Klassenvariablen, Seiteneffekte und Gültigkeit

Martin Wirsing

in Zusammenarbeit mit
Matthias Hölzl, Piotr Kosiuczenko, Dirk Pattinson

04/03

Ziele

- Verstehen von Klassenvariablen und Klassenmethoden
- Minimieren des Gebrauchs von Seiteneffekten
- Verstehen der Unterschiede zwischen
  Instanzvariablen / Instanzmethoden und Klassenvariablen / Klassenmethoden
- Bestimmen von Gültigkeitsbereich und Lebensdauer von Variablen

Klassenvariablen und Klassenmethoden

- Eine **Klassenmethode** oder **statische Methode** ist eine Methode, die keinen impliziten (Objekt-) Parameter benötigt.

  Beispiele

  - die Methode `main`,
  - Funktionen (funktionale Programme) wie z.B.
    `+` in `String`
    `sqrt` in `Math` (Berechnung der Quadratwurzel)

- Beim Aufruf einer außerhalb der eigenen Klasse definierten Klassenmethode genügt es den Klassennamen anzugeben.

  Beispiel

```
double r = Math.sqrt(2);
```


Klassenvariablen und Klassenmethoden

Eine **Klassenvariable** oder **statische Variable** ist ein Attribut, das (global) für alle Objekte einer Klasse gilt (und typischerweise bei der Erzeugung von neuen Objekten verändert wird).

Beispiel

Wir definieren eine erweiterte Klasse `Point2`, bei der jedes Objekt eine fortlaufende Nummer erhält. Dazu definieren wir eine Klassenvariable, die die bereits vergebenen Nummern zählt.

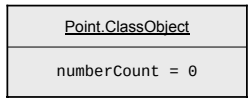
Klassenvariablen und Klassenmethoden

```
public class Point2
        {
                private static int numberCount = 0;        // Zaehler für Punkte
                /*
                        Die Klassenvariable wird sofort initialisiert, da sie in
                        den Konstruktoren fuer Instanzen nicht initialisiert
                        werden kann.
                */
                private int x,y;
                private int number;  // Nummer des aktuellen Punkts
                public Point2 (int dx, int dy)
                {
                        // erhoehe den Nummernzaehler
                        numberCount++;
                        // weise dem Punkt die neue Nummer zu
                        number = numberCount;
                        // besetze die Koordinaten
                        x = dx;
                        y = dy;
                }
                ...

                public static void main (String[] args)
                {
                        Point p1 = new Point(1,1);
                        Point p2 = new Point(2,2);
                        Point p3 = new Point(3,3);        // (1)
                }
        }
```

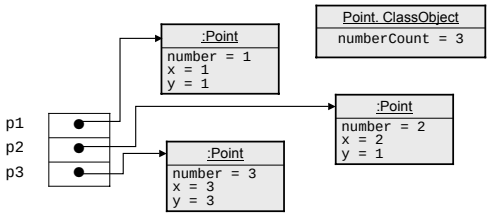

Klassenvariablen und Klassenmethoden

Zu Beginn der Ausführung von `main` werden zunächst die statischen Variablen initialisiert. Wir erhalten den Speicherzustand



Klassenvariablen und Klassenmethoden

Zum Zeitpunkt (1) wurden 3 Objekte erzeugt und die statische Variable jeweils hochgezählt. Es ergibt sich der Speicherzustand



Klassenvariablen und Klassenmethoden

Programmierstil:

Im Gegensatz zu Klassenmethoden sind Klassenvariablen **unerwünscht** und sollten vermieden werden, da Methoden, die solche Variablen lesen und modifizieren, Seiteneffekte haben.

Das Verhalten solcher Methoden hängt nicht allein von ihrer Eingabe ab und ist deswegen schwer zu verstehen und zu kontrollieren.

Zugriffs-, Mutatormethoden und Seiteneffekte

- Eine **Zugriffsmethode** (auch Anfrage, in UML query-Methode) greift auf Werte der impliziten oder expliziten Parameter zu und / oder führt weitere Berechnungen aus, ändert aber **nicht** den Speicherzustand.
- Eine **Mutatormethode** ändert (möglicherweise) den Speicherzustand des impliziten Parameters.
- Man spricht von einem **Seiteneffekt**, wenn durch den Methodenaufruf der Speicherzustand eines expliziten Parameters geändert wird.
  Allgemeiner: Ein Seiteneffekt ist **jede Art Änderung des beobachtbaren Verhaltens außerhalb des aktuellen Objekts**.

Zugriffs-, Mutatormethoden und Seiteneffekte

Beispiel: `Point`

- `getX()` und `getY()` sind Zugriffsmethoden
- `move` ist ein Mutator
- `moveNclear` ist Mutator und hat einen Seiteneffekt, da der aktuelle Parameter p geändert wird

Klassifikation von Methoden-Verhalten

sehr gut:	Funktionen, die den Zustand nicht verändern und Zugriffsmethoden
gut:	Mutatormethoden, die nur das aktuelle Objekt verändern
problematisch, aber ok:	Methoden, die explizite Parameter verändern können
schlecht:	Methoden, die Klassenvariablen oder nicht explizit genannte Objekte verändern oder Nachrichten drucken

Lebensdauer, Gültigkeit, Initialisierung von Variablen

- In Java gibt es 4 Arten von Variablen:
  - Instanzvariablen
  - Klassenvariablen
  - formale Parameter
  - lokale Variablen
- Die **Lebensdauer** einer Variablen ist die Zeit von der Erzeugung der Variablen bis zu dem Zeitpunkt, in dem sie im Speicher gelöscht wird.
- Die **Gültigkeit** ist der Teil des Programms, in dem auf die Variable zugegriffen werden kann.

Lebensdauer von Variablen

Lebensdauer

- Die **Instanzvariablen** eines Objekts werden erzeugt, wenn das Objekt konstruiert wird. Die Variable lebt, bis der Speicherbereinigungsalgorithmus das Objekt beseitigt.
- Eine **Klassenvariable** wird erzeugt, wenn die Klasse in den Speicher geladen wird (d.h. wenn das Klassenobjekt erzeugt wird). Sie lebt, bis die Klasse wieder aus dem Speicher entfernt wird (in Info2 bis zum Ende des Programms).
- Eine **lokale Variable** wird bei ihrer Deklaration erzeugt und lebt, bis der umfassende Block verlassen wird.
- Ein **formaler Parameter** ist ein lokale Variable des Methodenrumpfs. Er wird beim Methodenaufruf erzeugt und nach Ausführung des Methodenaufrufs wieder gelöscht.

Initialisierung von Variablen

Initialisierung

- Instanzvariable und Klassenvariable werden automatisch bei der Erzeugung mit einem Standardwert initialisiert (`0` für ganze Zahlen, `0.0` für Gleitzahlen, **`false`** für **`boolean`**, **`null`** für Objekte), falls der Initialwert nicht explizit spezifiziert wurde (im Konstruktor oder bei der Deklaration).
- Parametervariable werden mit lokalen Kopien der (R-) Werte der aktuellen Parameter initialisiert.
- Lokale Variablen müssen durch das Programm explizit initialisiert werden.

Gültigkeit von Variablen

Gültigkeit

- Instanzvariablen und Klassenvariablen werden üblicherweise als „private“ spezifiziert und sind deshalb nur in den Methodenrümpfen der eigenen Klasse gültig.
- Der Gültigkeitsbereich einer lokalen Variable erstreckt sich von der Deklaration bis zum Ende des einschließenden Blocks.
- Der Gültigkeitsbereich eines formalen Parameters ist der Rumpf des zugehörigen Blocks.
- Instanzvariablen können durch lokale Variablen und formale Parameter (eines Methodenrumpfs der gleichen Klasse) mit gleichem Namen verschattet werden. In diesem Bereich leben die Instanzvariablen, sind aber nicht gültig.

Call-by-Value (Java)

Beispiel:

In der `Point`-Methode `move`

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

Die Attribute `x, y` sind durch die formalen Parameter verschattet. Man kann aber mit **this**.x, **this**.y darauf zugreifen.

Zusammenfassung

- Eine **Klassenmethode** oder **statische Methode** ist eine Methode, die keinen impliziten (Objekt-) Parameter benötigt, wie etwa `main` oder die Addition. Eine **Klassenvariable** ist ein Attribut, das (global) für alle Objekte einer Klasse gilt. Klassenvariablen und Klassenmethoden sollten bei objekt-orientierten Programmen so wenig wie möglich verwendet werden.
- **Seiteneffekte** sind extern beobachtbare Veränderungen eines Methodenaufrufs, die nicht den Zustand des aktuellen Objekts betreffen. Sie sollten soweit wie möglich vermieden werden.
- Die **Lebensdauer** einer Variablen ist der Zeitraum von der Erzeugung der Variablen im Speicher bis zum Löschen der Variablen. Der **Gültigkeitsbereich** ist derjenige Teil des Programmtextes, in dem auf die Variable zugegriffen werden kann.