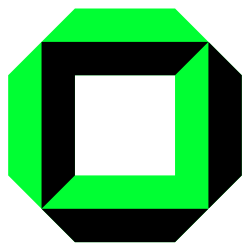
Info I – Übungsblatt 3

Joachim Breitner

mit Aufgaben von Martin Kiefel und Christian Maier

21. November 2005



Tutoriums-Homepage

`http://www.joachim-breitner.de/wiki/Infotut`

- Folien der Tutorien
- Themen-Wunschliste
- Links
- Es ist ein Wiki: Sei mutig!©

Übungsblatt-Rückblick

- Erfreuliches Ergebnis:
  - Wieder fast alle mehr als 50%
  - Schnitt 25 Punkte
  - 7× volle Punktzahl

EBNF-Tipps

- Punkt am Ende nicht vergessen!
- Keine trivialen Nichtterminale

  ```
  Punkt = ".".
  ```

- Nichtterminale wiederverwenden. Nicht:

  ```
  Ziffer = "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
  Ziffer0 = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
  ```

  Sondern:

  ```
  Ziffer = "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
  Ziffer0 = "0"|Ziffer.
  ```

- Nicht alles in eine große lange Regel packen!

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden
2. Ziffern 1,0,0,0 mit Längeneinheit Sekunden

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden
2. Ziffern 1,0,0,0 mit Längeneinheit Sekunden
3. 3 Stunden entspricht „3×1 Stunde“

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden
2. Ziffern 1,0,0,0 mit Längeneinheit Sekunden
3. 3 Stunden entspricht „3×1 Stunde“
4. 1000 Sekunden entspricht „$1 \cdot 10^3 \times 1$ Sekunden“

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden
2. Ziffern 1,0,0,0 mit Längeneinheit Sekunden
3. 3 Stunden entspricht „3×1 Stunde“
4. 1000 Sekunden entspricht „$1 \cdot 10^3 \times 1$ Sekunden“
5. länger ist eine Ordnungsrelation

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden
2. Ziffern 1,0,0,0 mit Längeneinheit Sekunden
3. 3 Stunden entspricht „3×1 Stunde“
4. 1000 Sekunden entspricht „$1 \cdot 10^3 \times 1$ Sekunden“
5. länger ist eine Ordnungsrelation
6. 3600 Sekunden entsprechen einer Stunde

Bezugssysteme

Aufgabenstellung

Wie kann man die Korrektheit der folgenden Aussage erkennen?

„3 Stunden sind länger als 1000 Sekunden“

Benötigte Informationen:

1. Ziffer 3 mit Längeneinheit Stunden
2. Ziffern 1,0,0,0 mit Längeneinheit Sekunden
3. 3 Stunden entspricht „3×1 Stunde“
4. 1000 Sekunden entspricht „$1 \cdot 10^3 \times 1$ Sekunden“
5. länger ist eine Ordnungsrelation
6. 3600 Sekunden entsprechen einer Stunde

Aussage ist wahr, da $10800 > 1000$

Nachricht und Interpretation

Nachricht

1100–111–01–0–10–01–11–010–101–010–1100–1–11–01

1. Zeichen
2. Buchstaben entschlüsseln:
3. In Wörter einteilen
4. Satz aus Wörtern zusammensetzen und Satzzeichen einfügen

Interpretationsvorschrift

H = 11	M = 0	E = 010	A = 10	C = 1
L = 101	T = 01	I = 1100	S = 111	– = Trennzeichen

Nachricht und Interpretation

Nachricht

1100–111–01–0–10–01–11–010–101–010–1100–1–11–01

1. Zeichen
2. Buchstaben entschlüsseln:
   I–S–T–M–A–T–H–E–L–E–I–C–H–T
3. In Wörter einteilen
4. Satz aus Wörtern zusammensetzen und Satzzeichen einfügen

Interpretationsvorschrift

H = 11	M = 0	E = 010	A = 10	C = 1
L = 101	T = 01	I = 1100	S = 111	– = Trennzeichen

Nachricht und Interpretation

Nachricht

1100–111–01–0–10–01–11–010–101–010–1100–1–11–01

1. Zeichen
2. Buchstaben entschlüsseln:
   I–S–T–M–A–T–H–E–L–E–I–C–H–T
3. In Wörter einteilen
   IST MATHE LEICHT
4. Satz aus Wörtern zusammensetzen und Satzzeichen einfügen

Interpretationsvorschrift

H = 11	M = 0	E = 010	A = 10	C = 1
L = 101	T = 01	I = 1100	S = 111	– = Trennzeichen

Nachricht und Interpretation

Nachricht

1100–111–01–0–10–01–11–010–101–010–1100–1–11–01

1. Zeichen
2. Buchstaben entschlüsseln:
   I–S–T–M–A–T–H–E–L–E–I–C–H–T
3. In Wörter einteilen
   IST MATHE LEICHT
4. Satz aus Wörtern zusammensetzen und Satzzeichen einfügen
   Ist Mathe leicht?

Interpretationsvorschrift

H = 11	M = 0	E = 010	A = 10	C = 1
L = 101	T = 01	I = 1100	S = 111	– = Trennzeichen

Lebendigkeit und Sichtbarkeit von Variablen

Variablen in Java (und den meisten anderen Programmiersprachen) sind nicht überall gleichermaßen verwendbar. Wir unterscheiden dabei zwei Eigenschaften:

Lebendigkeit Eine Variable nennt man an einer Stelle im Programm *lebendig*, wenn sie schon und noch im Speicher liegt.

Sichtbarkeit Eine Variable nennt man an einer Stelle im Programm *sichtbar*, wenn sie an der Stelle im Code verwendbar ist.

Sichtbarkeit $\Rightarrow$ Lebendigkeit

Lebendigkeit und Sichtbarkeit von Variablen

Variablen in Java (und den meisten anderen Programmiersprachen) sind nicht überall gleichermaßen verwendbar. Wir unterscheiden dabei zwei Eigenschaften:

Lebendigkeit Eine Variable nennt man an einer Stelle im Programm *lebendig*, wenn sie schon und noch im Speicher liegt.

Sichtbarkeit Eine Variable nennt man an einer Stelle im Programm *sichtbar*, wenn sie an der Stelle im Code verwendbar ist.

$$\text{Sichtbarkeit} \Rightarrow \text{Lebendigkeit}$$

Aufgabenstellung

Geben Sie an, welche Variablen an den mit `/* 1 */`, `/* 2 */` und `/* 3 */` gekennzeichneten Stellen im folgenden Programm leben und welche davon sichtbar sind. Begründen Sie Ihre Antwort. Verwenden Sie für die Begründung der Sichtbarkeit die aus der Vorlesung bekannte Darstellung. (Die Variable `args` soll nicht beachtet werden.)

Sichtbarkeit von Variablen

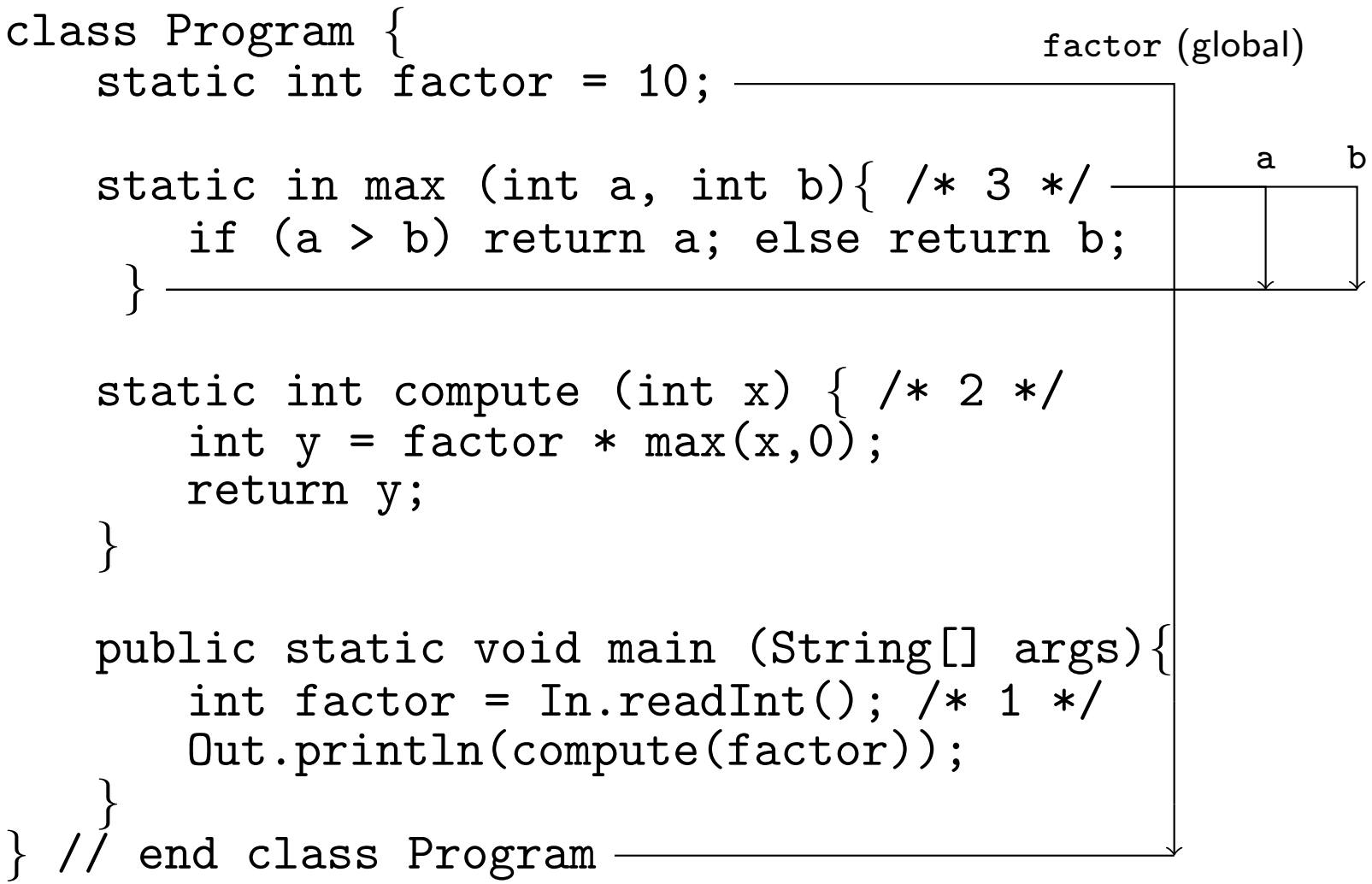
```
class Program {
    static int factor = 10;

    static in max (int a, int b){ /* 3 */
        if (a > b) return a; else return b;
     }

    static int compute (int x) { /* 2 */
        int y = factor * max(x,0);
        return y;
    }

    public static void main (String[] args){
        int factor = In.readInt(); /* 1 */
        Out.println(compute(factor));
    }
} // end class Program
```

Sichtbarkeit von Variablen

```
class Program {
    static int factor = 10;

    static in max (int a, int b){ /* 3 */
        if (a > b) return a; else return b;
     }

    static int compute (int x) { /* 2 */
        int y = factor * max(x,0);
        return y;
    }

    public static void main (String[] args){
        int factor = In.readInt(); /* 1 */
        Out.println(compute(factor));
    }
} // end class Program
```

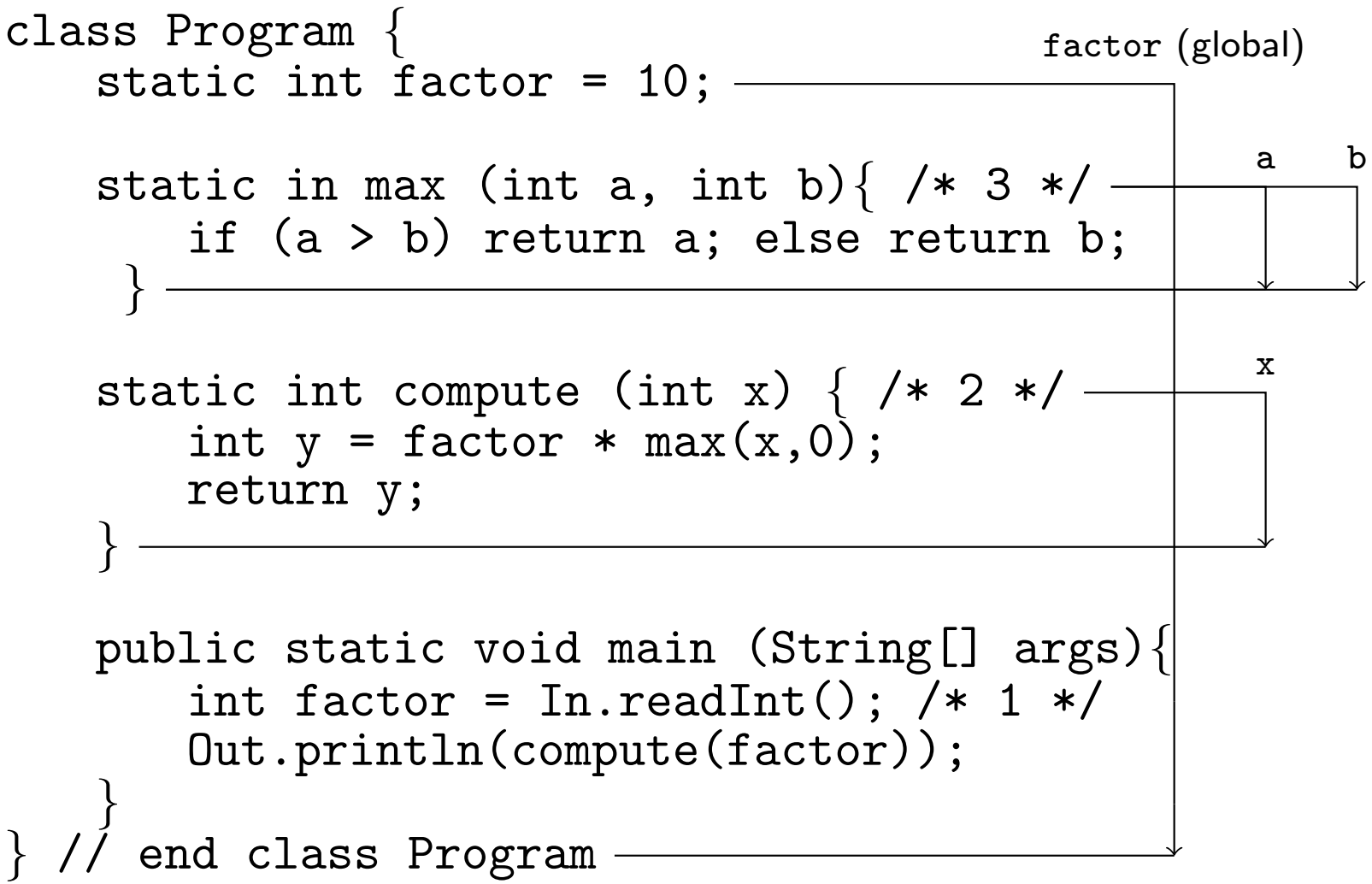
factor (global)

Sichtbarkeit von Variablen

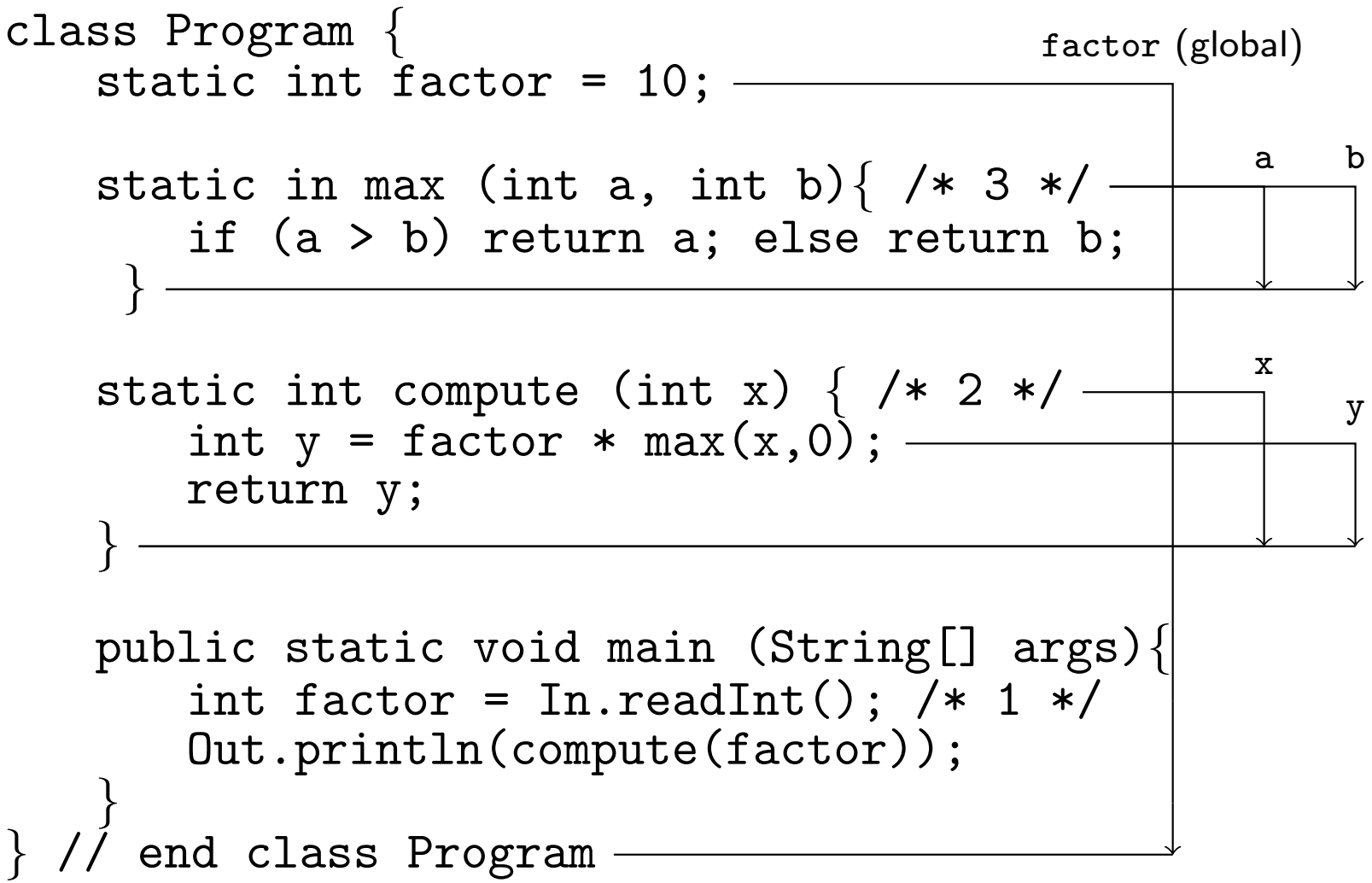



```
class Program {
    static int factor = 10;

    static in max (int a, int b){ /* 3 */
        if (a > b) return a; else return b;
     }

    static int compute (int x) { /* 2 */
        int y = factor * max(x,0);
        return y;
    }

    public static void main (String[] args){
        int factor = In.readInt(); /* 1 */
        Out.println(compute(factor));
    }
} // end class Program
```

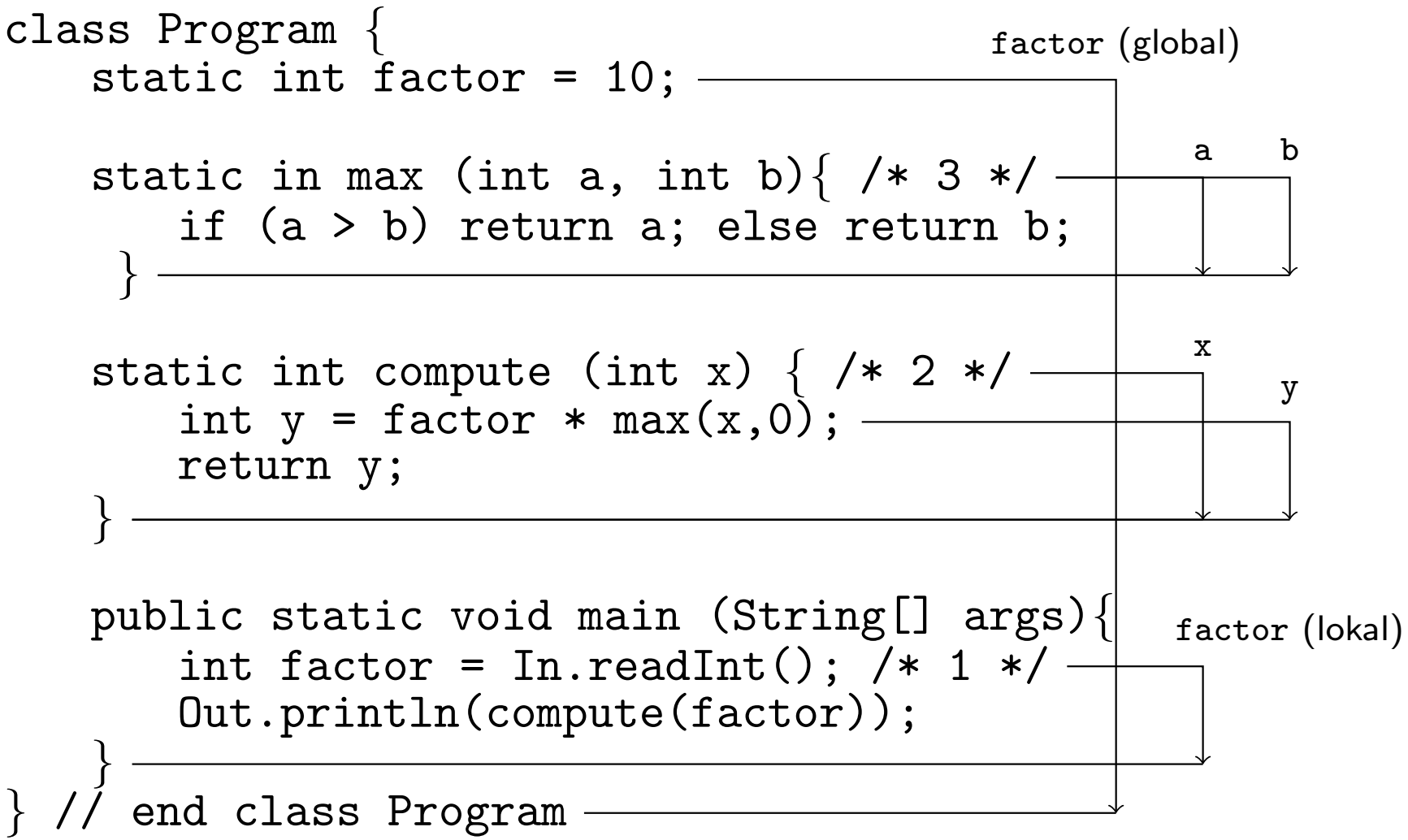
Sichtbarkeit von Variablen



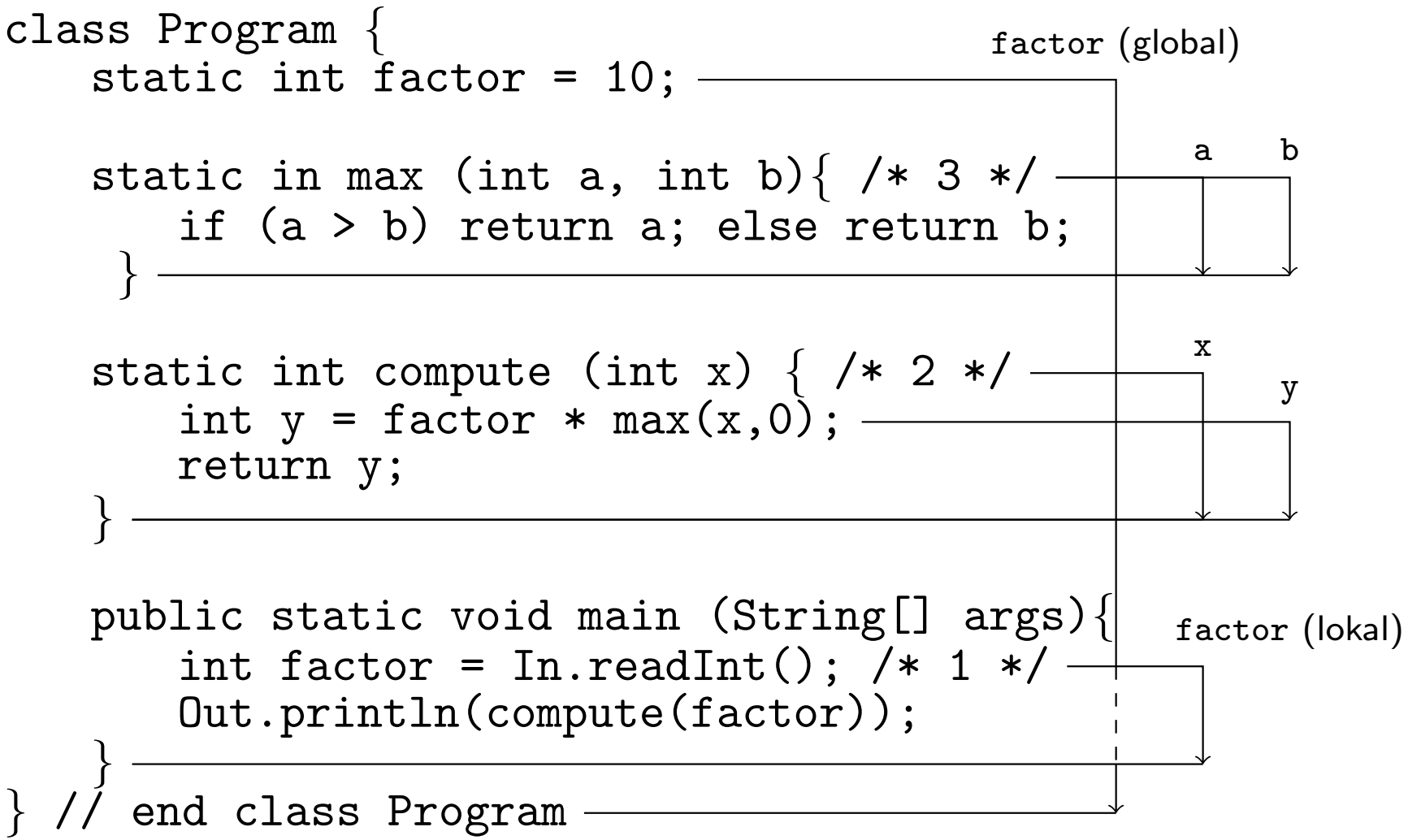
Sichtbarkeit von Variablen



Sichtbarkeit von Variablen



Sichtbarkeit von Variablen



Lebendigkeit von Variablen

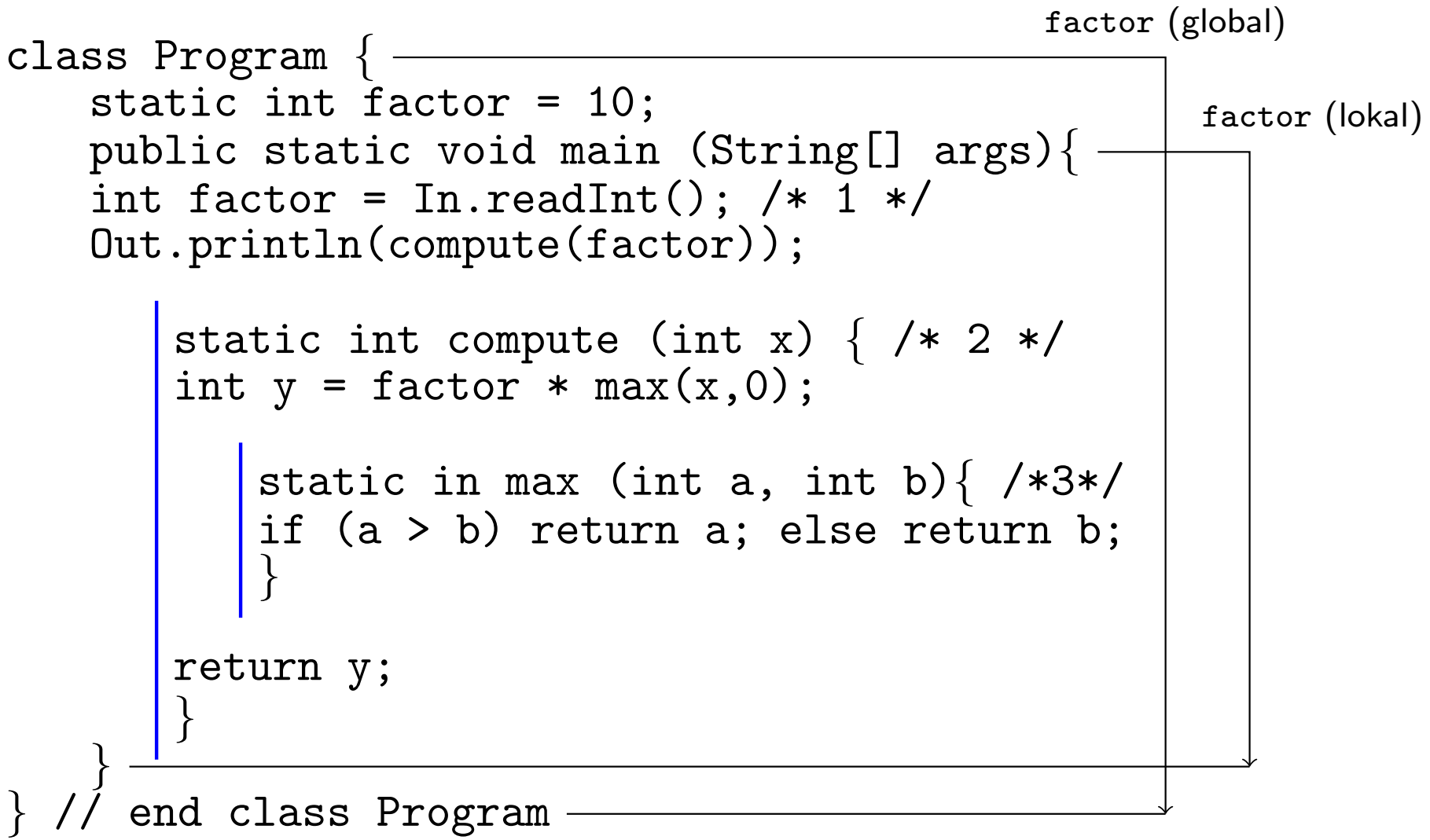
```
class Program {
    static int factor = 10;
    public static void main (String[] args){
    int factor = In.readInt(); /* 1 */
    Out.println(compute(factor));

        static int compute (int x) { /* 2 */
        int y = factor * max(x,0);

            static in max (int a, int b){ /*3*/
            if (a > b) return a; else return b;
            }

        return y;
        }
    }
} // end class Program
```

Lebendigkeit von Variablen

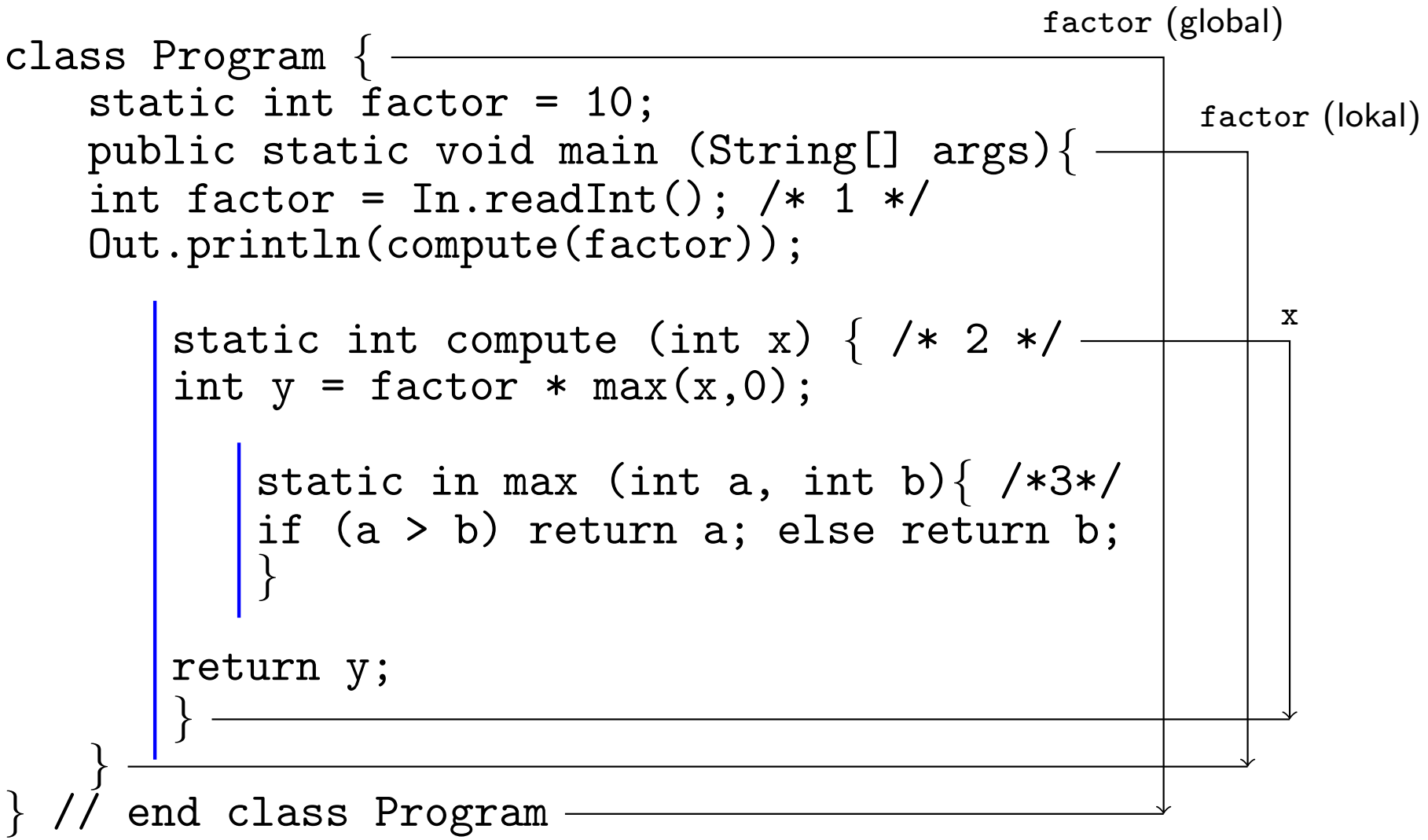
`factor` (global)

```
class Program {
    static int factor = 10;
    public static void main (String[] args){
    int factor = In.readInt(); /* 1 */
    Out.println(compute(factor));

        static int compute (int x) { /* 2 */
        int y = factor * max(x,0);

            static in max (int a, int b){ /*3*/
            if (a > b) return a; else return b;
            }

        return y;
        }
    }
} // end class Program
```

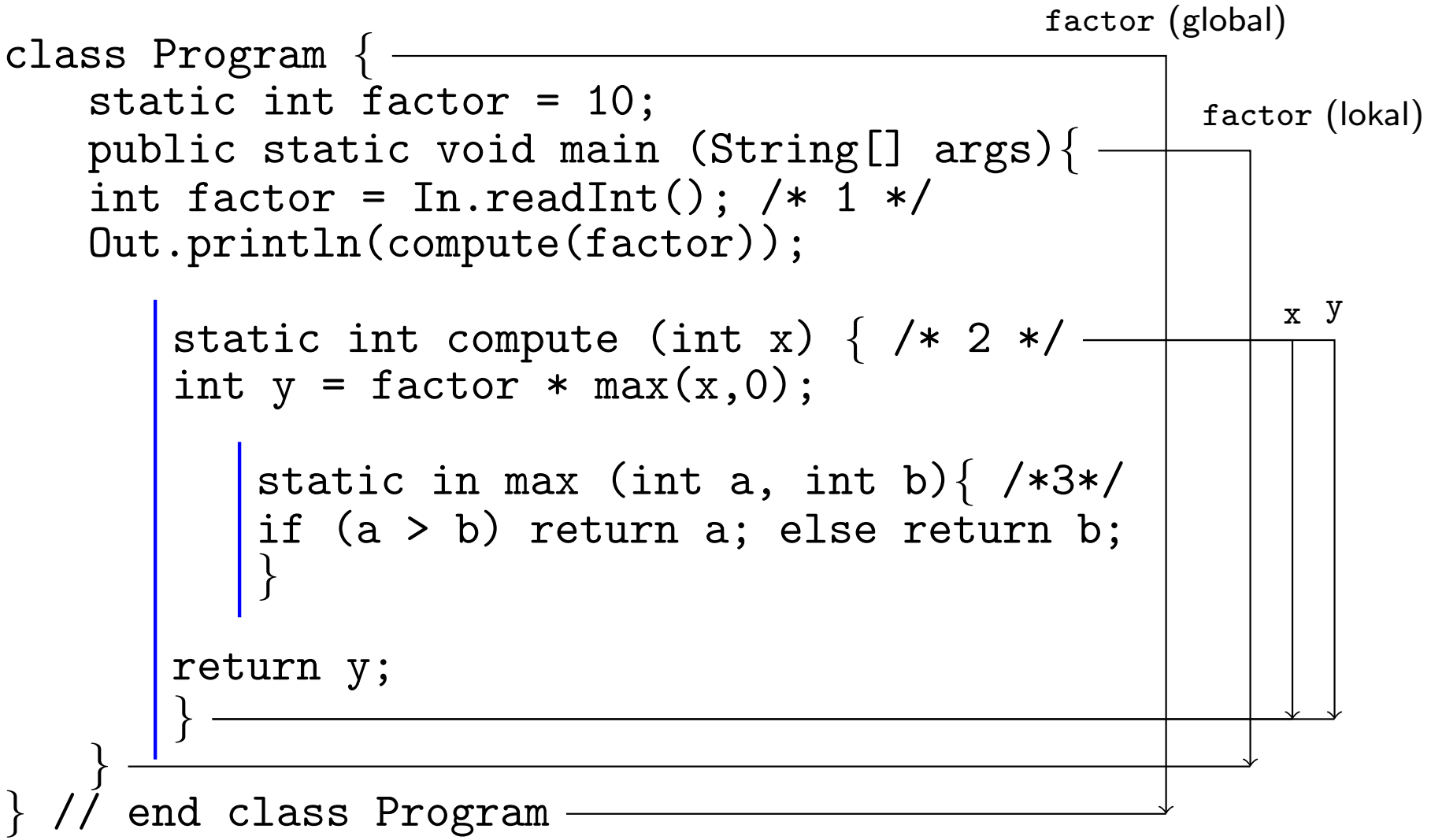
Lebendigkeit von Variablen



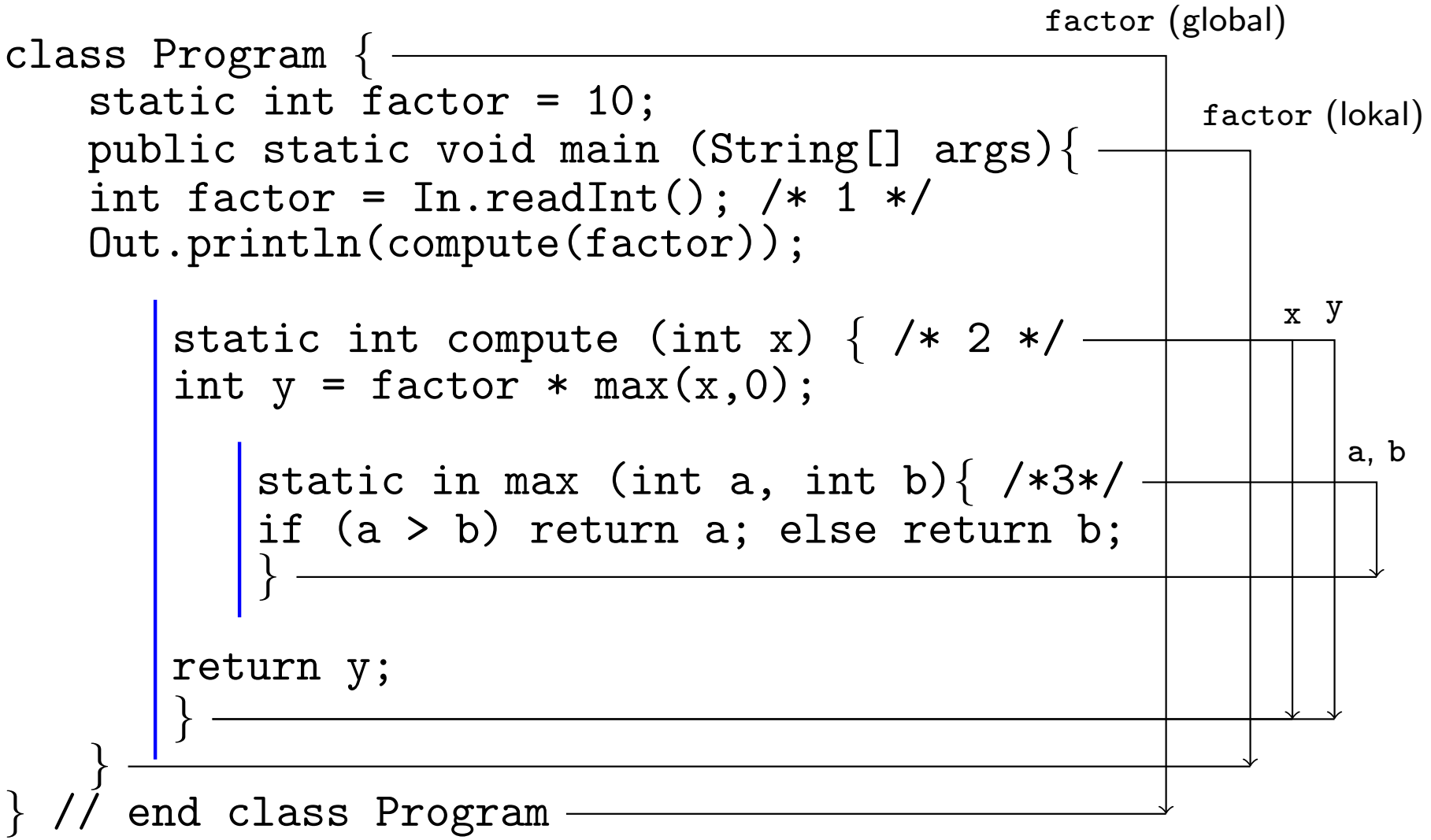
Lebendigkeit von Variablen



Lebendigkeit von Variablen



Lebendigkeit von Variablen



Lösung

	es leben	**es sind sichtbar**
`/* 1 */`	`factor` (global), `factor` (lokal)	`factor` (lokal)
`/* 2 */`	`factor` (gl.), `factor` (l.), `x`, `y`	`factor` (g.), `x`
`/* 3 */`	`factor` (gl.), `factor` (l.), `x`, `y`, `a`, `b`	`factor` (g.), `a`, `b`

An Stelle 1 leben nur die in der `main()`-Methode definierte lokale Variable `factor` sowie die globale Variable `factor`.
Da die Methode `compute()` von der Methode `main()` aufgerufen wird, leben an Stelle 2 neben den in der Methode selbst definierten Variablen `x` und `y` auch die in der Methode `main()` lebenden Variablen `factor` (lokal) und `factor` (global).
Da die Methode `max()` von der Methode `compute()` aufgerufen wird, leben an Stelle 3 neben den in der Methode selbst definierten Variablen `a` und `b` auch die in der Methode `compute()` lebenden Variablen `factor` (global), `factor` (lokal), `x` und `y`.

Finde die 10 Fehler!

```
public class fehlerklasse {
  public void main( String arg ) {
    int newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat );

    monkeydance = In.readInt;

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public void main( String arg ) {
    int newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat );

    monkeydance = In.readInt;

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String arg ) {
    int newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat );

    monkeydance = In.readInt;

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    int newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat );

    monkeydance = In.readInt;

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat );

    monkeydance = In.readInt;

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat" );

    monkeydance = In.readInt;

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat" );

    monkeydance = In.readInt();

    newval += monkeydance

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat" );

    monkeydance = In.readInt();

    newval += monkeydance;

    Out.println( "neuer Wert ist"  newval  "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat" );

    monkeydance = In.readInt();

    newval += monkeydance;

    Out.println( "neuer Wert ist" + newval + "\n" );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein maechtiger Pirat" );

    monkeydance = In.readInt();

    newval += monkeydance;

    Out.println( "neuer Wert ist " + newval );
  }
}
```

Finde die 10 Fehler!

```
public class Fehlerklasse {
  public static void main( String[] arg ) {
    float newval = 3,5f;
    int monkeydance;

    Out.println( "Ich bin Guybrush Threepwood und bin
        ein erbärmlicher Pirat" );

    monkeydance = In.readInt();

    newval += monkeydance;

    Out.println( "neuer Wert ist " + newval );
  }
}
```

Methoden

Methoden sind benannte Anweisungsfolgen, die...

- zur Zerlegung und Modularisierung eines Programmes dienen.
- parametrisiert oder auch parameterlos sein.
- werden als Funktion oder Prozedur bezeichnet.

Wer weiß es?

1. Was ist der Unterschied zwischen Funktionen und Prozeduren?
2. Was besagt das Schlüsselwort `void`?
3. Was bedetet folgende Methodendeklaration?

```
public static int max(int a, int b)
```

Methoden

Methoden sind benannte Anweisungsfolgen, die...

- zur Zerlegung und Modularisierung eines Programmes dienen.
- parametrisiert oder auch parameterlos sein.
- werden als Funktion oder Prozedur bezeichnet.

Wer weiß es?

1. Was ist der Unterschied zwischen Funktionen und Prozeduren?
2. Was besagt das Schlüsselwort `void`?
3. Was bedetet folgende Methodendeklaration?
   ```
   public static int max(int a, int b)
   ```

Namenskonventionen

Namenskonventionen zu Methodennamen

- Mit einem Kleinbuchstaben beginnen
- Mit einem Verb beginnen
- Falls aus mehreren Wörtern gebildet, beginnen die Folgewörter mit einem Großbuchstaben.

Beispiele: `add()`, `getFileFromInternet()`, `calcSqrRoot()`

Namenskonventionen zu Klassennamen

- Mit einem Großbuchstaben beginnen
- Falls aus mehreren Wörtern gebildet, beginnen die Folgewörter mit einem Großbuchstaben.

Beispiele: `NumGuesser`, `HelloWorld`, `IPO`, `Quake3`

Schleifen

Ziel

Gewisse Berechnungen sollen wiederholt ausgeführt werden, bis eine gewisse Bedingunge eintritt

Dies kann man mit einer `while`-Schleife (Abweisschleife) erreichen.

Syntax der `while`-Schleife

- `while (<Schleifenbedingung>) <Schleifenrumpf>`
- als EBNF-Regel:
  `WhileStatement = "while" "(" Expr ")" Statement.`
- Durch Bildung eines Blockes können beliebig viele Anweisungen den Schleifenrumpf bilden.

Schleifen

Ziel

Gewisse Berechnungen sollen wiederholt ausgeführt werden, bis eine gewisse Bedingunge eintritt

Dies kann man mit einer `while`-Schleife (Abweisschleife) erreichen.

Syntax der `while`-Schleife

- `while (`<Schleifenbedingung>`)` <Schleifenrumpf>
- als EBNF-Regel:
  `WhileStatement = "while" "(" Expr ")" Statement.`
- Durch Bildung eines Blockes können beliebig viele Anweisungen den Schleifenrumpf bilden.

Beispielaufgabe: Kubikwurzel

Erstellen Sie ein Java-Programm, welches die Kubikwurzel der `float`-Zahl x berechnet. Die Berechnung soll mittels der Näherungsformel von Newton erfolgen:

$$y_i = \frac{2y_{i-1} + \frac{x}{y_{i-1}^2}}{3}$$

y_i ist der Näherungswert der Kubikwurzel nach dem i-ten Schritt. Beginnen Sie mit $y_0 = \frac{x}{3}$. Die Iteration soll so lange fortgesetzt werden, bis sich y_i und y_{i-1} um weniger als 10^{-5} unterscheiden. Prüfen Sie diese Bedingung mittels
`Math.abs(`y_i `-` y_{i-1}`) < 1.0E-5`.

Beispiellösung: Kubikwurzel

```
class Root {
  static float calcRoot3(float x) {          // this function returns the cubic
                                             // root of a given float value
    float y1 = x / 3;                        // start with y1 == x / 3
    float y0 = 0;                            // and with y0 == 0
    while (Math.abs(y0 - y1) > 1.0E-5) {     // loop until |y0 - y1| < 1.0E-5
      y0 = y1;                               // set y0 the old y1
      y1 = (2 * y0 + x / (y0 * y0)) / 3;     // calculate a new y1
    }
    return y1;                               // return result
  }

  public static void main(String[] arg) {
    Out.print("Geben Sie eine Zahl ein: ");// Print text to the console
    float x = In.readFloat();                // read the value x from the user
    Out.println("Kubikwurzel von " + x       // print the cubic root
    + " ist " + calcRoot3(x);                // using the function root3
  }
}
```

Fragen

Fragen?


MODEHAUS CARL SCHUPP
DRESDNER BANK