TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Software Entwicklung 1

Annette Bieniusa

AG Softech
FB Informatik
TU Kaiserslautern

Lernziele

- Prozeduren als statische Methoden Java zu deklarieren und an geeigneter Stelle aufzurufen
- Die Semantik eines Prozeduraufrufs zu beschreiben
- Overloading in Prozeduren mit Hilfe von Beispielen zu erklären
- Haupt- und Seiteneffekte von Prozeduren zu ermitteln
- Gültigkeitsbereiche von Variablen benennen zu können

Prozeduren

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

Dies ermöglicht:

- *Wiederverwendung*
- *Schnittstellenbildung und Information Hiding*

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 (**Funktionsprozeduren**).

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;
    }

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

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

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:

ProzedurDeklaration →
  `public static` TypOderVoid ≪ *Bezeichner* ≫ (FormaleParameter)
    Anweisungsblock

TypOderVoid → Typ | `void`

FormaleParameter →
    FormalParamListe
  | ϵ

FormalParamListe →
    FormalParam , FormalParamListe
  | FormalParam

FormalParam →
    Typ ≪ *Bezeichner* ≫

Rückgabeanweisung

Syntax in Java:

```
Anweisung  → ...
           | return ;
           | return Ausdruck ;
```

Semantik:

- Einfaches `return`
  - Nur in Prozeduren **ohne** Rückgabewert erlaubt
  - Rückgabetyp `void`
  - Beendet die Ausführung der Prozedur und setzt die Ausführung an der Aufrufstelle fort
- `return` mit Ausdruck
  - Nur in Prozeduren **mit** Rückgabewert (muss deklariertem Rückgabetyp entsprechen)
  - Werte zunächst den Ausdruck aus
  - Beende die Ausführung der Prozedur und setze Ausführung an der Aufrufstelle fort
  - Prozeduraufruf wertet zu dem Rückgabewert aus

Prozeduraufruf (engl. procedure call)

Syntax in Java:

$$\texttt{MethodenAufruf} \rightarrow \ll Bezeichner \gg \ (\texttt{AktuelleParameterListe})$$

wobei

$$\begin{aligned} \texttt{AktuelleParameterListe} &\rightarrow \varepsilon \\ &\mid \texttt{AusdruckListe} \end{aligned}$$

$$\begin{aligned} \texttt{AusdruckListe} &\rightarrow \texttt{Ausdruck} \\ &\mid \texttt{Ausdruck} \ , \ \texttt{AusdruckListe} \end{aligned}$$

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.

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




    StdOut.println(maximum);
}
```

Überladen / Overloading von Methoden

Methoden können den gleichen Namen haben, wenn sie über die Anzahl oder Typen der Parameter unterschieden werden können.

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

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

Frage

Warum wird die folgende Variante nicht vom Java-Compiler akzeptiert?

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

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 über den Rückgabewert (in anderen Sprachen auch über das Verändern von Variablenparametern)
- **Seiteneffekte**: Alle anderen Effekte, zum Beispiel: 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");

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

Frage

Was sind die Haupteffekte und die Seiteneffekte der Prozedur `summiereEingabe()`?

```
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;
}
```

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:

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

Beispiel: Maximum

```
// Berechnet das Maximum der Arrayeintraege
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 ($\rightarrow$ 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 $a_0, \ldots, a_{n-1}$:

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

```
// Berechnet den Mittelwert der Arrayeintraege
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 $a_0, \ldots, a_{n-1}$:

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

```
// Berechnet die korrigierte Stichprobenvarianz der
    Arrayeintraege
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.

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

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

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

Beispiel: Median II

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

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).

Was ist der Gültigkeitsbereich von einsatz, t, cash?

```
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 }
```

Globale und lokale Variablen

Globale und lokale Variablen

	Globale Variablen	*Lokale Variablen*
Deklarationsort	außerhalb von Prozeduren	innerhalb einer Prozedur
Speicherzuweisung	genau eine Speichervariable	zu jeder Prozedurinkarnation eine Speichervariable
Lebensdauer	gesamter Ablauf des Programms	zugehörige Prozedurinkarnation bzw. Anweisungsblock, in der sie deklariert wurde

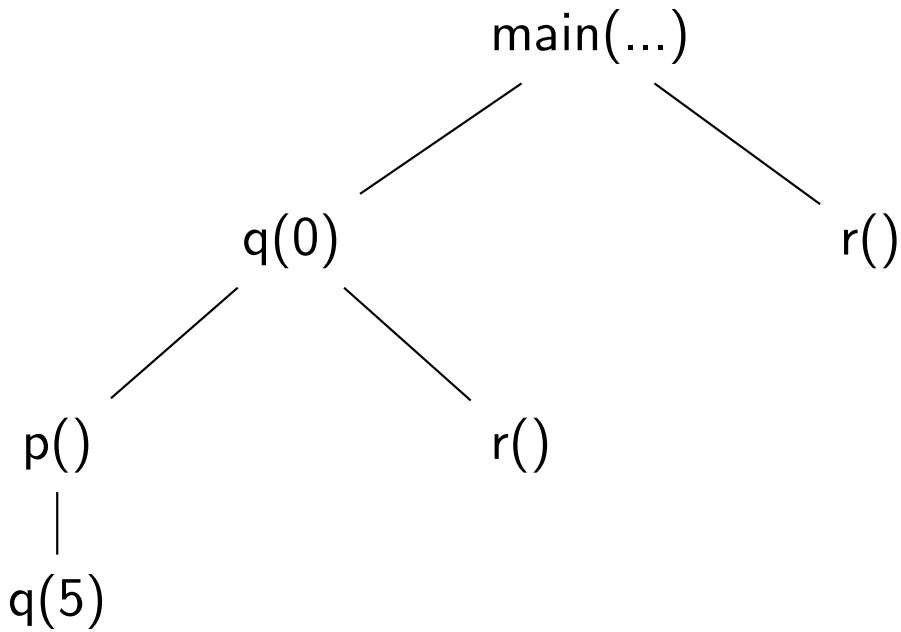
Prozedurinkarnation

Beim Aufruf einer Prozedur p wird eine neue **Inkarnation** von p erzeugt.

- Speicherplatz für die Parameter und lokalen Variablen wird angelegt
- Anweisung wird gemerkt, an der nach Ausführung der Prozedurinkarnation fortzusetzen ist

- Lebensdauer ein Prozedurinkarnation beginnt mit Erzeugung und endet mit Ausführung des Rumpfes
- Lebensdauer verschiedener Inkarnationen der gleichen Prozedur können sich überlappen
  $\Rightarrow$ Kopien der lokalen Variablen!

Prozeduraufrufbaum



Lebensdauer von Arrays

- Beginnt mit Erzeugung und endet mit Terminierung des Programms
- Dies gilt i.A. **nicht** für die referenzierenden Variablen!

```
static int[] erzeugeArray (int n) {
  if (n < 0) {
    n = 0;
  }
  return new int[size];
}

static void verwendeArray() {
  int[] a = erzeugeArray(78);
  ...
}
```

Statische und dynamische Aspekte der Programmierung

Statisch: Aspekte, die sich auf das Programm beziehen und die man aus ihm ersehen kann, *ohne es auszuführen*

- Programmvariablen
- Prozedurdeklarationen
- Anweisungen und Ausdrücke

Dynamisch: Aspekte, die sich auf die Ausführungszeit beziehen

- Speichervariablen
- Prozedurinkarnationen
- Ablauf, Ausführung
- Aufrufbäume
- Lebensdauer (von Prozedurinkarnationen, usw.)

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
- Prozedurinkarnationen und Lebensdauer von Variablen