Praktikum Datamining und Sequenzanalyse

Einführung

Kai Dührkop – kai.duehrkop@uni-jena.de
Markus Fleischhauer – markus.fleischauer@uni-jena.de

Organisatorisches

Vorträge

Zu bestimmten Terminen im SR 3423

freies Arbeiten

Montag und Freitag im Linuxpool

https://bio.informatik.uni-jena.de/lehre/winter-1819/data-mining-und-sequenzanalyse/

Projekte

1. Exakte Suche

naive Suche, KMP, Boyer-Moore

2. Alignments

globales/lokales Alignment, Alignmentscore mit linearem Speicher

3. Clustering, phylogenetische Bäume

UPGMA, WPGMA , Neighbor-Joining

Projekte bauen aufeinander auf!

Gruppen

- 2-4 Personen pro Gruppe
- Mindestens 3 Gruppen

- jede Gruppe bekommt einen gemeinsamen Projektbereich im gitlab.
- Jedes Projekt bekommt ein eigenes git-Repository

`https://bio.informatik.uni-jena.de/gitlab/`

Gruppen

Bis Donnerstag, 18.10.2018:

- Gitlab Account erstellen
  - echte Namen angeben
- Gruppen bilden
- Gruppen zusammen mit gitlab usernamen per email an uns

`https://bio.informatik.uni-jena.de/gitlab/`

Ziele

- Teamarbeit
- Implementierung bekannter Algorithmen
- Auswirkung der Implementierung auf die Performance
- Arbeiten mit:
  - Versionskontrolle
  - Build-Tool
  - IDE (integrated development environment)

Aufgaben

Ein Aufgabenblatt pro Projekt (online)

1. Programmieren
   - Wartbarer Code – Objektorientiert (OOP)
   - Effizienter Code
   - Benutzerinterface (**CLI** oder Swing-GUI)
   - Dokumentation
2. Evaluation (Protokoll)
3. Präsentation (Vorträge)
   - Siehe Leitfaden Projektpräsentation (online)

https://bio.informatik.uni-jena.de/lehre/winter-1819/data-mining-und-sequenzanalyse/

Tools

- Objektorientierte Programmierung mit **Java**
- Dokumentation mit **Javadoc**
- Arbeiten in einer **Linux** Umgebung
- Versionskontrolle mit **git**
- Projektmanagement mit **Gradle**
- Benutzerinterface per **CLI** (oder GUI)

Auswertung und Vortrag

Auswertung

- Aufgabenzettel bearbeiten (**online**)
- Zeitmessungen
- auf mögliche Fehler eingehen
- Protokoll **vor** Vortrag abgeben!

Vortrag (Leitfaden online)

- Jede Gruppe trägt einmal vor (Dauer 30 - 45 min)
  - Vorstellung und Diskussion der Ergebnisse
  - Vorführen der Benutzerschnittstelle
  - Details zur Implementierung

Linux

Linux bringt Vielfalt

Distributionen

Mint Debian Ubuntu openSUSE

Fedora Mageia Manjaro CentOS

LXLE Arch elementary OS

...

Desktopumgebungen

KDE Gnome LXDE Xfce

MATE Unity Cinnamon Panteon

Verzeichnisstruktur

UNIX, Linux, ZETA

- / [Wurzelverzeichnis]
  - bin/
  - lib/
  - etc/
  - dev/
    - sda1 [Gerätedatei]
    - sda2 [Gerätedatei]
  - home/
    - benutzername/
    - anwendername/
      - Office Dokument.odt
      - Foto.png
      - Verzeichnis/
  - media/
    - disk1/ [Einhängpunkt für Datenträger]
    - usbdisk1/ [Einhängpunkt für Datenträger]
      - tabelle.ods
      - Bild.jpg
      - Text.txt
      - Unterordner/
  - usr/

Mac OSX

- / [Wurzelverzeichnis]
  - bin/ [versteckt]
  - System/
  - etc/ [versteckt]
  - dev/ [versteckt]
    - disk [Gerätedatei]
    - disk2 [Gerätedatei]
  - Users/
    - benutzername/
    - anwendername/
      - Office Dokument.odt
      - Foto.png
      - Verzeichnis/
  - Volumes/
    - macosx/ [Einhängpunkt für Datenträger]
    - usbmedium/ [Einhängpunkt für Datenträger]
      - tabelle.ods
      - Bild.jpg
      - Text.txt
      - Unterordner/
  - Applications/
  - usr/ [versteckt]

Windows NT/2000/XP

- C:\ [Laufwerk]
  - Programme\
  - Dokumente und Einstellungen\
    - Adminstrator\
    - Benutzername\
      - Office Dokument.odt
      - Foto.png
      - Verzeichnis\
  - Windows\
    - System32
- E:\ [Laufwerk]
- F:\ [Laufwerk, USB-Medium]
  - tabelle.ods
  - Bild.jpg
  - Text.txt
  - Unterordner\

Besonderheiten der Shell

Hilfe zu Befehlen

`info, man <Befehl>`	Hilfe zu einem Befehl
`<Befehl> --help`	kurze Hilfe

Completion

Name-Completion mit Tab-Taste

Befehle

`ls`	Anzeige des Verzeichnisinhalts
`cd`	Wechseln des Verzeichnisses
`mkdir`	Erzeugen eines Verzeichnisses
`rmdir`	Löschen eines leeren Verzeichnisses
`rm`	Löschen einer Datei
`rm -r`	Löscht rekursiv einen nicht leeren Ordners
`mv`	Verschieben
`cp`	Kopieren
`cat / less`	Anzeigen von Dateien
`grep`	Suchen in einer Datei
`head`	Anfang einer Datei anzeigen
`df -h`	Anzeigen der Festplattenbelegung

Benutzerrechte

- Jede Datei und jedes Verzeichnis ist einem Eigentümer und einer Gruppe zugeordnet.
- verschiedene Rechte für Eigentümer, Gruppe und andere
- Anzeigen der Rechte mit z.B. `ls -la`

`chmod`	Setzen der Dateirechte
`chown`	Ändern des Eigentümers
`chgrp`	Ändern der Gruppe
umask	Setzen der Standardrechte für neue Dateien

Java und OOP

Grundlagen

Klassen definieren Eigenschaften v. Objekten (Blaupausen)

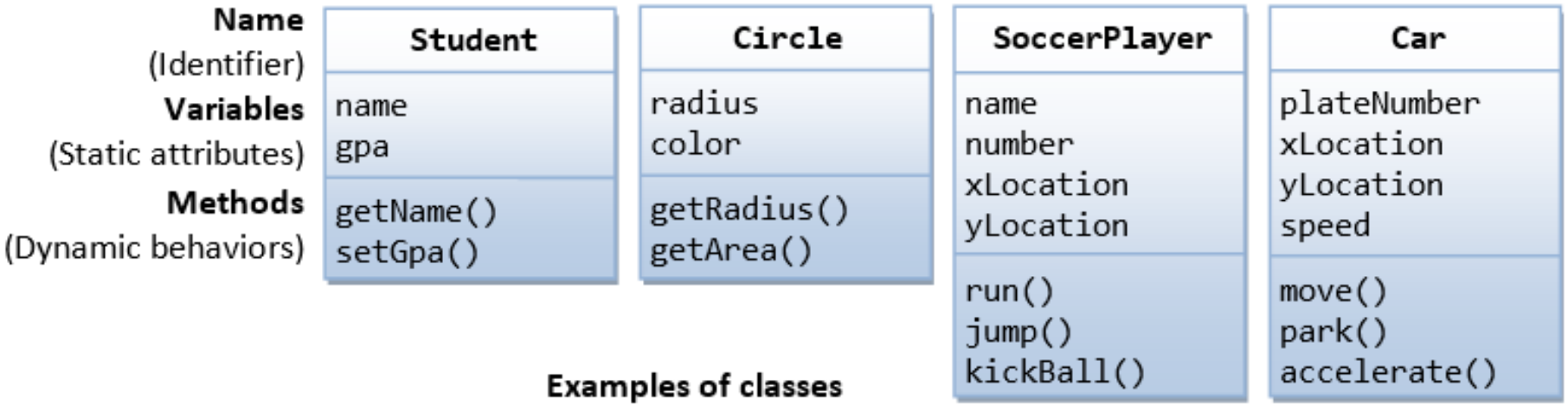
1. Name/Identifier
2. Attribute (Felder, Variablen)
3. Behaviour (Methoden)

Objekte sind Instanzen von Klassen

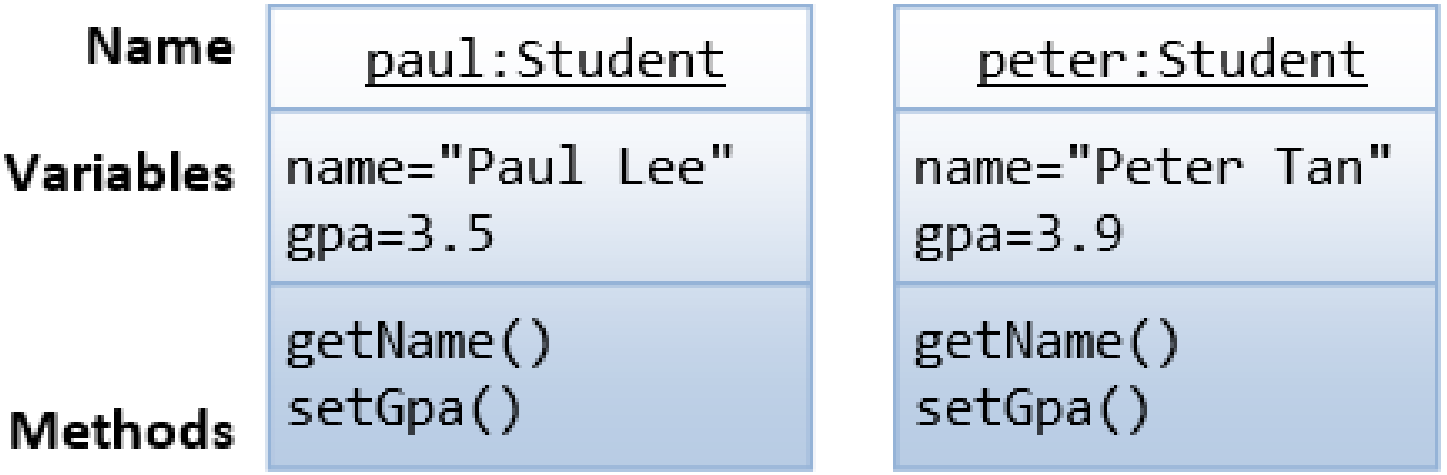
1. Felder – Belegung
2. Methoden – Abhängig v. Belegung

Repräsentation von physikalischen oder logischen Einheiten der Echtwelt

Klassen


Name
(Identifier)
Variables
(Static attributes)
Methods
(Dynamic behaviors)
Student
name
gpa
getName()
setGpa()
Circle
radius
color
getRadius()
getArea()
SoccerPlayer
name
number
xLocation
yLocation
run()
jump()
kickBall()
Car
plateNumber
xLocation
yLocation
speed
move()
park()
accelerate()
Examples of classes

Objekte - Instanzen



Two instances - paul and peter - of the class Student

Sichtbarkeit

Modifier	Class	Package	Subclass	World
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗

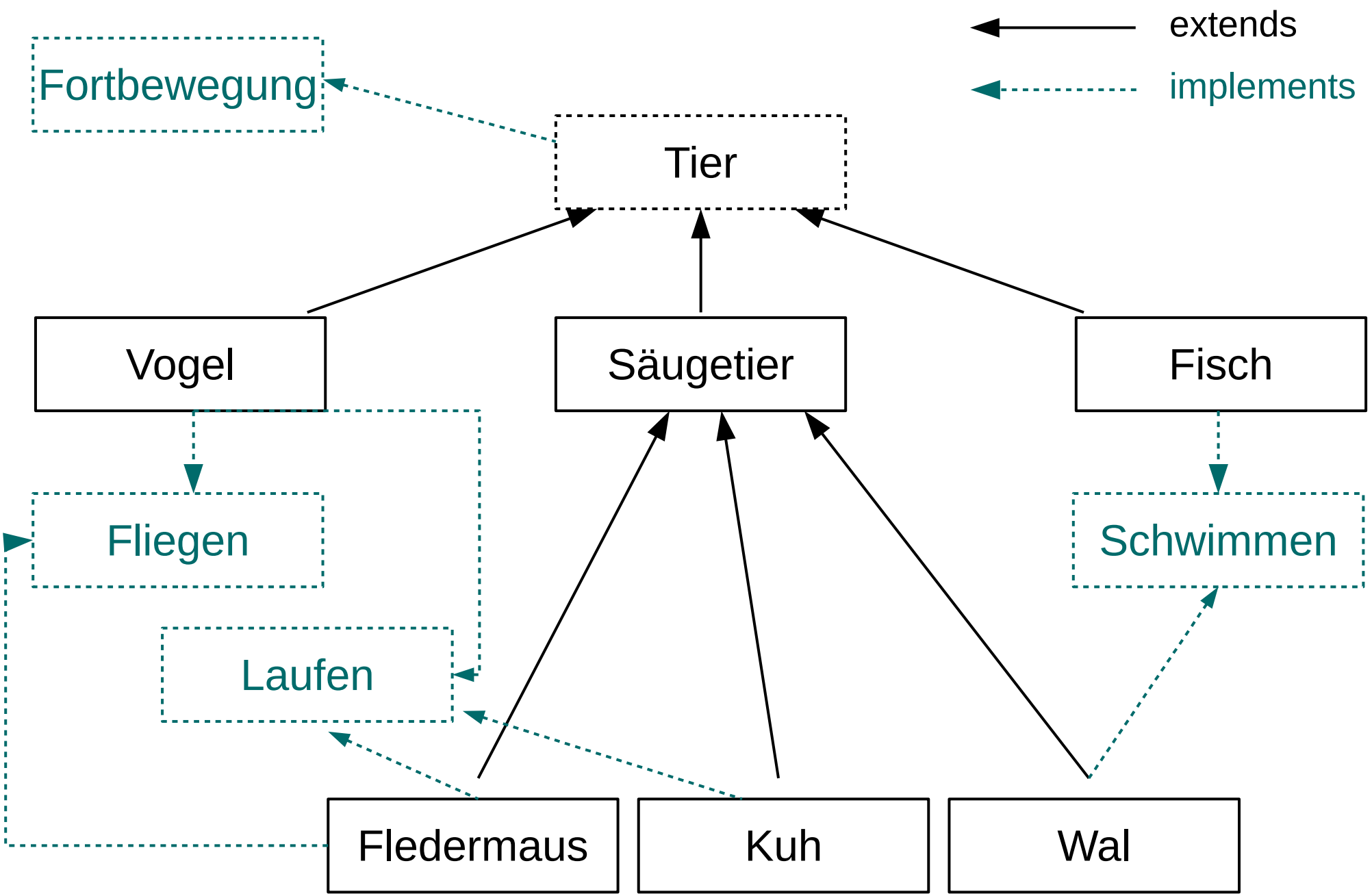
Klassentypen in Java

Klasse	• Instanzierbar • Nur aus-implementiert Methoden • Beliebige Felder und Sichtbarkeiten • Einfachvererbung
Abstrakte Klasse	• NICHT Instanzierbar • Abstrakte und implementierte Methoden • Beliebige Felder und Sichtbarkeiten • Einfachvererbung
Interface	• NICHT Instanzierbar • abstrakte Methoden mit default implementierung • keine Felder, nur public Methoden • Mehrfachvererbung

Wann welche Klassentypen

Klasse	• Beliebige Objekte
Abstrakte Klasse	• Vermeidung von doppeltem Code • Basis Implementierung bei der essentielle Funktionalität fehlt
Interface	• Definition von Schnittstellen • Gewährleistung von Funktionalitäten ohne die konkrete Implementierung vorzugeben

Vererbung


extends
implements
Fortbewegung
Tier
Vogel
Säugetier
Fisch
Fliegen
Schwimmen
Laufen
Fledermaus
Kuh
Wal

Interfaces?

```
public interface Distance{
    public double distance(Point p1, Point p2);
}

public class ManhattanMetrik() implements Distance{
    public double distance(Point p1, Point p2) {…};
}

public class EuclideanMetrik() implements Distance{
    public double distance(Point p1, Point p2) {…};
}

public class NearestNeighbor{
    public NNResult cluster(List<Point> points, Distance distance){
        …
    }
}
```

Mehrfachvererbung?

```
public abstract class Tier{
    public abstract void move();
}

public interface CanFly{
    public void fly();
}

public class Bird extends Tier implements CanFly{
    public void move(){
        //do something
    }
    public void fly(){
        //do something
    }
}
```

Polymorphismus

```
public class Clock{
   public static getFormat(){…}
   public void setTime(long ns){…}
   public void setTime(int h, int m, int s, int ms){...}
}

public class MoreSpecificClock extends Clock{
   public static getFormat(){…}
  @Override
   public void setTime(long ns){…}
}
```

Ueberladen

Ueberdecken

Ueberschreiben

Zugriff auf Attribute durch Getter/Setter

```
public class WithoutEncapsulation{

    public String value;

    public void printLength(){
        System.out.println("Länge: " + value.length());
    }

    public static void main(String[] args){
        WithoutEncapsulation we = new WithoutEncapsulation();
        we.value = null;
        we.printLength();
    }

}
```

NullPointerException

Zugriff auf Attribute durch Getter/Setter

```
public class WithEncapsulation{

    private String value;

    public void setValue(String value){
        if(value == null) this.value = “”;
        else this.value = value;

    public void printLength(){
        System.out.println(“Länge: “ + value.length());
    }

    public static void main(String[] args){
        WithoutEncapsulation we = new WithoutEncapsulation();
        we.setValue(null);
        we.printLength();          →    Länge: 0
    }

}
```

Hinweise

Zeit messen

via command line time

```
#$ time sleep 10
real 0m10.116s
user 0m0.001s
sys 0m0.007s
```

via Java

```
System.currentTimeMillis()
System.nanoTime()
long time = System.nanoTime();
//do something
long duration = System.nanoTime() - time;
```

Zeit messen

Vorgehen

- mehrfach messen
- Minimum aller Messungen verwenden

Testumgebung beschreiben

- Betriebssystem
- Hauptspeicher
- CPU
- Java VM version
- Java Heapspace

IO

```
try(BufferedReader reader =
          new BufferedReader(new FileReader(path))){
   String temp = null;
   while((temp = reader.readLine()) != null){
      //werte aus
   }
}catch(IOException e){
   // ...
}
```

Fragen?