

# Praktikum Data Mining und Sequenzanalyse

Einführung

Marvin Meusel – [marvin.meusel@uni-jena.de](mailto:marvin.meusel@uni-jena.de)

Markus Fleischhauer – [markus.fleischhauer@uni-jena.de](mailto:markus.fleischhauer@uni-jena.de)

# Organisatorisches

---

## Vorträge

Zu bestimmten Terminen im SR 3423

## freies Arbeiten

Montag und Freitag im Linuxpool

`bio.informatik.uni-jena.de/lehre/`

# Themen

---

## Exakte Suche

naive Suche, KMP, Boyer-Moore

## Alignments

globales/lokales Alignment, Alignmentsscore mit linearem Speicher

## Clustering, phylogenetische Bäume

UPGMA, WPGMA, Neighbor-Joining ?

# Gruppen

---

- 2-3 Personen pro Gruppe
- jede Gruppe bekommt eine eigene Linux-Gruppe  
biodmXY
- jede Gruppe bekommt ein gemeinsames  
git-Repository /home/stud/bipr/biodmXY

# Ziele

---

- Teamarbeit
- Implementierung von Algorithmen
- Auswirkung der Implementierung auf Laufzeit
- Arbeiten unter Linux

# Aufgaben

---

- Programmieren
  - „schön programmieren“ - OOP
  - Benutzerinterface (CLI oder Swing-GUI)
  - Dokumentation
- Auswertung
- Vorträge

# Tools

---

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

# Auswertung und Vortrag

## Auswertung

- Aufgabenzettel bearbeiten
- Zeitmessungen
- auf mögliche Fehler eingehen
- Aufgabenzettel vor Vortrag abgeben (PDF)!

## Vortrag

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



**Linux**

# Linux bringt Vielfalt

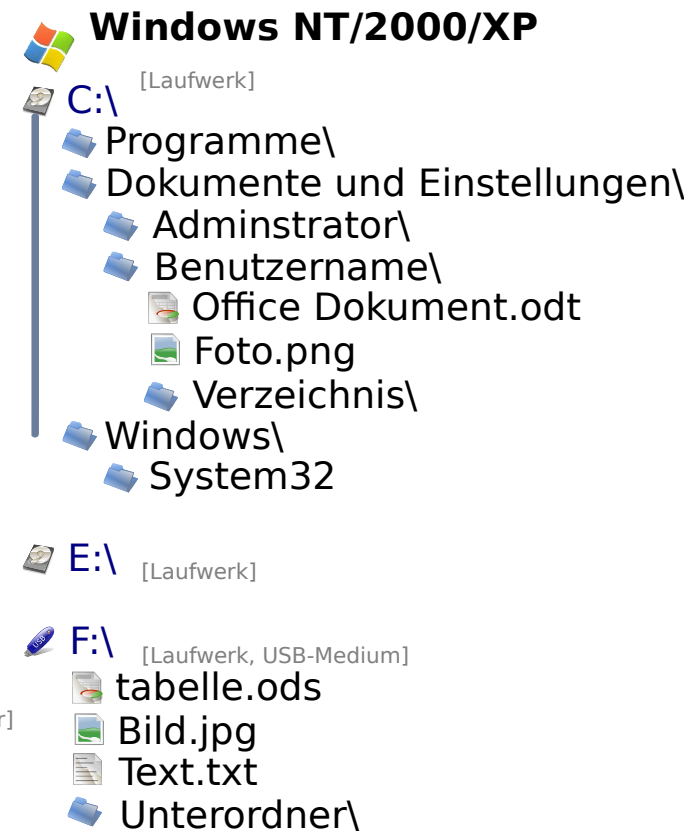
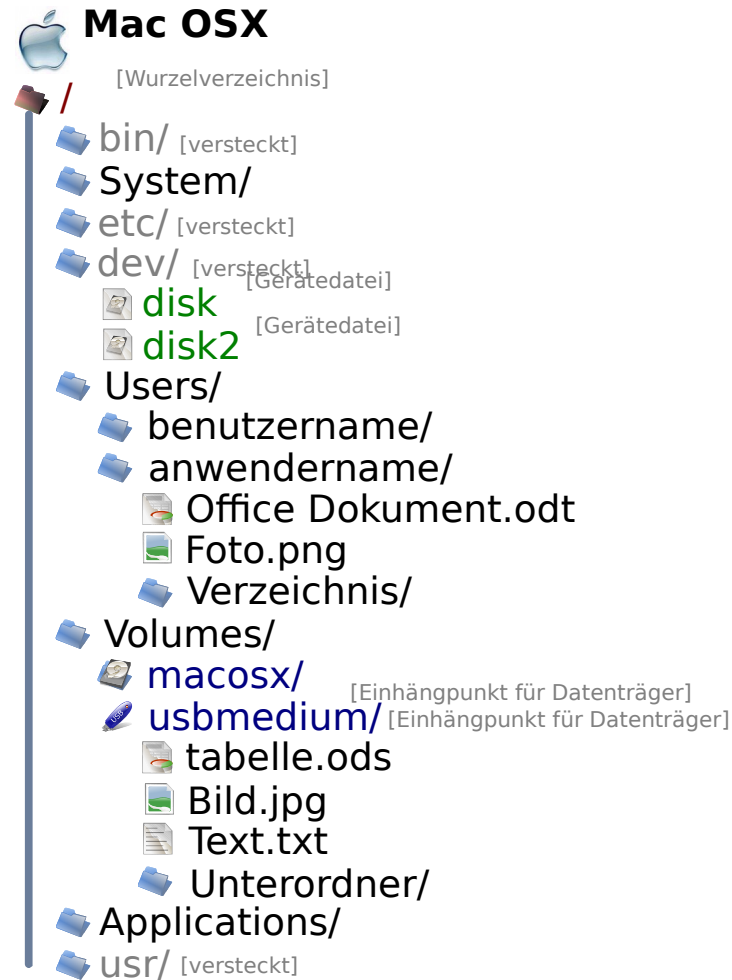
## derzeit beliebte Distributionen

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

## Desktopumgebungen

KDE	Gnome	LXDE	Xfce
MATE	Unity	Cinnamon	Panteon

# Verzeichnisstruktur



# 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);  
    }  
}
```



# OOP

---

## Prinzipien

Abstraktion

Datenkapselung

Polymorphie

Vererbung

## Die OOP Welt

Klasse

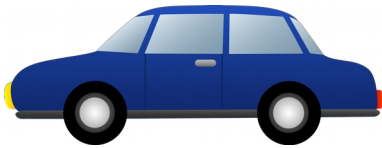
Objekt

Attribute

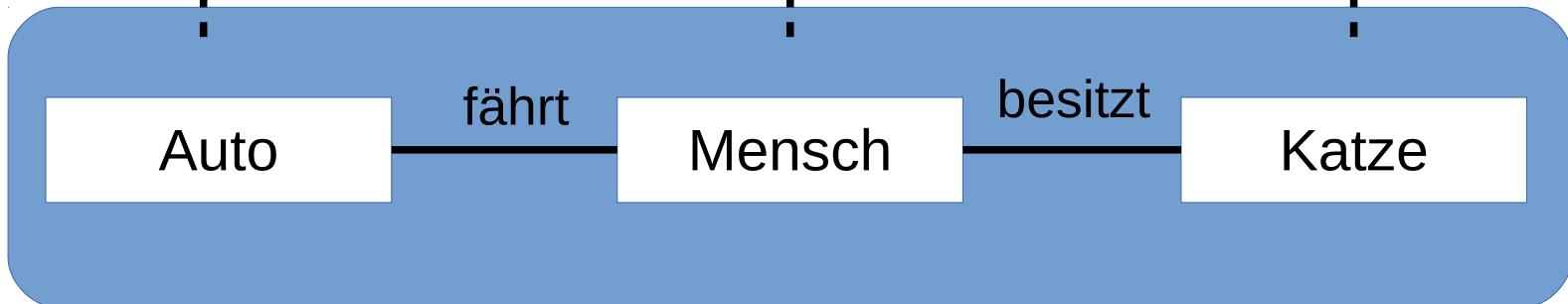
Methoden

# Grundlage

Realität



Modell



# Attribute und Methoden

## Tier

Größe

Länge

Gewicht

bewege(pos : Point)

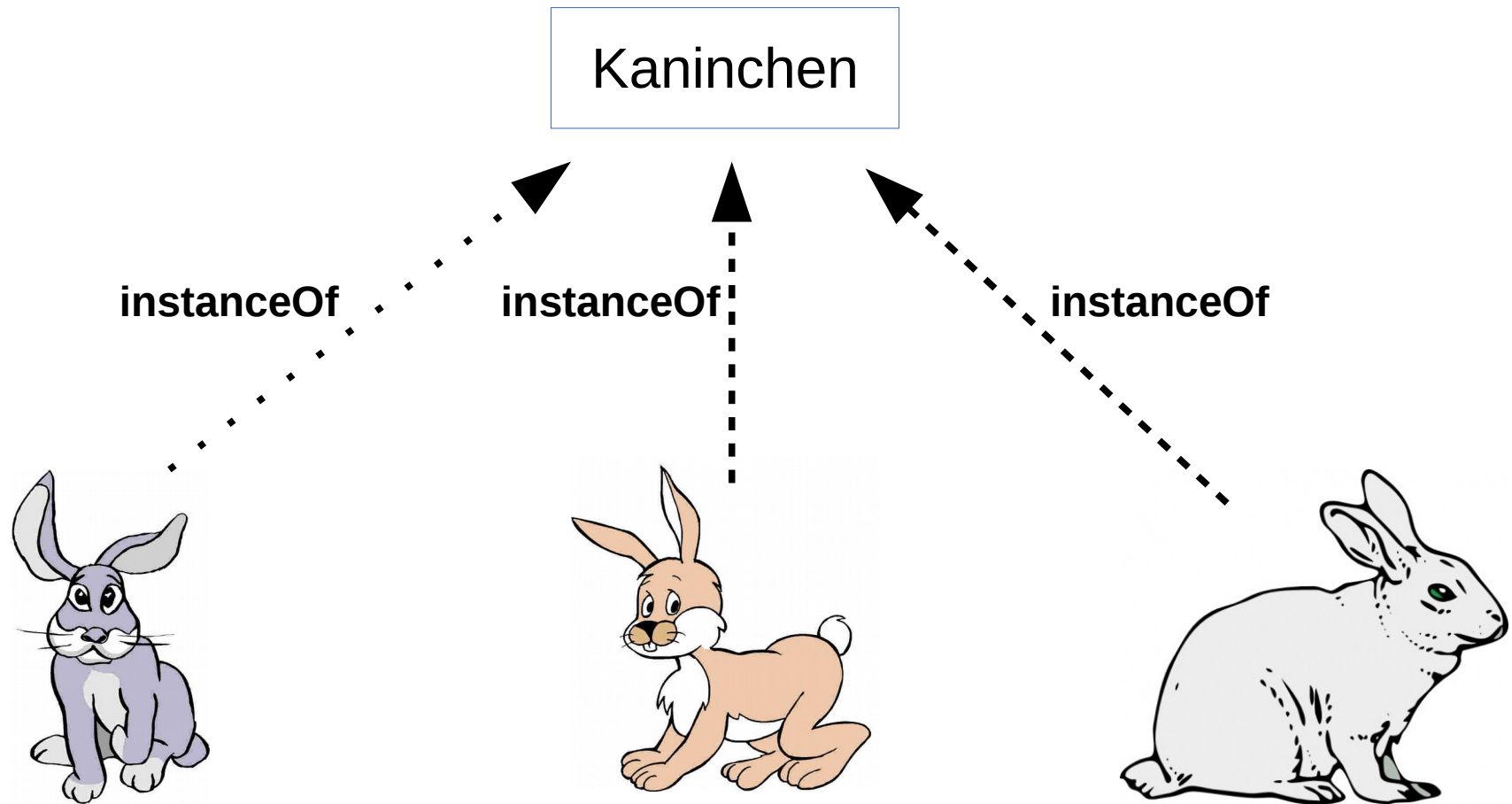
esse(other : Essen)

schlafe()


setSize(size : int)

getSize()


# Klassen und Objektinstanzen



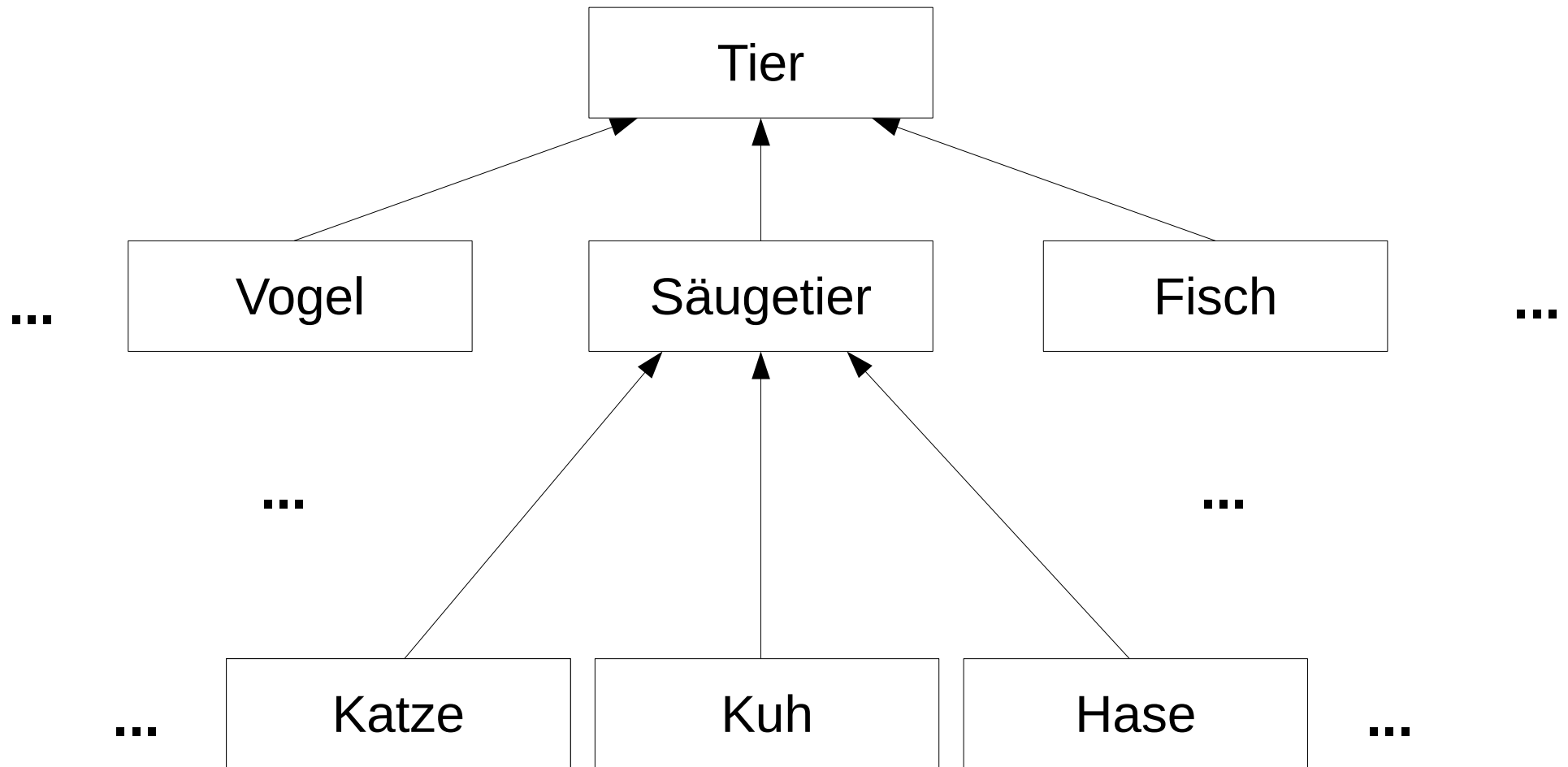
# Zugriff auf Attribute durch Getter/Setter

```
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 dynamisch/statisch

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

```
public class BetterClock extends Clock{  
  
    @Override  
    public void setTime(long ns){...}  
  
}
```

# 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){
    // ...
}
```

# Links

---

[distrowatch.com](http://distrowatch.com)

[galileo-press.de/openbook](http://galileo-press.de/openbook)

[ubuntuusers.de](http://ubuntuusers.de)

[wiki.ubuntuusers.de/Shell/Befehlsübersicht](http://wiki.ubuntuusers.de/Shell/Befehlsübersicht)