

Praktikum Datamining und Sequenzanalyse

Einführung

Kai Dührkop

Markus Fleischhauer

- kai.duehrkop@uni-jena.de

- markus.fleischhauer@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-1718/data-mining-und-sequenzanalyse/>

Themen

Exakte Suche

naive Suche, KMP, Boyer-Moore

Alignments

globales/lokales Alignment, Alignmentsscore mit linearem Speicher

Clustering, phylogenetische Bäume

UPGMA, WPGMA, Neighbor-Joining

Gruppen

- 2-4 Personen pro Gruppe
- Mindestens 3 Gruppen → **Gruppen bilden!**
- jede Gruppe bekommt ein gemeinsames git-Repository:
- `https://bio.informatik.uni-jena.de/git/`

Ziele

- Teamarbeit
- Implementierung bekannter von Algorithmen
- Auswirkung der Implementierung auf die Performance
- Arbeiten mit Versionskontrolle, Build Tool und IDE

Aufgaben(blatt)

- Programmieren
 - Wartbarer Code – Objektorientiert (OOP)
 - Effizienter Code
 - Benutzerinterface (**CLI** oder Swing-GUI)
 - Dokumentation
- Evaluation (Protokoll)
- Präsentation (Vorträge)

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
- Aufgabenzettel **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

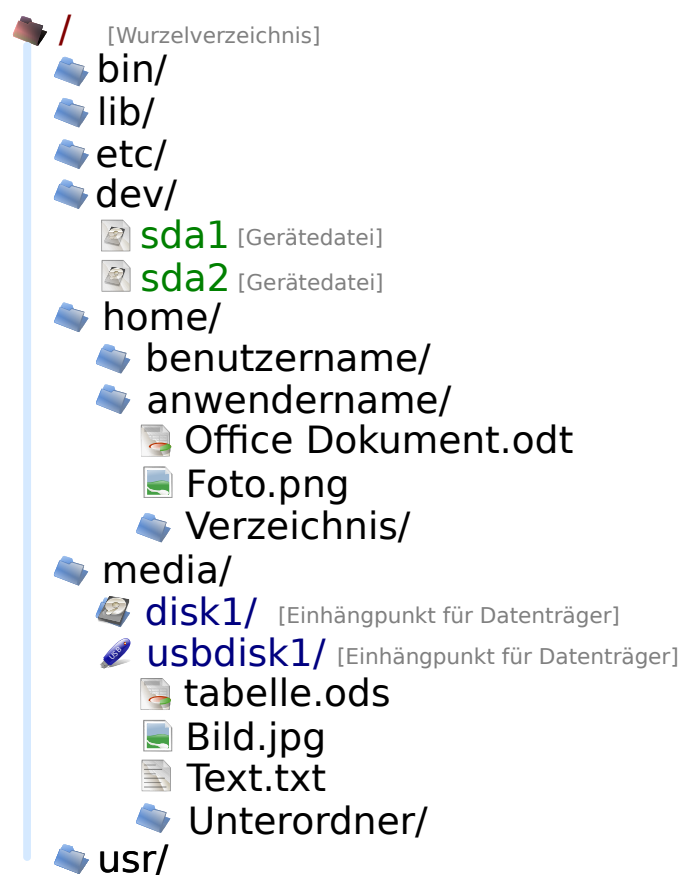
Mint	Debian	Ubuntu	openSUSE
Fedora	Mageia	Manjaro	CentOS
LXLE	Arch	elementary OS	
	...		

Desktopumgebungen

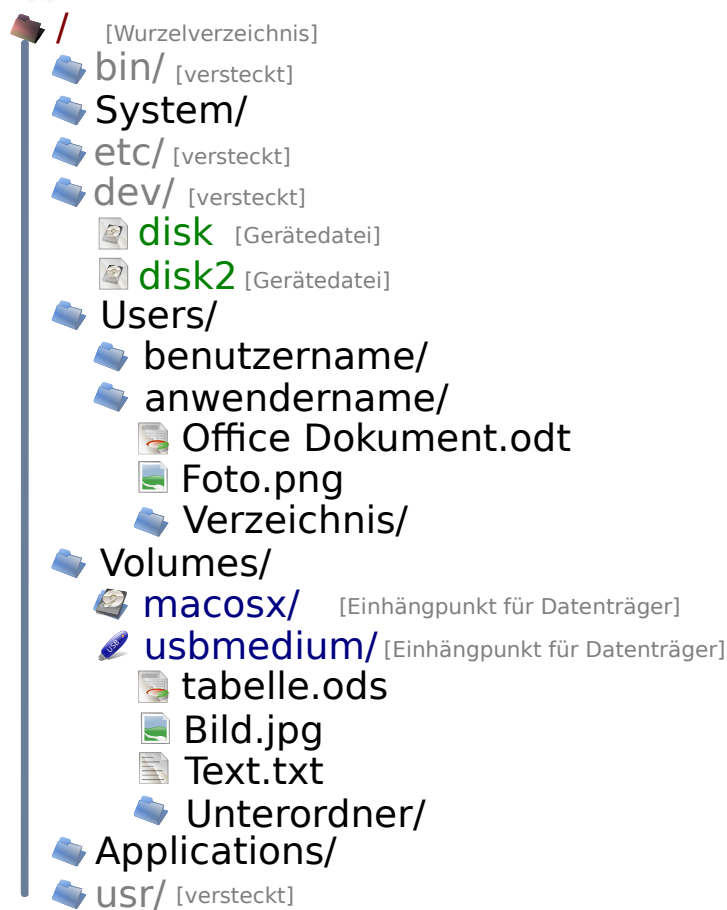
KDE	Gnome	LXDE	Xfce
MATE	Unity	Cinnamon	Panteon

Verzeichnisstruktur

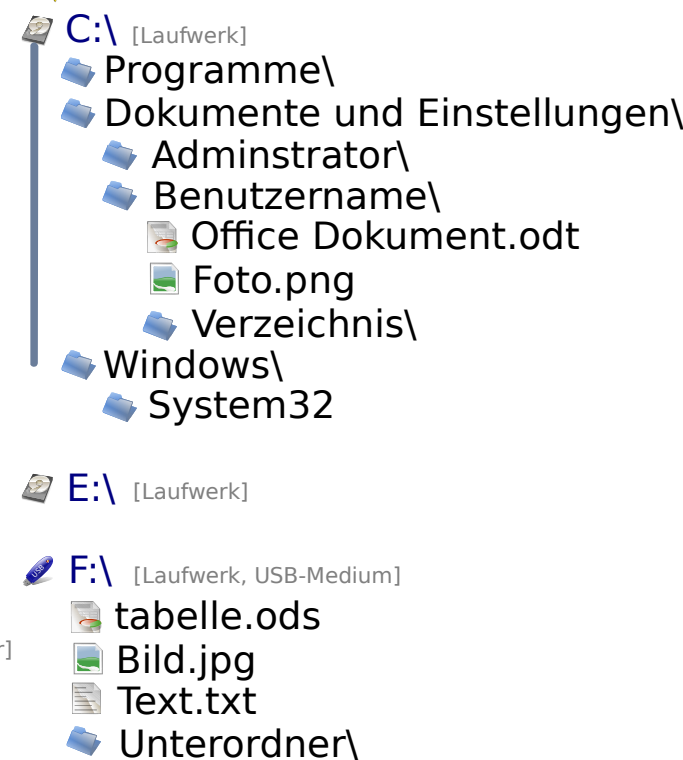
UNIX, Linux, ZETA



Mac OS X



Windows NT/2000/XP



Besonderheiten der Shell

Hilfe zu Befehlen

`info, man <Befehl>` Hilfe zu einem Befehl

`<Befehl> --help` kurze Hilfe

Completion

Name-Completion mit Tab-Taste

Befehle

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

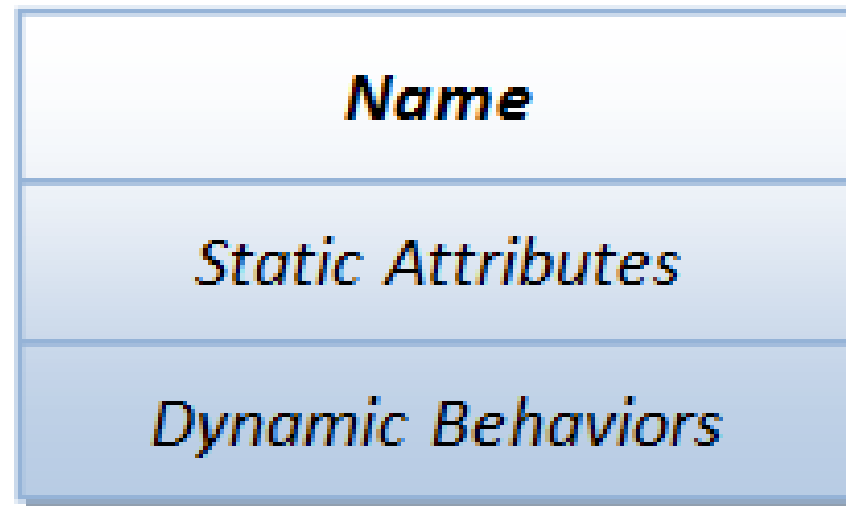
<code>chmod</code>	Setzen der Dateirechte
<code>chown</code>	Ändern des Eigentümers
<code>chgrp</code>	Ändern der Gruppe
<code>umask</code>	Setzen der Standardrechte für neue Dateien

Java und OOP

Bitte nicht so!

```
public class Class1{  
  
    public static void save(String path, double score){...}  
    public static double[][] load(String path){...}  
    public static double algo1(double[][] data){...}  
    public static double algo2(double[][] data){...}  
  
    public static void main(String[] args){  
        double[][] data = load(args[0]);  
        double score1 = algo1(data);  
        double score2 = algo2(data);  
        save(args[1],score1);  
        save(args[2],score2);  
    }  
}
```


Grundlagen



A class is a 3-compartment box

Klassen

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


Examples of classes

Instanzen


Name	<u>paul:Student</u>	<u>peter:Student</u>
Variables	name="Paul Lee" gpa=3.5	name="Peter Tan" gpa=3.9
Methods	getName() setGpa()	getName() setGpa()

Two instances - paul and peter - of the class Student

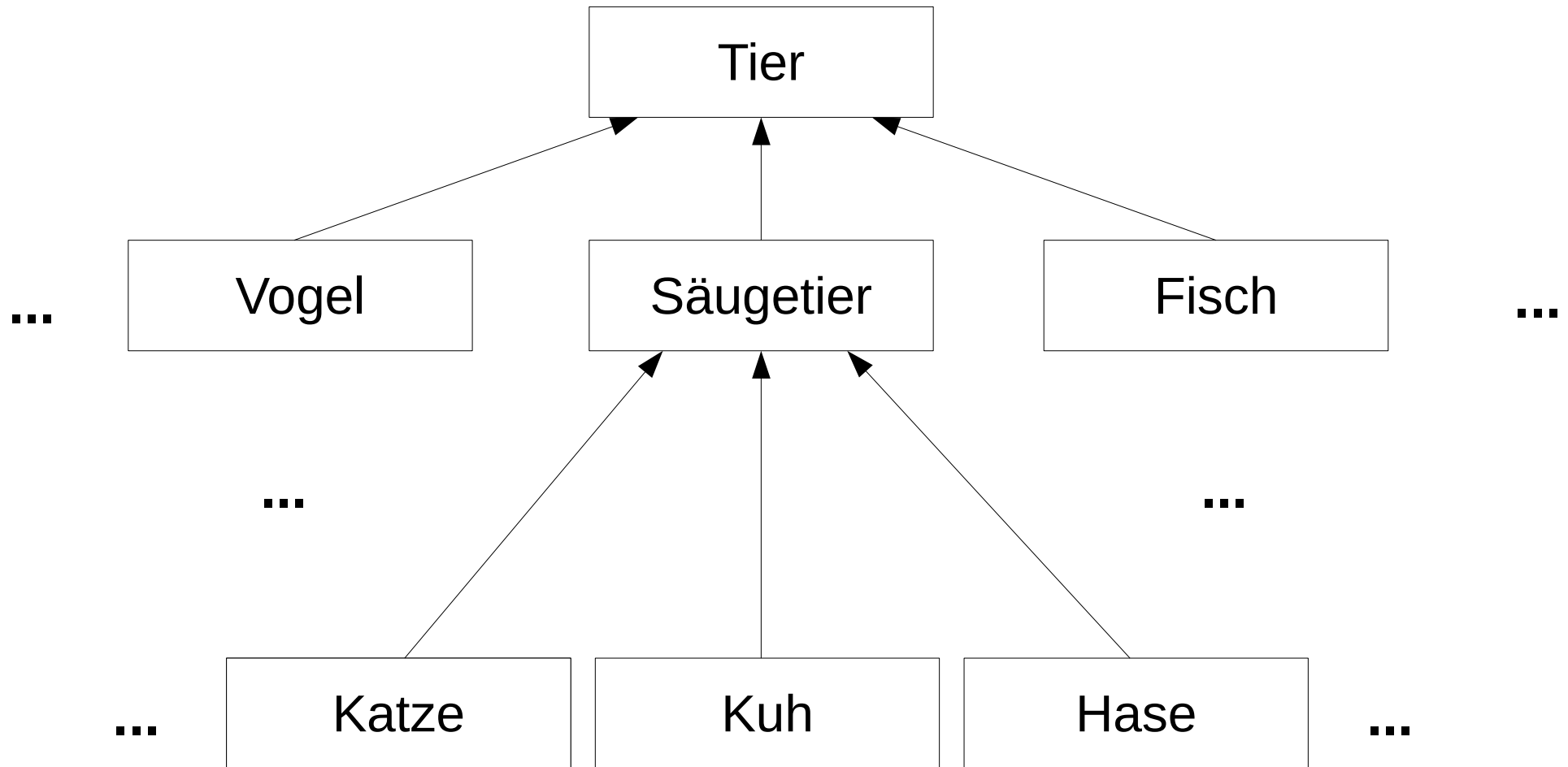
Zugriff auf Attribute durch Getter/Setter

```
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  
    }  
}
```

Vererbung



Interfaces?

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

public class ManhattanDistance() implements Distance{
    public double distance(Point p1, Point p2) {...};
}

public class EuclideanDistance() 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){...} Ueberladen  
    public void setTime(int h, int m, int s, int ms){...}  
}
```

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

Sichtbarkeit

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

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
- Systemspeicher
- CPU
- Java VM version
- Heapspace

IO

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

Fragen?