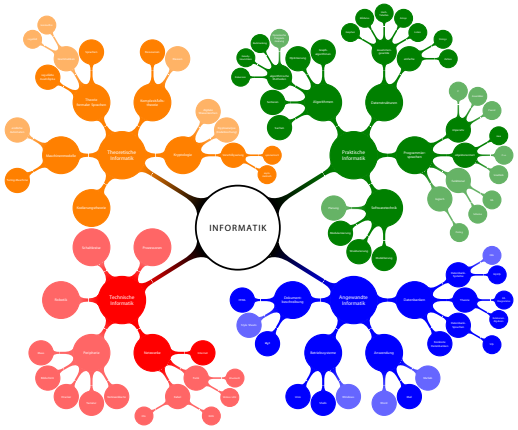
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Kapitel 23

Objekte – Methoden und Nachrichten

Objekte beim Tratschen

Vorlesung Einführung in die Informatik 1 vom 23. Januar 2014 von Till Tantau



Lernziele von Kapitel 23

1. Konzept der Nachricht kennen
2. Nichtstatische Methoden implementieren können
3. Konstruktoren kennen und implementieren können

Gliederung von Kapitel 23

Wiederholung: Die zentralen Eigenschaften von Objekten.

1. Objekte haben eine Identität.
2. Objekte haben Attribute.
   - Attribute sind Eigenschaften wie Höhe, Breite, Oberflächenstruktur oder was auch immer,
   - die man benutzen und verändern kann.
3. *Objekte haben Fähigkeiten.*
   - Man kann Objekten *Nachrichten schicken*.
   - Dazu werden *Objekt-Methoden* benutzt.

Wiederholung: Beispiele


```
// In der Datei Student.java
class Student
{
  String nachname, vorname;
  int matrikelnummer;
}

// In der Datei Color.java
class Color
{
  float red_part;   // Rot-Anteil; Zahl zwischen 0 und 1
  float green_part; // Grün-Anteil
  float blue_part;  // Blau-Anteil
}
```

Objekte tauschen Nachrichten aus.

- Objekte können anderen Objekten *Nachrichten* schicken.
- Objekte können die *Fähigkeit haben*, auf bestimmte Nachrichten zu *reagieren*.
- Der Ablauf einer Nachricht ist folgender:
  1. Objekt A schickt *eine Nachricht an Objekt B*. Objekt A *wartet* nun auf Antwort.
  2. Objekt B hat die *Fähigkeit*, auf die Nachricht zu reagieren. Dazu führt es *Berechnungen* durch.
  3. Objekt B schickt *eine Antwort zurück*.
  4. Objekt A kann nun *weiter rechnen*.

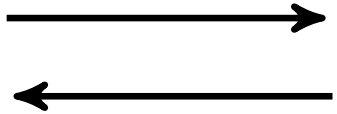
Beispiel eines Nachrichtenaustauschs.

- Wir haben ein Antigen-Objekt und ein Antikörper-Objekt.
- Das Antigen-Objekt möchte vom Antikörper-Objekt wissen, wie hoch die Affinität von Epitop und Paratop ist.
- Dazu schickt es eine Nachricht.

Cholera-Antigen

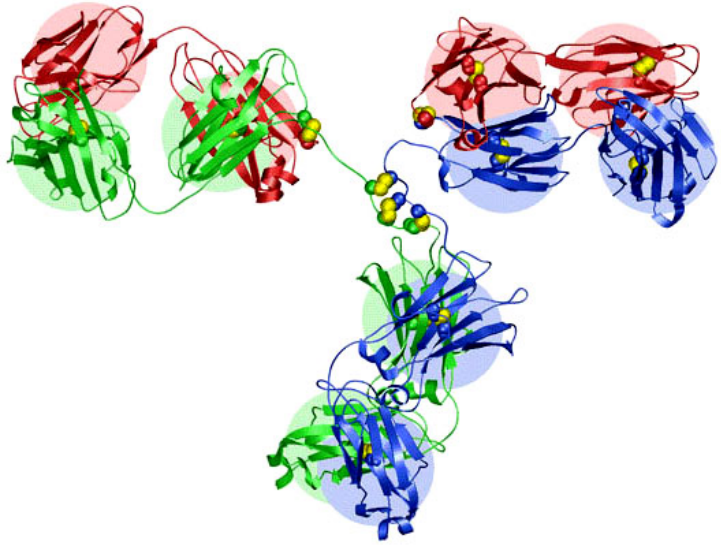
Unknown author, GNU Free Documentation License

1. Passt Dein Paratop zu meinem Epitop?



3. Antwort: »Nee«

Antiköper



Author David Goodsell, Scripps Research Institute, Public Domain

2. Berechnet Affinität

Schritt 1: Abschicken einer Nachricht.

- Wir wollen dem Objekt `main_antibody` (eine Instanz der Klasse `Antibody`) Nachrichten schicken.
- Die erste Nachricht heißt `moveUp` und soll dem Antikörper bitten, sich nach oben zu bewegen.
- Die zweite Nachricht heißt `doYouMatch` und fragt den Antikörper, ob sein Paratop zu einem gegebenen Epitop passt. Diese Nachricht bekommt noch einen *Parameter*, nämlich das Epitop.

Syntax der Nachrichtenverschickungen

```
main_antibody . moveUp ()
```

Variable, die auf ein Objekt verweist — Nachrichtenname

```
main_antibody . doYouMatch ("0010101101")
```

Variable, die auf ein Objekt verweist — Nachrichtenname — Parameter

Beispielcode, in dem Nachrichten verschickt werden.


```
...
// Irgendwo in einem Programm

// Erzeuge neuen Antikörper
Antibody main_antibody = new Antibody ();

// Setze einige Attribute
main_antibody.x = 5;
main_antibody.y = 6;
main_antibody.paratope = "0010101101";

// Schicke main_antibody die Nachricht moveUp
main_antibody.moveUp ();

// Schicke main_antibody nochmal die Nachricht moveUp
main_antibody.moveUp ();

// Schicke main_antibody die Nachricht doYouMatch
if (main_antibody.doYouMatch ("1101010010")) {
  System.out.println ("you match");
}
```

5-Minuten-Aufgabe

Zur Übung

Finden Sie alle Stellen, wo einem Objekt eine Nachricht geschickt wird:

```
Antibody a = new Antibody ();
String s = "00101000101";

a.x = Math.sqrt (9);
a.x = a.y;

a.moveUp ();

if (s.length () > 5 && a.doYouMatch (s)) {
  System.out.println ("Schwierig");
}
```

Schritt 2: Verarbeitung der Nachricht.

Idee

- Das Nachrichtenprotokoll folgt EVA: Das Objekt erhält eine *Eingabe* (die Nachricht), es führt eine *Verarbeitung* (Berechnung) durch und es liefert eine *Ausgabe*.
- Nun galt: *»Methoden sind kleine EVAs.«*
- Deshalb benutzen wir *besondere Methoden, um Nachrichten zu verarbeiten*.

Syntax der Methoden zur Nachrichtenverarbeitung

- `static` *fehlt*.
- Der Methoden-Code wird »vom Objekt ausgeführt«.
- Im Code verweist die Variable `this` auf das Objekt.
- Der Rückgabewert der Methode ist gerade der Wert, der an den Absender der Nachricht zurückgesendet wird.

Beispielcode, der die Nachrichten verarbeitet.


```
class Antibody {
  // Attribute
  int x, y;
  String paratope;

  // Erste Objekt-Methode
  void moveUp ()
  {
    this.y = this.y + 1;
  }

  // Zweite Objekt-Methode
  boolean doYouMatch (String epitope)
  {
    if (this.paratope.equals(epitope)) {
      return true;
    }
    else {
      return false;
    }
  }
}
```

5-Minuten-Aufgabe

Zur Übung

Ein *Zeit-Objekt* speichert eine Uhrzeit, bestehend aus Stunden, Minuten und Sekunden:

```
class Time
{
  int hours;
  int minutes;
  int seconds;
}
```

Geben Sie den Code zweier Methoden an, die auf folgende Nachrichten reagieren:

1. Im Objekt gespeicherte Zeit soll auf Mitternacht zurückgesetzt werden.
2. Im Objekt gespeicherte Zeit soll um eine Minute vorgestellt werden.

Beispielablauf einer Kommunikation.


```
Studi a = new Studi ();
Studi b = new Studi ();
Studi c = new Studi ();

a.name = "...";
b.name = "...";
c.name = "...";

a.sprich("Hallo.");
a.vorstellung ();

b.sprich("Hallo␣"+a.name);
b.vorstellung ();

c.vorstellung_von (b);
```

```
class Studi {
  String name;

  void sprich (String text)
  {
    System.out.println (text);
  }

  void vorstellung ()
  {
    this.sprich
      ("Ich␣bin␣" + this.name);
  }

  void vorstellung_von(Studi s)
  {
    s.sprich
      ("Das␣ist␣" + this.name);
  }
}
```

Schritt 3: Antworten auf eine Nachricht.

- Ein Objekt kann auf eine Nachricht *antworten*, indem es am Ende einer Methode einen Wert mit `return` zurückgibt.
- An dem Ort, an dem die Nachricht verschickt wurde, geht die Berechnung erst weiter, wenn die Antwort angekommen ist.
- Soll auf eine Nachricht nicht geantwortet werden, so gibt man als Typ `void` an.
  Auch in diesem Fall geht die Berechnung erst weiter, wenn die Methode fertig abgearbeitet wurde.

Konstruktoren dienen der Bequemlichkeit.

Die Erzeugung eines Objektes geschieht oft in zwei Phasen:

1. Das Objekt wird mittels `a = new Antibody ()` erzeugt.
2. Die Attribute des Objektes werden mittels `a.x = 0; a.y = 0;` auf Startwerte gesetzt.

Dieses Verfahren lässt sich mit *Konstruktoren* vereinfachen:

- Konstruktoren sind spezielle Methoden,
- die unmittelbar nach der Erzeugung eines Objektes automatisch einmalig aufgerufen werden.
- Ihre Aufgabe ist es, die Attribute des Objektes auf sinnvolle Anfangswerte zu setzen.

Syntax von Konstruktoren.

- Der Name des Konstruktors einer Klasse wie `Antibody` ist der *Name der Klasse* (also wiederum `Antibody`).
- Er hat *keinen Rückgabewert*, noch nicht einmal **`void`**.
- Man ruft den Konstruktor nicht explizit auf, sondern er wird automatisch aufgerufen, wenn ein Objekt mittels **`new`** erzeugt wurde.

Beispiel

```
class Antibody {
  int x,y;
  String paratope;

  // Konstruktor:
  Antibody ()
  {
    this.x = 0;
    this.y = 0;
  }
}
```

```
...
Antibody a, b;

// Noch nichts passiert

a = new Antibody ();
// Jetzt wurde der
// Konstruktor
// aufgerufen

b = new Antibody ();
// Jetzt nochmal,
// aber für das
// zweite Objekt
```

Syntax von Konstruktoren mit Parametern.

- Konstruktoren können auch Parameter haben,
- deren konkrete Werte bei **new** mit angegeben werden.
- Eine Klasse kann mehrere Konstruktoren haben, die sich dann aber in den Parametern unterscheiden müssen. Der richtige Konstruktor wird automatisch ausgewählt.

Beispiel

```
class Antibody {
  int x,y;
  String paratope;

  // Konstruktor:
  Antibody
    (int a, int b)
  {
    this.x = a;
    this.y = b;
  }
}
```

```
...
Antibody a;

// Noch nichts passiert

a = new Antibody (2,3);
// Jetzt wurde der
// Konstruktor
// aufgerufen

// Jetzt gilt
// a.x == 2 und a.y == 3
```

5-Minuten-Aufgabe

Zur Übung

Entwerfen Sie zwei sinnvolle Konstruktoren für die Zeit-Klasse.

```
class Time
{
  int hours;
  int minutes;
  int seconds;
}
```

Zusammenfassung

Syntax für den Aufruf von Objektmethoden (= Nachricht versenden)

```
some_object.methodName(parameter1,parameter2,...);
```

Syntax für Objektmethoden

```
return_type methodName (typ1 parameter1, typ2 parameter2,
    ...)
{
  // Körper, in dem auf this zugegriffen werden kann
  return some_value;
}
```

Syntax für Konstruktoren

```
ClassName (typ1 parameter1, typ2 parameter2, ...)
{
  // Körper, in dem auf this zugegriffen werden kann
}
```

Zusammenfassung

Welche statische Methode einer nichtstatischen entspricht

Statt

```
class Antigen {
  int x,y;
  void moveUp (int how_far) {
    this.y = this.y + how_far;
  }
}
...
Antigen a = new Antigen ();
a.moveUp();
```

könnte man auch schreiben:

```
class Antigen {
  int x,y;
  static void moveUp (Antigen that, int how_far) {
    that.y = that.y + how_far;
  }
}
...
Antigen a = new Antigen ();
Antigen.moveUp(a);
```