

Software Entwicklung 1

Annette Bieniusa / Arnd Poetzsch-Heffter

AG Softech
FB Informatik
TU Kaiserslautern

Prozeduren

Prozeduren

Prozeduren erlauben es von konkreten Anweisungen bzw. Anweisungssequenzen zu abstrahieren.

D.h. Prozeduren legen die Parameter einer zusammengesetzten Anweisung fest und geben ihr einen Namen.

Dies ermöglicht:

- *Wiederverwendung*
- *Schnittstellenbildung und Information Hiding*

Darüber hinaus ermöglichen Prozeduren die rekursive Ausführung von Anweisungen.

Begriffsklärung: Prozedur

Eine *Prozedur* ist eine Abstraktion einer Anweisung.

Sie gibt der Anweisung einen Namen und legt fest, was die Parameter der Anweisung sind.

Prozeduren können Ergebnisse liefern.

Beispiel: Prozedur als Abstraktion I

Aufgabe: Berechne den Absolutbetrag einer ganzen Zahl, gespeichert in Variable i , und schreibe ihn in die Variable x .

Algorithmus:

- 1 Wenn i größer oder gleich 0, liefere i als Ergebnis; andernfalls $-i$.
- 2 Weise das Ergebnis an x zu.

Beispiel: Prozedur als Abstraktion II

```
public class Absolutbetrag {  
    public static void main(String[] args){  
        int i;  
        int x;  
        i = StdIn.readInt();
```

```
int result;  
if (i >= 0) {  
    result = i;  
} else {  
    result = -i;  
}
```

Abstraktion bzgl. `i`,
`result` wird zum Ergebnis.

```
        x = result;  
        System.out.println(x);  
    }  
}
```

Beispiel: Prozedur als Abstraktion III

```
public class Absolutbetrag {
    public static int abs( int n ) {
        if (n >= 0) {
            return n;
        } else {
            return -n;
        }
    }

    public static void main(String[] args){
        int i;
        int x;
        i = StdIn.readInt();

        x = abs(i);
        StdOut.println(x);
    }
}
```

Deklaration von Prozeduren in Java

In Java werden Prozeduren als statische Methode implementiert.

Syntax in Java:

```
public static <Typ> <Prozedurbezeichner> ( <formaleParameter> )  
    <Anweisungsblock>
```

Semantik:

Die Semantik ist durch den Aufrufmechanismus und den Anweisungsblock, den sogenannten *Prozedurrumpf* bestimmt. (Genauerer siehe unten).

Rückgabeanweisung

Syntax in Java:

```
return ;  
return <Ausdruck> ;
```

Semantik:

- Das einfache `return` ist nur in Prozeduren **ohne** Rückgabewert erlaubt. Es beendet die Ausführung der Prozedur und setzt die Ausführung an der Aufrufstelle fort. Prozeduren ohne Rückgabe haben als Rückgabotyp `void`.
- Das `return <Ausdruck>` ist nur in Prozeduren **mit** Rückgabewert erlaubt. Zunächst wird der Ausdruck ausgewertet. Dann wird die Ausführung der Prozedur beendet. Der Wert des Ausdrucks wird als Ergebnis zurückgeliefert und die Ausführung an der Aufrufstelle fortgesetzt. Der Typ des Rückgabewertes muss dem deklarierten Rückgabotyp entsprechen.

Prozeduraufruf (engl. procedure call)

Syntax in Java:

`<Prozedurbezeichner> (<AktuelleParameterliste>)`

wobei

<code><AktuelleParameterliste></code>	\rightarrow	ϵ
		<code><Ausdrucksliste></code>
<code><Ausdrucksliste></code>	\rightarrow	<code><Ausdruck></code>
		<code><Ausdruck> , <Ausdrucksliste></code>

Semantik:

- 1 Werte die Ausdrücke der aktuellen Parameter aus.
- 2 Übergebe die Ergebnisse an die formalen Parameter.
- 3 Führe die Anweisungen der Prozedur aus.
- 4 Liefere den Rückgabewert, wenn vorhanden.

Prozeduren, die Ergebnisse liefern, heißen ***Funktionsprozeduren***.

Beispiel: Maximum von drei Zahlen I

Aufgabe

Schreiben Sie den folgenden Code so um, dass die statische Methode `int max(int a, int b)` verwendet wird!

Beispiel: Maximum von drei Zahlen II

```
public class MaxDrei {
    //Berechnet das Maximum von zwei int-Werten
    public static int max(int a, int b) {
        if (a > b) {
            return a;
        } else {
            return b;
        }
    }

    public static void main(String[] args) {
        int x = StdIn.readInt(); // Initialisieren von x
        int y = StdIn.readInt(); // Initialisieren von y
        int z = StdIn.readInt(); // Initialisieren von z

        int max;
        // Hier soll die Berechnung des Maximums von x,y,z erfolgen!

        StdOut.println(max);
    }
}
```

Begriffsklärung: Effekte

Die Ausführung von Prozeduren verändert den Speicherzustand und bewirkt Ein- und Ausgaben.

Zusammenfassend sprechen wir von den **Effekten** der Ausführung einer Prozedur.

- **Haupteffekte**: Liefern von Ergebnissen, Verändern von Variablenparametern
- **Seiteneffekte**: Ein- und Ausgabe, Verändern globaler Variablen, Erzeugen und Löschen dynamischer Variablen

Beispiel: Prozedur mit Seiteneffekt I

```
// Lese Zahlen von der Eingabe ein und addiere sie,  
// bis eine Null eingegeben wird.  
// Gebe die Summe der eingegebenen Zahlen aus  
// und liefere die Anzahl der Eingaben zurueck.
```

```
public static int summiereEingaben() {  
    int anz = 0;  
    int summe = 0;  
    int eingabe = 0;  
    do {  
        StdOut.print("Summand (0 beendet):");  
        eingabe = StdIn.readInt();  
        summe = summe + eingabe;  
        anz++;  
    } while( eingabe != 0 );  
  
    StdOut.println( "Summe: " + summe );  
    return anz - 1;  
}
```

Beispiel: Prozedur mit Seiteneffekt II

Anwendung der Prozedur `summiereEingaben`:

```
public static void main( String[] args ) {
    boolean b = true;
    while( b ) {
        int anzahl;
        anzahl = summiereEingaben();
        StdOut.println("Es wurden " + anzahl + " Zahlen summiert");

        StdOut.println("Beenden mit j/n:");
        String abbruch = StdInt.readString();
        if( abbruch.equals("j") ) {
            b = false;
        }
    }
}
```

Bemerkungen

- Die Hauptprozedur `main` wird vom Betriebssystem aufgerufen. Das Starten eines Java-Programms entspricht dem Aufruf der Hauptprozedur.
- Argumente vom Programmaufruf werden `main` als String-Array mitgegeben.

Bibliotheken

Bibliotheken

- Sammlungen von Prozeduren (bzw. statischen Methoden), die gedacht sind in verschiedenen Programmen eingesetzt zu werden, nennen wir **Bibliotheken** (engl. *libraries*).
- Dabei werden Prozeduren, die ähnliche Aufgabengebiete abdecken, in jeweils eigene Bibliotheken zusammengefasst.
- Beispiele: StdOut, StdIn, Math
- Bibliotheken definieren dabei eine Programmierschnittstelle (API, *Applications Programming Interface*).
- Für jede durch eine Bibliothek definierte Prozedur benötigen Programmierer um die Prozedur verwenden zu können
 - die Signaturen der Prozeduren
 - Beschreibung bzw. Spezifikation des Verhaltens der Prozedur

Beispiel: Datenanalyse

Wir wollen im Folgenden eine Bibliothek zur Analyse von Datensammlungen entwickeln mit folgender API:

<code>public class Statistics</code>	
<code>double max(double[] a)</code>	Berechnet das Maximum der Arraykomponenten
<code>double min(double[] a)</code>	Berechnet das Minimum der Arraykomponenten
<code>double mean(double[] a)</code>	Berechnet den Mittelwert
<code>double var(double[] a)</code>	Berechnet die Varianz
<code>double stddev(double[] a)</code>	Berechnet die Standardabweichung
<code>double median(double[] a)</code>	Berechnet den Zentralwert
<code>void sort(double[] a)</code>	Sortiert die Einträge des Arrays
<code>double[] readArray()</code>	Liest ein Array von <code>double</code> -Werten ein
<code>void printArray(double[] a)</code>	Gibt ein Array von <code>double</code> -Werten aus

Beispiel: Maximum

```
// Berechnet das Maximum der Arrayeinträge
public static double max(double[] a) {
    double max = a[0];
    for (int i = 1; i < a.length; i++) {
        if (a[i] > max) {
            max = a[i];
        }
    }
    return max;
}
```

- Arrays können auch Länge 0 haben! Diesen Fall werden wir in einer der nächsten Vorlesungen näher betrachten (→ Spezifikation).
- Für die Implementierung der Bibliotheksprozeduren hier behandeln wir nur zunächst nur Arrays, die mindestens ein Element enthalten (d.h. `a.length > 0`).

Beispiel: Mittelwert

Mathematische Definition des Mittelwertes von n Messwerten a_0, \dots, a_{n-1} :

$$\bar{a} = \frac{1}{n} \sum_{i=0}^{n-1} a_i$$

```
// Berechnet den Mittelwert der Arrayeinträge
public static double mean(double[] a) {
    double sum = 0.0;
    for (int i = 0; i < a.length; i++) {
        sum = sum + a[i];
    }
    return sum/a.length;
}
```

Beispiel: Varianz

Mathematische Definition der Varianz von n Messwerten a_0, \dots, a_{n-1} :

$$s^2 = \frac{1}{n-1} \sum_{i=0}^{n-1} (a_i - \bar{a})^2$$

```
// Berechnet die korrigierte Stichprobenvarianz der
// Arrayeinträge
public static double var(double[] a) {
    double m = mean(a);
    double sum = 0.0;
    for (int i = 0; i < a.length; i++) {
        sum = sum + (a[i] - m) * (a[i] - m);
    }
    return sum / (a.length - 1);
}
```

Beispiel: Einlesen von Werten

```
// Einlesen des Arrays: Als erster Wert wird die  
// Grosse des Arrays eingelesen, dann die einzelnen Eintraege.  
public static double[] readArray() {  
    int n = StdIn.readInt();  
    double[] a = new double[n];  
  
    for(int i = 0; i < n; i++){  
        a[i] = StdIn.readDouble();  
    }  
    return a;  
}
```

Beispiel: Median I

Der Median kann auf folgende Weise bestimmt werden:

- 1 Alle Werte werden (aufsteigend) geordnet.
- 2 Wenn die Anzahl der Werte ungerade ist, ist die mittlere Wert der Median.
- 3 Wenn die Anzahl der Werte gerade ist, definieren wir den Median als arithmetisches Mittel der beiden mittleren Werte.

$$\tilde{a} = \begin{cases} a_{n/2} & \text{falls } n \text{ ungerade ist} \\ \frac{1}{2}(a_{n/2} + a_{n/2-1}) & \text{if } n \text{ gerade ist} \end{cases}$$

Wie sieht eine Implementierung der Methode `double median(double[] a)` aus?

Hinweis: Verwenden Sie dazu die Methode `void sort(double[] a)`!

Beispiel: Median II

Beispiel: Median III

Achtung:

- Beim Aufruf von `void sort(double[] a)` wird als Argumentvariable `a` die **Referenz** auf ein Array übergeben.
- Dieses referenzierte Array wird durch den Aufruf der Methode `void sort(double[] a)` verändert.
- Die Methode liefert daher weder ein neues Array noch eine neue Referenz zurück (im Gegensatz zu `double[] readArray()`).
- Um diesen (häufig unerwünschten) Effekt zu verhindern, kann man eine Kopie des Arrays erzeugen, die dann der Methode übergeben und dort modifiziert wird, während das ursprüngliche Array unverändert bleibt.

Hinweis: Die Implementierung von `void sort(double[] a)` sowie Alternativen betrachten wir in der Vorlesungseinheit zu “Suchen und Sortieren” im Detail.

Gültigkeitsbereich I

Der Gültigkeitsbereich (engl. *scope*) eines Variablennamens sind die Anweisungen, die auf diesen Variablennamen Bezug nehmen können.

In Java gilt:

- Für Parameter von statische Methoden entspricht der Gültigkeitbereich dem Methodenrumpf.
- Für Variablen, die in einem Anweisungsblock einer statischen Methode deklariert werden, entspricht der Gültigkeitsbereich den Anweisungen von der Deklaration bis zum Ende dieses Blocks.
- In Anweisungsblöcken, die nicht ineinander verschachtelt sind, können Variablen mit identischen Namen definiert werden (vgl. die Zählvariablen in den for-Schleifen aus der letzten Vorlesung).

Gültigkeitsbereich II

```
1  public class Wettspiel {
2      public static void main (String[] args) {
3          int einsatz    = Integer.parseInt(args[0]);
4          int ziel       = Integer.parseInt(args[1]);
5          int versuche   = Integer.parseInt(args[2]);
6          int gewinne    = 0;
7
8          for (int t = 0; t < versuche; t++) {
9              int cash = einsatz;
10             while (cash > 0 && cash < ziel) {
11                 if (Math.random() < 0.5) {
12                     cash++;
13                 } else {
14                     cash--;
15                 }
16                 if (cash == ziel) {
17                     gewinne++;
18                 }
19             }
20         }
21         System.out.println(100 * gewinne/versuche + "% Erfolg");
22     }
23 }
```

Gültigkeitsbereich III

- Der Gültigkeitsbereich von `versuche` entspricht den Anweisungen in Zeile 5-21.
- Der Gültigkeitsbereich von `cash` entspricht den Anweisungen in Zeile 9-20.
- Der Gültigkeitsbereich von `t` entspricht den Anweisungen in Zeile 8-20.

Bemerkungen

- Jede .java-Datei kann eine beliebige Anzahl von statischen Methoden definieren.
- Die Reihenfolge der Methoden in der Datei ist beliebig.
- Eine Methode kann immer nur (max.) einen Wert zurückliefern, auch wenn sie mehrere `return`-Anweisungen enthält.
- Methoden können den gleichen Namen haben, wenn sie über die Anzahl oder Typen der Parameter unterschieden werden können (→ Überladen / Overloading)

```
public static int abs(int x) {  
    if (x >= 0) {  
        return x;  
    } else {  
        return -x;  
    }  
}
```

```
public static int abs(double x) {  
    if (x >= 0.0) {  
        return x;  
    } else {  
        return -x;  
    }  
}
```

Zusammenfassung

- Prozeduren als statische Methoden in Java
- Haupt- und Seiteneffekte beim Ausführen von statischen Methoden
- Beispiel für Bibliothek von Prozeduren: `Statistics.java`
- Gültigkeitsbereiche von (lokalen) Variablen