Objects first mit BlueJ und Greenfoot

Eine projektorientierte Unterrichtssequenz zum Einstieg in die objektorientierte Programmierung

Thomas Karp

Objects first?

- OOP ist wichtig! „Echte“ Programme werden fast immer objektorientiert geschrieben.
- Argumente dagegen
  - public static void main(String[] args)
    - Durch BlueJ sehr spät, wenn überhaupt, notwendig
  - Macht nur für große Probleme Sinn
    - Bei Auswahl geeigneter Beispiele nicht
  - Komplex
    - Klassen und Objekte entsprechen unseren Denkmustern mehr als Hauptprogramme , Unterprogramme und Variablen → Wahl geeigneter Beispiele
  - Großer Overhead notwendig (Main-Methode etc.)
    - Durch interaktives Aufrufen in BlueJ nicht!
    - Durch Codepad/Direkteingabe bietet BlueJ quasi „Java-Interpreter“

Objects first?

„Soweit ich das nun anhand der Recherche beurteilen kann, gibt es kaum mehr Gründe,

nicht mit Objekten einen Programmierkurs zu beginnen. „Objects First“ ist eines von

drei wichtigen Konzepten [Coo03, 191] für die Einführung in die Software-Entwicklung.

Deshalb ist es auch legitim, dieses Konzept zu wählen. Wichtig ist meines Erachtens nur,

dass zu Beginn mit Hilfe von verschiedenen Werkzeugen, wie Karel the Robot, BlueJ,

Alice,… die Konzepte den Studenten oder Schülern nähergebracht werden, bevor sie

selbständig Code schreiben.“

Aus: Lernansatz: „Objects First” von Alexander Jäger unter Leitung von Prof. Dr. Peter Hubwieser

Unterrichtsmaterial

- Alte Version: http://infoskript.de
- Neue Version in Arbeit: http://inf-schule.de/(index.php?version=10&seite=informatik/oopjava)
- Projektorientierte Aufgaben
  - Jedes Programm soll „etwas Richtiges tun“
  - Keine Offenheit im Sinne von Projektunterricht
  - Meistens Erweiterungen möglich/gewünscht
- Wechsel zwischen BlueJ und Greenfoot
  - Greenfoot-Programm oft ansprechender
  - Nur Greenfoot erzeugt falsche Sicht (Ausschließliches Denken in World- und Actor-Klassen)
  - Erfahrung hat problemlosen Wechsel bestätigt

Los geht‘s !!!

Grundbegriffe der OOP (ca. 4h)

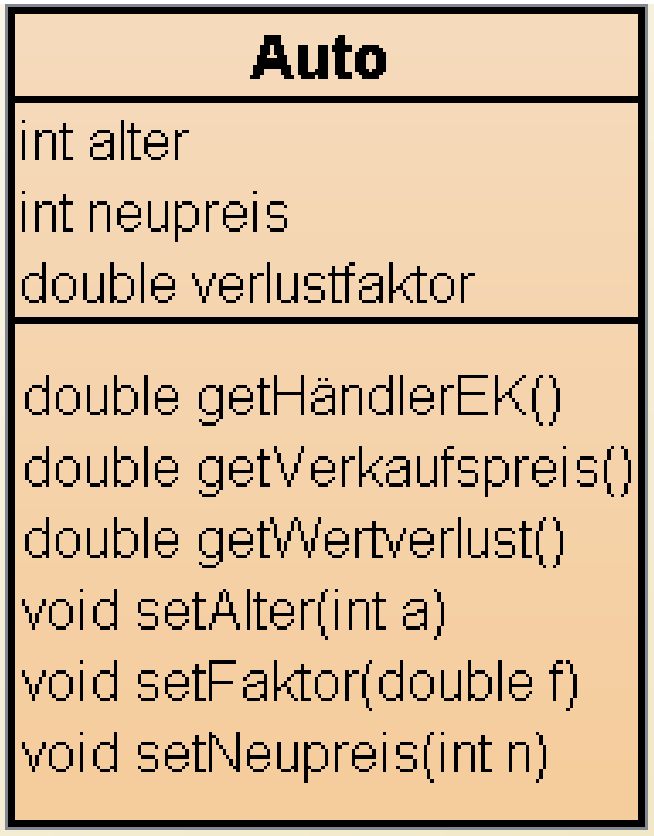
- inf-schule.de → OOP mit Java → Aufbau von Klassen → Hasen als Objekte
- „Vokabellernen“
- Interaktiv und anschaulich

Autobewertung (ca. 5-6h)

- Mathematische Modellierung des Preises als Exponentialfunktion, z.B.
  $\text{preis}(\text{alter}) = \text{neupreis} \cdot \text{verlustfaktor}^{\text{alter}}$
- Verlustfaktor z.B. 0.9
- Schreibweise in Java:
  Math.pow(verlustfaktor, alter)

Autobewertung

Beispiel für ein Klassendiagramm



- Mögliche Erweiterungen
  - Zahl der Vorbesitzer
  - Kilometerstand
  - …

Timer (ca.2h)

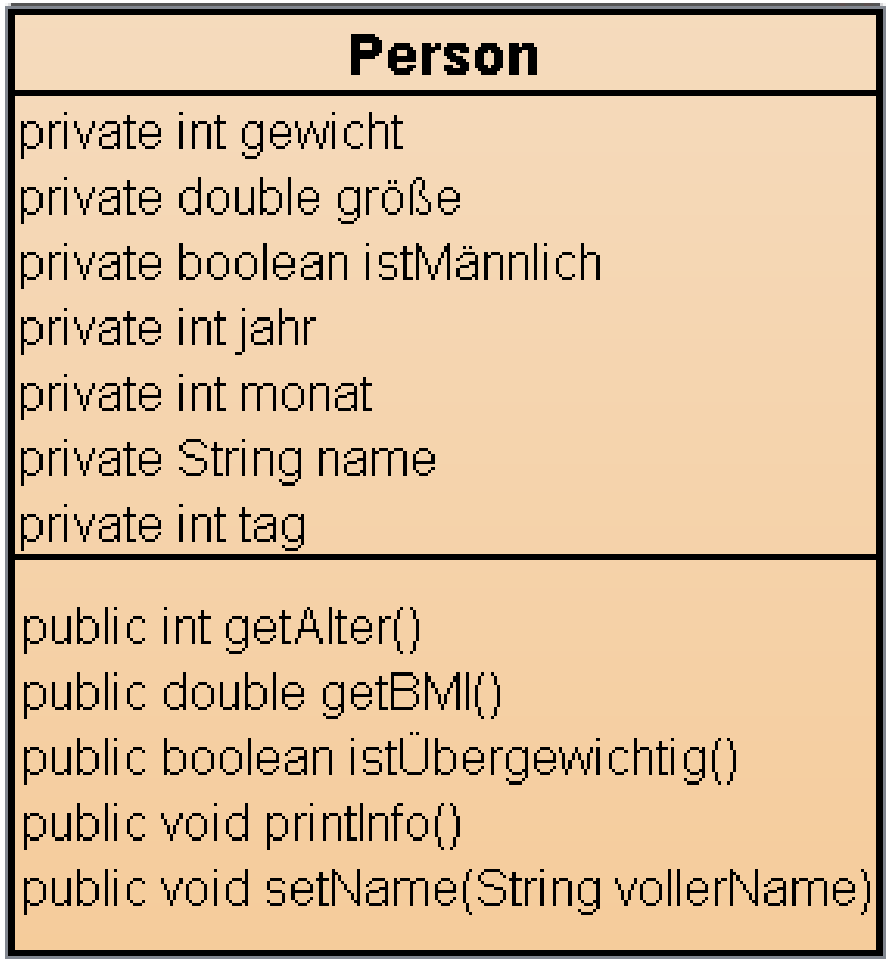
- Attribute werden immer als private definiert
  - In echten Projekten auf jeden Fall
  - In der Schule Ansichtssache
- Vorteile
  - Einfache Regeln
  - Geheimnisprinzip
- Nachteile
  - Get-/Set-Methoden notwendig
  - Tipparbeit

Schweine im Weltall (ca. 6h)

- Siehe inf-schule.de

Personal Info Center (ca. 5h)

Mögliches Klassendiagramm (nicht vollständig)



Personal info Center

Erweiterungsmöglichkeiten

- Berechnung, ob übergewichtig berücksichtigt Alter und Geschlecht
- Ausgabe ob Normalgewicht, leicht/stark übergewichtig
- Horoskop zufällig oder nach beliebiger Strategie
- Sternzeichen vereinfacht (nur Monate) oder richtig berechnet (komplexere Bedingungen)
- …

Frog (ca.4h)

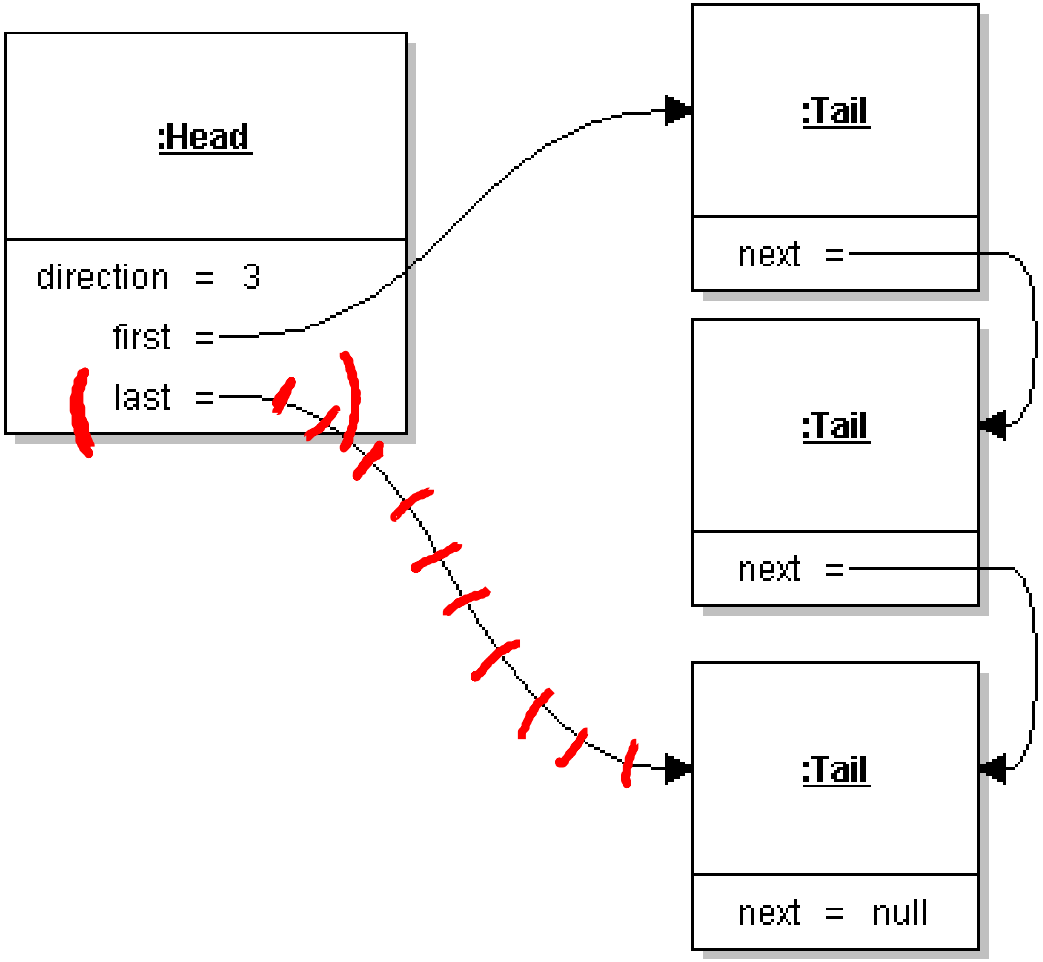
- Siehe Infoskript

Snake (ca.6-8h)

- Ergebnisse des Rollenspiels
  - Objekte haben eigentlich keine Namen, sondern nur die Referenzen darauf
  - Eine Referenz können wir uns vorstellen als Arm, der auf ein anderes Objekt zeigt
  - Bewegung der Schlange
    - Mein Nachfolger soll dort hin gehen, wo ich gerade bin.
    - Danach bewege ich mich dort hin wo ich hin wollte.
  - Kopf kennt mindestens erstes Element
  - Zum Einfügen praktisch: Kopf merkt sich letztes Element, oder Einfügen wird auch rekursiv erledigt.
  - Körperelement kennt nächstes Körperelement

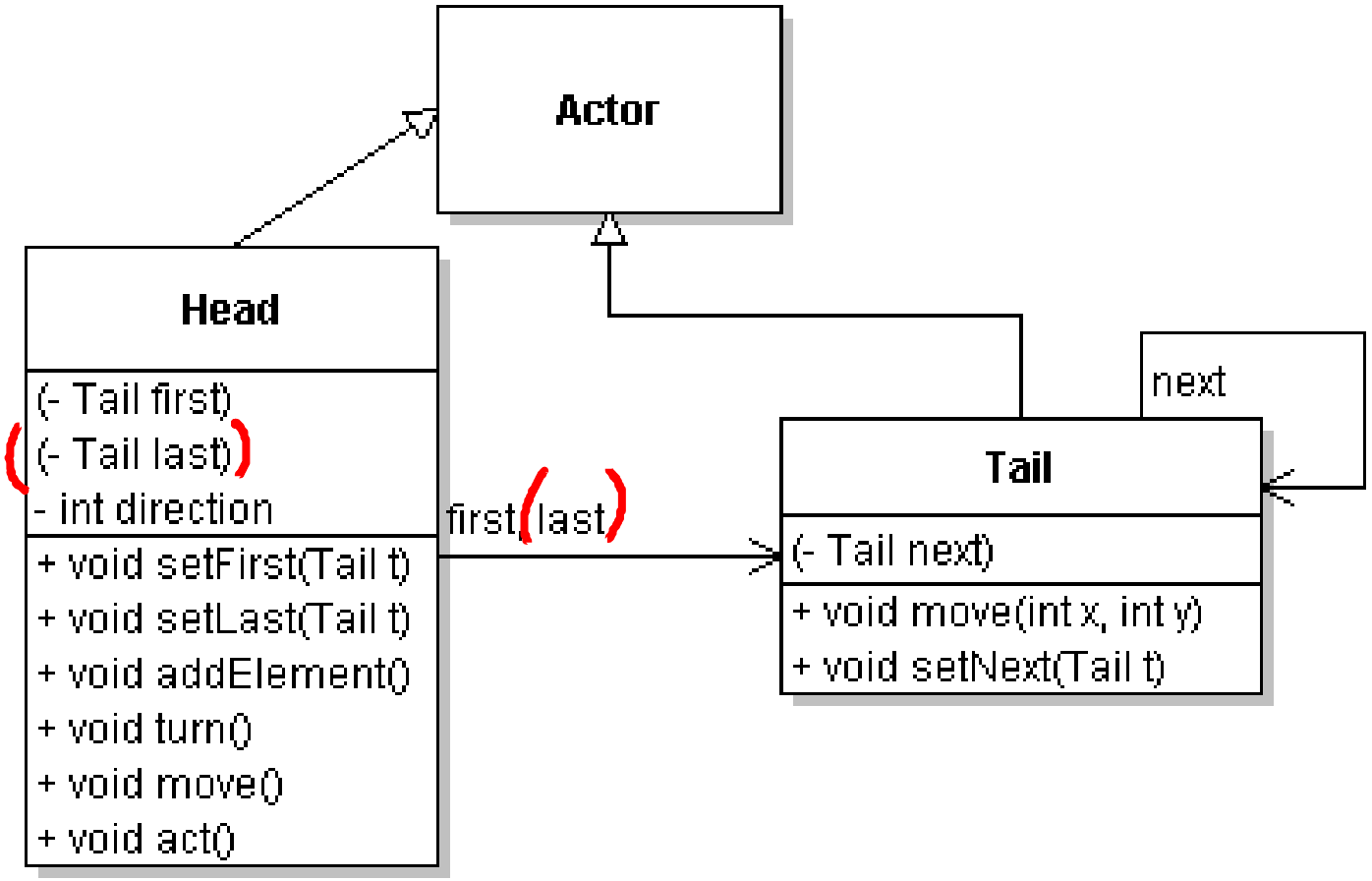
Snake

Mögliches Objektdiagramm



Snake

Mögliches Klassendiagramm



Snake

Mögliche Implementierungsschritte

- Steuern des Kopfes per Tastatur
  - 180°-Wende ist verboten
  - move/turn-Methoden
- Tail-Klasse
- Interaktives Erzeugen einer Tail-Sequenz und bewegen des ersten Tail-Elements durch Aufruf von move()
- setFirst(…) / setLast(…)-Methoden in Head
  - Interaktives Erzeugen einer Schlange und Steuern per Tastatur
- Beim Fressen Schlange automatisch verlängern