

- Lokale Variablen: (Indirekt) auch für Parameter und Instanzvariablen verantwortlich.
- Parameter. Die „ersten lokalen Variablen“.
- Instanzvariablen.

  Woher kommen die? Aus dem impliziten Parameter `this`.
- Klassenvariablen. Javas Version von globalen Variablen.

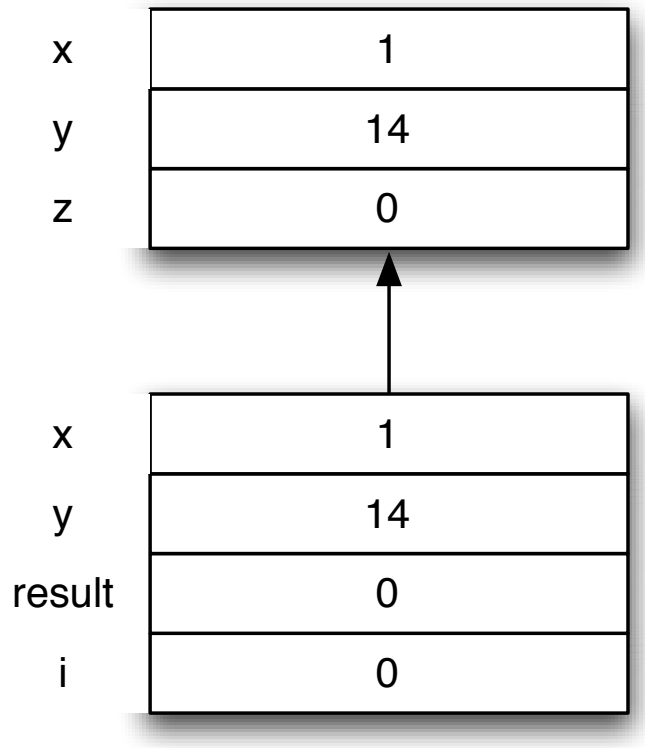
  Gedanke: Werte, die für alle Instanzen gelten.

Frage

Warum kann ich immer nur auf die lokalen Variablen der aktuellen Methode zugreifen, wenn doch alle lokalen Variablen auf dem Stack stehen?

Stack-Frames

```
public class StackFrames {
    public void doit() {
        int x=1; int y=14;
        int z = times(x,y);
    }
    private int times(int x, int y) {
        int result = 0;
        for(int i=0; i < x; i++) {
            result = result + y;
        }
        return result;
    }
}
```



Null

- Was für einen Typ hat `null`?

```
String s = null;
Integer i = null;
```

  Eigentlich gibt es für jeden Typ ein eigenes `null`.

- Was sind NullPointer-Exceptions wenn man this einfach als weiteren Parameter sieht?

  Ein Hinweis auf einen unzulässigen Parameter-Wert.