hhu.

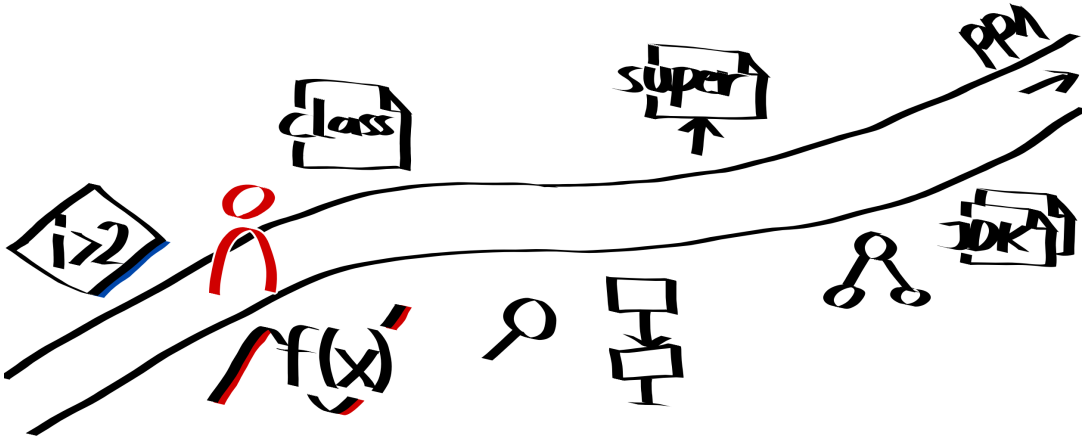
Kapitel 2: Methoden & IO

VL 7: Schreiben eigener Methoden

Wintersemester 2023/24

Stand 31. Oktober 2023, 06:39 Uhr

Wo stehen wir gerade?



Vorlesungsfeedback

Methoden[1]

- Status Quo: Alle unsere Befehle stehen in main-Methode
- Probleme:
  - keine Wiederverwendbarkeit
  - schlechte Übersicht bei langem Code

[1] Methods; in anderen Programmiersprache sind auch die Begriffe *Funktion* und *Prozedur* zu finden

Vorbild: Funktionen in der Mathematik I

- $f(x) = 2\sin(x^2) \cdot x$ definiert wiederverwendbare Funktion
- Funktionswerte für verschiedene Eingaben berechenbar: $f(0)$ ist 0, $f(1)$ ist 1,68 ...

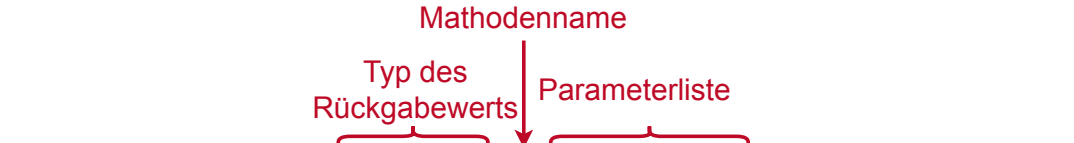
Example.java — Java

```java
public class Example {

    private static double f(double x) {
        return 2 * Math.sin(x * x) * x;
    }

    public static void main(String[] args) {
        System.out.println(f(0));
        System.out.println(f(1));
    }

}
```

Vorbild: Funktionen in der Mathematik II

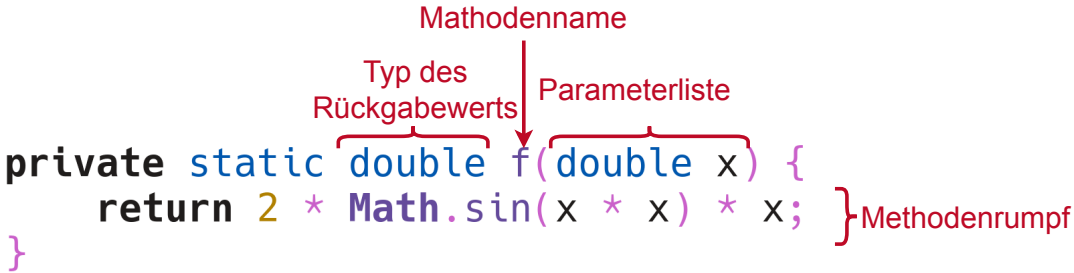

```
% java Example
0.0
1.682941969615793
```

Aufbau einer Methode I



- Signatur (Methodenname + Typ der Parameter): `f(double)`
  - einmalig pro ~Datei
- Typ des Rückgabewerts: `double`
- Parameterliste: `double x`
  - nur innerhalb der Methode sichtbar
  - unabhängig von Variablen mit gleichem Namen an anderen Stellen

Aufbau einer Methode II



- `return`
  - definiert Wert der folgenden Expression als Rückgabewert
  - beendet Methode an dieser Stelle & kehrt zum Aufrufort zurück
- `private static` → später
  - fürs Erste Methoden (außer main) immer mit `private static`

Weitere Anmerkungen zu Methoden

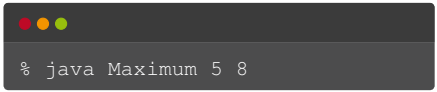
- beliebig viele Parameter mit beliebigen Typen möglich
  - auch unterschiedliche Typen
  - auch Array-Typen
- Methodenrumpf kann beliebige Abfolge von Befehlen beinhalten
  - auch Aufrufe anderer (oder derselben$^{\rightarrow \text{später}}$) Methoden

```
1 private static void printMaximum(int a, int b) {
2     if(a > b) {
3         System.out.println(a);
4         return;
5     }
6     System.out.println(b);
7 }
```

Kontrollfluss bei Methodenaufrufen

Heinrich Heine Universität Düsseldorf

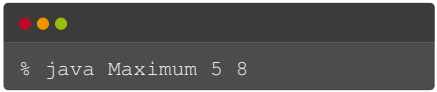
```java
public class Maximum {

    private static int maximum(int a, int b) {
        if (a > b) {
            return a;
        }
        return b;
    }

    public static void main(String[] args) {
        int number1 = Integer.parseInt(args[0]);
        int number2 = Integer.parseInt(args[1]);

        int result = maximum(number1, number2);

        System.out.println(result);
    }

}
```



Kontrollfluss bei Methodenaufrufen

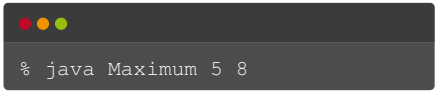
Heinrich Heine Universität Düsseldorf

```
public class Maximum {

    private static int maximum(int a, int b) {
        if (a > b) {
            return a;
        }
        return b;
    }

    public static void main(String[] args) {
        int number1 = Integer.parseInt(args[0]);
        int number2 = Integer.parseInt(args[1]);

        int result = maximum(number1, number2);

        System.out.println(result);
    }

}
```



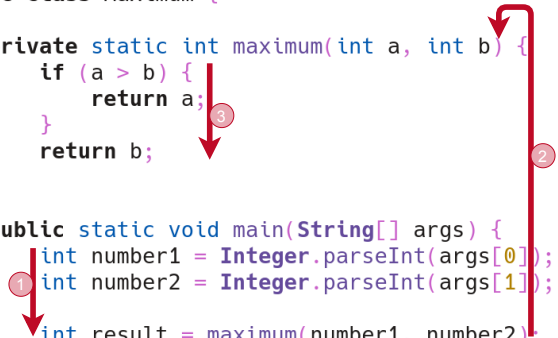
Kontrollfluss bei Methodenaufrufen


```
public class Maximum {

    private static int maximum(int a, int b) {
        if (a > b) {
            return a;
        }
        return b;
    }

    public static void main(String[] args) {
        int number1 = Integer.parseInt(args[0]);
        int number2 = Integer.parseInt(args[1]);

        int result = maximum(number1, number2);

        System.out.println(result);
    }

}
```



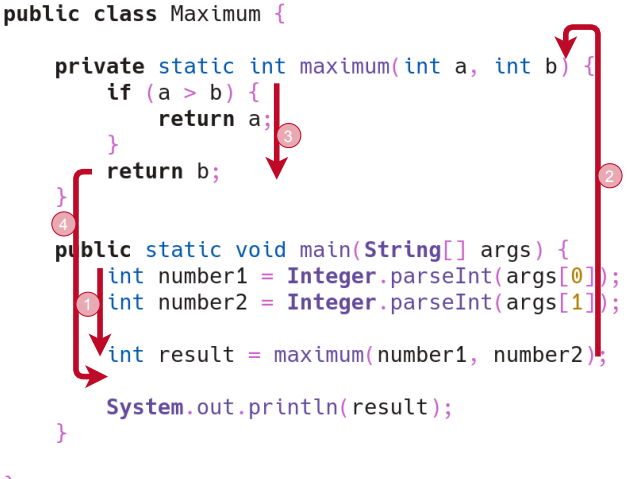
Kontrollfluss bei Methodenaufrufen


```
public class Maximum {

    private static int maximum(int a, int b) {
        if (a > b) {
            return a;
        }
        return b;
    }

    public static void main(String[] args) {
        int number1 = Integer.parseInt(args[0]);
        int number2 = Integer.parseInt(args[1]);

        int result = maximum(number1, number2);

        System.out.println(result);
    }

}
```



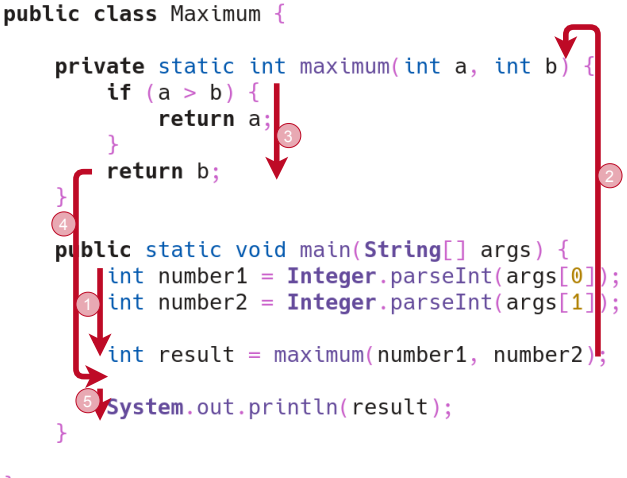

```
% java Maximum 5 8
```

Kontrollfluss bei Methodenaufrufen




```
% java Maximum 5 8
```

Kontrollfluss bei Methodenaufrufen




```
% java Maximum 5 8
8
```

Der Rückgabetyp void

- `void` = keine Rückgabe (nur Nebeneffekte)
- kein `return` notwendig
- `return;` bricht Methode vorzeitig ab (ohne Rückgabewert)

```java
private static void printMaximum(int a, int b) {
    if(a > b) {
        System.out.println(a);
        return;
    }
    System.out.println(b);
}
```

main-Methode revisited

Heinrich Heine Universität Düsseldorf

```
public static void main(String[] args)
```

- main-Methode für Compiler „nur“ ganz normale Methode
- `void`: kein Rückgabewert
- `String[] args`: Array von Zeichenketten$^{\rightarrow \text{später}}$ als Parameter (von JVM mit Konsolenargumenten befüllt)
- erste Methode, die von JVM ausgeführt wird

Fremde Methoden . . .

Heinrich Heine Universität Düsseldorf

. . . haben wir die ganze Zeit schon benutzt

- ohne Parameter: `Math.random()`
- ohne Rückgabewert: `System.out.println(42)`
- mit Rückgabewert und einem Parameter: `Math.sqrt(4)`
- mit Rückgabewert und zwei Parametern: `Math.max(1, 3)`
- Vorteil:
  - „Standardaufgaben“ nicht jedes Mal selbst implementieren
  - wir müssen nur wissen, *was* eine Methode tut, nicht *wie*
  - (hoffentlich) intensiv getestet
  - *Implement once (correctly), then forget how you did it and use it everywhere.*

StdDraw

Heinrich Heine Universität Düsseldorf

- Wir können auch Methoden verwenden, die andere Menschen geschrieben haben und in anderen java-Dateien liegen.
- Fallbeispiel: StdDraw
  - anderer Mensch = Prof. Wayne
- Motivation: (relativ) einfach grafische Ausgaben
  - Methoden in StdDraw kümmern sich um Anzeigen von Fenster usw.
  - das *Wie* interessiert uns (heute) nicht
- Verwendung:
  - Datei `StdDraw.java` (siehe Ilias) muss neben eigener java-Datei liegen
  - Methodenaufruf: `StdDraw.Methodenname`

Koordinatensystem

- Arbeiten mit Bildpunkten (Pixel)
- Pixel hat x- und y-Koordinate
- unten links: (0,0)
- `setCanvasSize(int width, int height)`
  setzt Größe der Zeichenfläche (in Pixel)
- `setXScale(double x0, double x1)`
  definiert Koordinatenbereich für x-Achse

Koordinatensystem: Standardskalierung

Heinrich Heine Universität Düsseldorf

Standard-Zeichenfläche:


```
1 StdDraw.setPenRadius(0.01);
2 StdDraw.point(0.1, 0.9);
3 StdDraw.line(0, 0, 0.9, 0.9);
```

Koordinatensystem: Benutzerdefinierte Skalierung

Heinrich Heine Universität Düsseldorf

Zeichenfläche mit Achsen von 0 bis 10


```
1  StdDraw.setScale(0, 10);
2
3  StdDraw.setPenRadius(0.01);
4  StdDraw.point(0.1, 0.9);
5  StdDraw.line(0, 0, 0.9, 0.9);
```

Application Programming Interface

Heinrich Heine Universität Düsseldorf

Definition

Die **API** ist die Summe aller öffentlicher Methoden und Konstanten einer Klasse oder eines Interfaces.

für jetzt:

- Die API besteht aus Methoden, die andere Menschen geschriebene haben und die ich in meinem eigenen Programm benutzen kann.

API von StdDraw (Auszug) I

- `void line(double x0, double y0, double x1, double y1)`
  Linie von (x0,y0) nach (x1,x2) zeichnen
- `void point(double x, double y)`
  Punkt bei (x,y) zeichnen
- `void circle(double x, double y, double r)`
  `void filledCircle(double x, double y, double r)`
  (ausgefüllten) Kreis mit Radius r und Mittelpunkt bei (x,y) zeichnen
- `void square(double x, double y, double r)`
  `void filledSquare(double x, double y, double r)`
  (ausgefülltes) Quadrat mit halber (sic!) Kantenlänge r und Mittelpunkt bei (x,y) zeichnen
- `void polygon(double x[], double y[])`
  `void filledPolygon(double x[], double y[])`
  (ausgefülltes) Polygon mit Eckpunkten (x[0],y[0]), (x[1],y[1]), . . . , (x[0],y[0]) zeichnen

API von StdDraw (Auszug) II

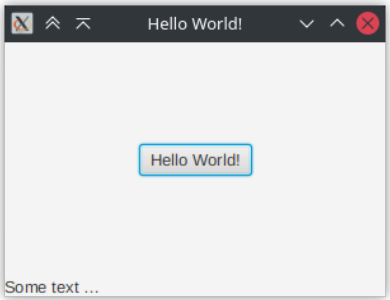
Heinrich Heine Universität Düsseldorf

mehr Details:

- in den Kommentaren in der Datei `StdDraw.java`
- `https://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html`

Praxis: JavaFx, Weboberflächen

- JavaFx: Grafische Benutzeroberflächen mit Fenstern, Buttons etc.



- Spring Boot: Webanwendungen, die im Browser angezeigt werden
  → Programmierpraktikum

Lernziele

Heinrich Heine
Universität Düsseldorf

Sie können am Ende der Woche ...

- eigene Methoden **schreiben**.
- Methoden mit vorgegebener Signatur und Verhalten **schreiben**.
- vorgegebene Methoden aus anderen Java-Dateien **verwenden**.
- die Funktionalität einer Methode anhand ihrer Dokumentation **herausfinden**.

Wichtige Begriffe

Heinrich Heine
Universität Düsseldorf

Methode
Rückgabewert
API

Signatur
return

Parameter
void

Chaos Game

Heinrich Heine Universität Düsseldorf

Gegeben sei ein gleichseitiges Dreieck mit den Eckpunkten *A*, *B* und *C*.
Spiele folgendes Spiel:

1. Starte beim Punkt *A*.
2. Wiederhole:
   1. Wähle einen zufälligen Eckpunkt.
   2. Bestimmte den Punkt, der zwischen diesem Eckpunkt und dem zuletzt gewählten Punkt liegt.
   3. Markiere den Punkt.

Welches Muster entsteht?