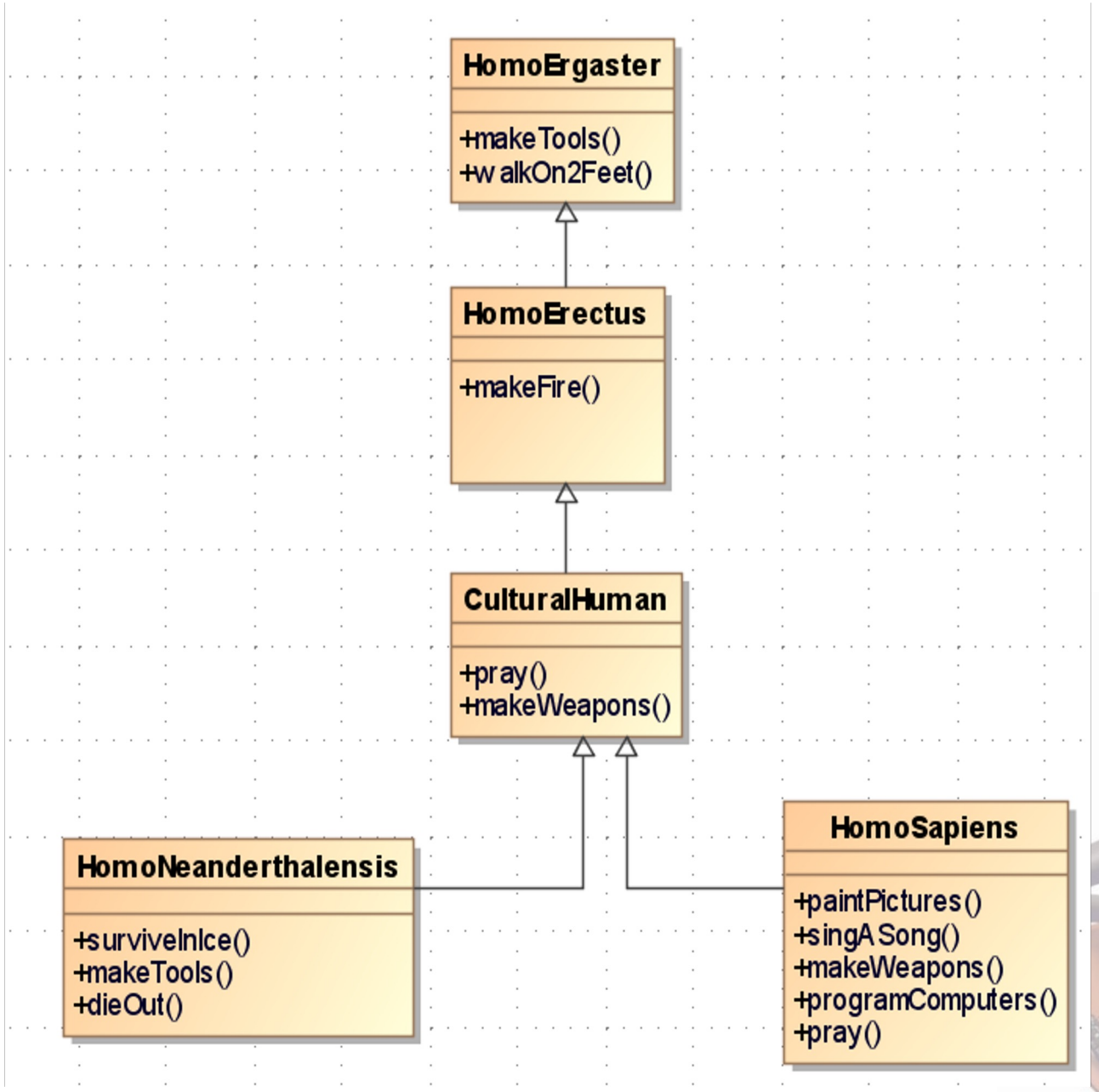
Java Fundamentals

Inheritance (Vererbung)

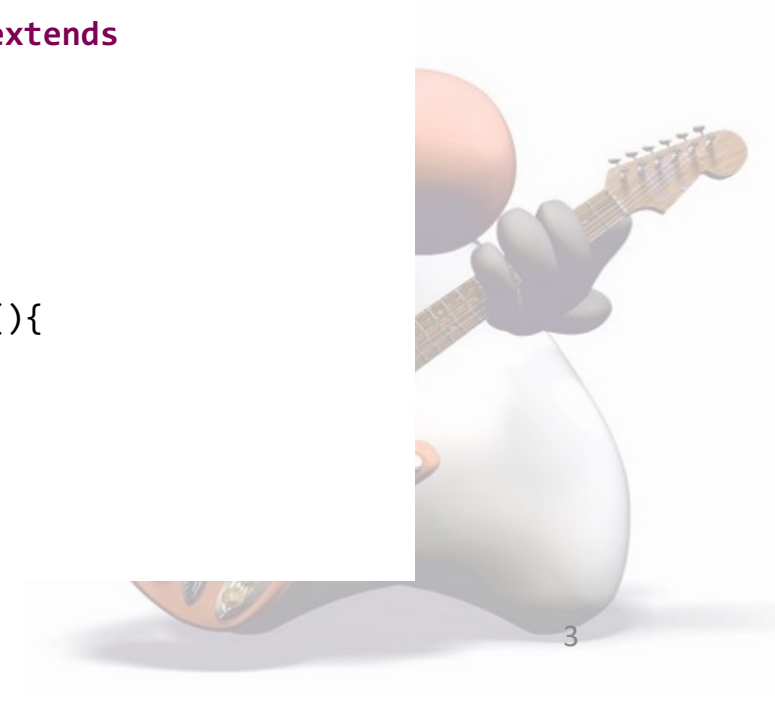

HomoErgaster
+makeTools()
+walkOn2Feet()
HomoErectus
+makeFire()
CulturalHuman
+pray()
+makeWeapons()
HomoNeanderthalensis
+surviveInIce()
+makeTools()
+dieOut()
HomoSapiens
+paintPictures()
+singASong()
+makeWeapons()
+programComputers()
+pray()

```java
public class HomoErgaster {
        public void makeTools(){
                // ...
        }

        public void walkOn2Feet(){
                //...
        }
}
```

```java
public class HomoErectus extends HomoErgaster{
        public void makeFire(){
                // ...
        }
}
```

```java
public class CulturalHuman extends HomoErectus {
        public void pray(){
                // ...
        }

        public void makeWeapons(){
                //..
        }
}
```

```java
public class HomoNeanderthalensis extends
CulturalHuman{
        @Override
        public void makeTools(){
                // ..
        }

        public void surviveInIce(){
                //..
        }

        public void dieOut(){
                // ..
        }
}
```



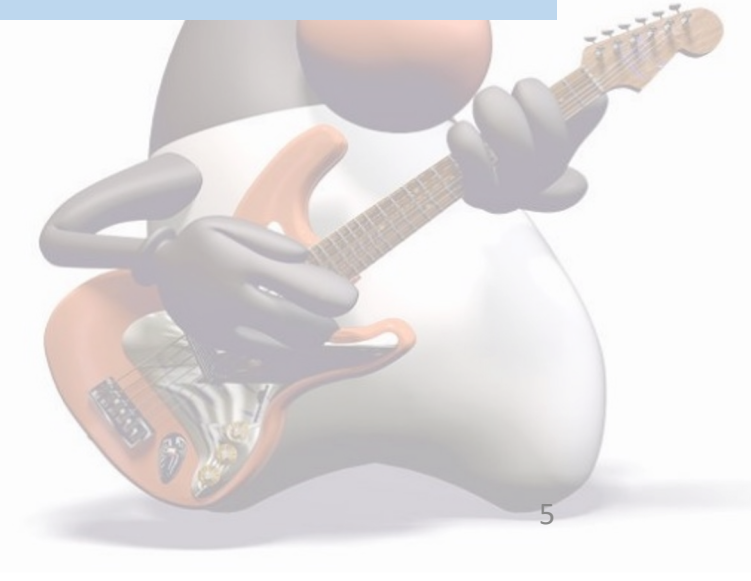
Homo Neanderthalensis kann alles, was seine Vorfahren konnten

```
public class Evolution {
	public static void main(String[] args) {
		HomoNeanderthalensis tom = new HomoNeanderthalensis();
		tom.walkOn2Feet();
		tom.makeFire();
		tom.makeTools();
		tom.makeWeapons();
		tom.pray();
		tom.surviveInIce();
	}
}
```

Vererbung

Eine Klasse kann mithilfe des Schlüsselwortes extends von einer anderen Klasse erben

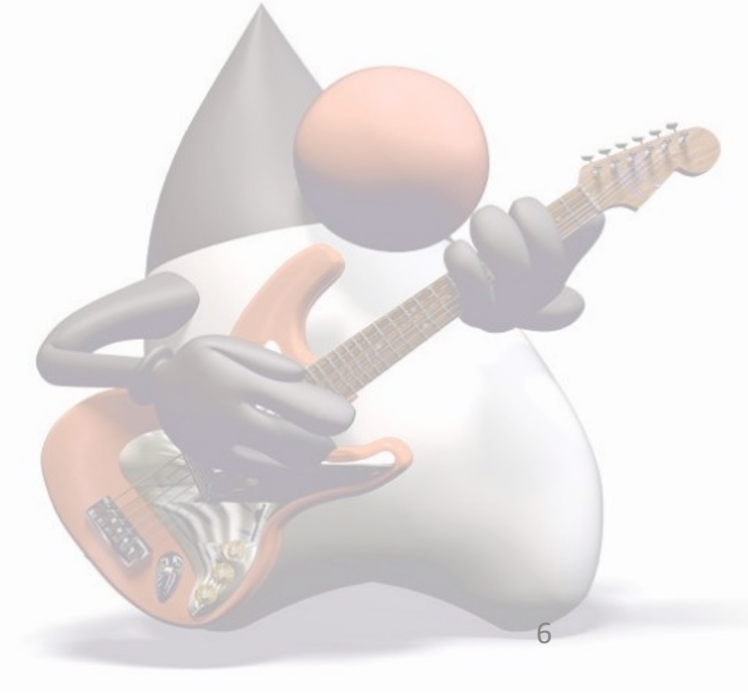
Klassen die von einer anderen Klasse erben, besitzen alle ihre Methoden und Attribute.



Wenn Klasse B von Klasse A erbt

- B erbt alle Attribute von A
- B erbt alle Methoden von A
- B erbt keine Konstruktoren

- Man spricht davon das A die Superklasse von B ist
- Man spricht davon das B die Sub-Klasse von A ist
- Man kann auch sagen das B ein A ist



Object ist Superklasse aller Klassen

```
public class HomoErgaster extends
Object{
        public void makeTools(){
                // ...
        }

        public void walkOn2Feet(){
                //...
        }
}
```

Ist das selbe wie…

```
public class HomoErgaster {
        public void makeTools(){
                // ...
        }

        public void walkOn2Feet(){
                //...
        }
}
```

Wichtige Object Methoden

public String toString();

public boolean equals();

Eine Object Variable kann alle Instanzen egal welchen Klassentyps halten

```
Object object = new HomoErgaster();
Object o2 = new Account();
```

Jede Klasse hat eine Implementierung von toString() – geerbt von Object

```
public void println(Object o){
        String printString = o.toString();
    // ...
}
```

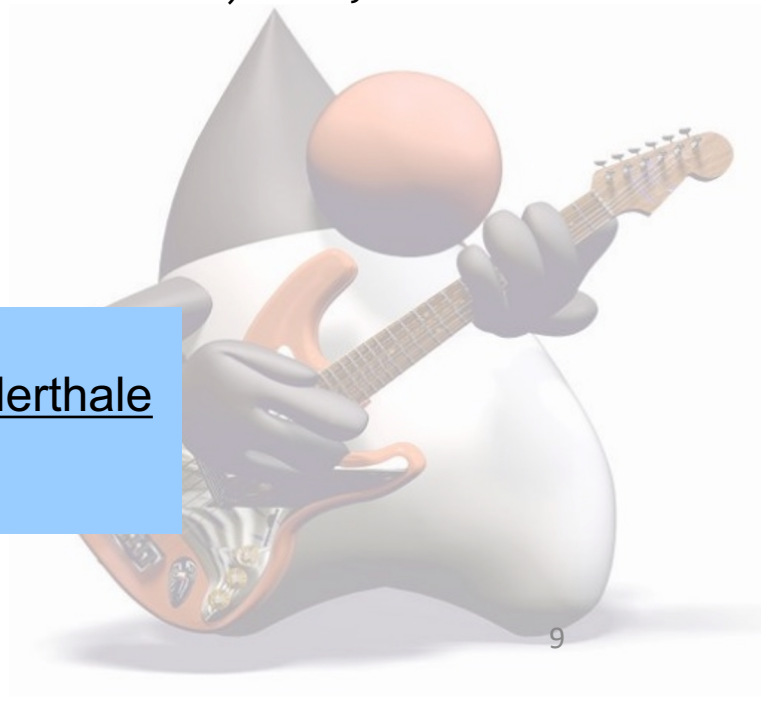
Polymorphie

Die JVM weiß immer, was der echte Typ einer Referenz ist und kann den Original Typ wiederherstellen.

```
public boolean dieOut(CulturalHuman human){
        if (human instanceof HomoNeanderthalensis){
                HomoNeanderthalensis neanderthalensis = (HomoNeanderthalensis)human;
                neanderthalensis.dieOut();
                return true;
        } else {
                return false;
        }
}
```

CulturalHuman human

:HomoNeanderthalensis



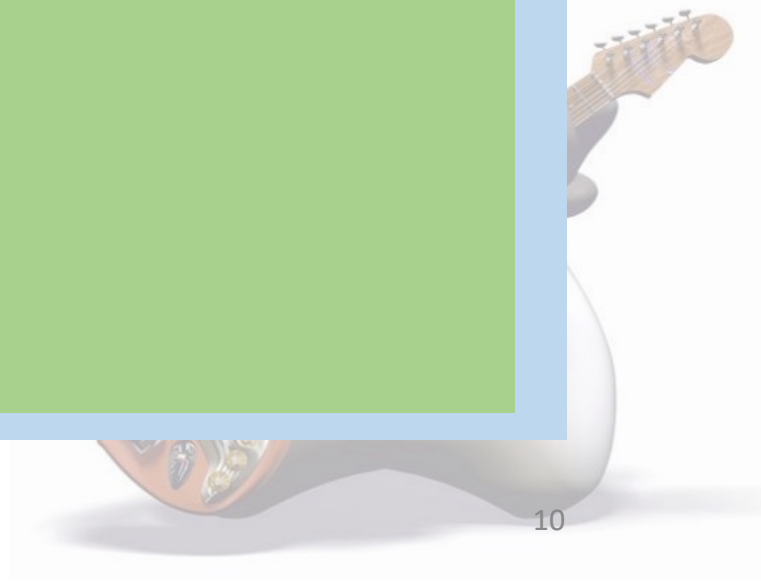
Vererbung

Eine Klasse kann mithilfe des Schlüsselwortes extends von einer anderen Klasse erben

Klasse die von einer anderen Klasse erben besitzen alle ihre Methoden und Attribute.

Können auch weiter spezialisiert werden

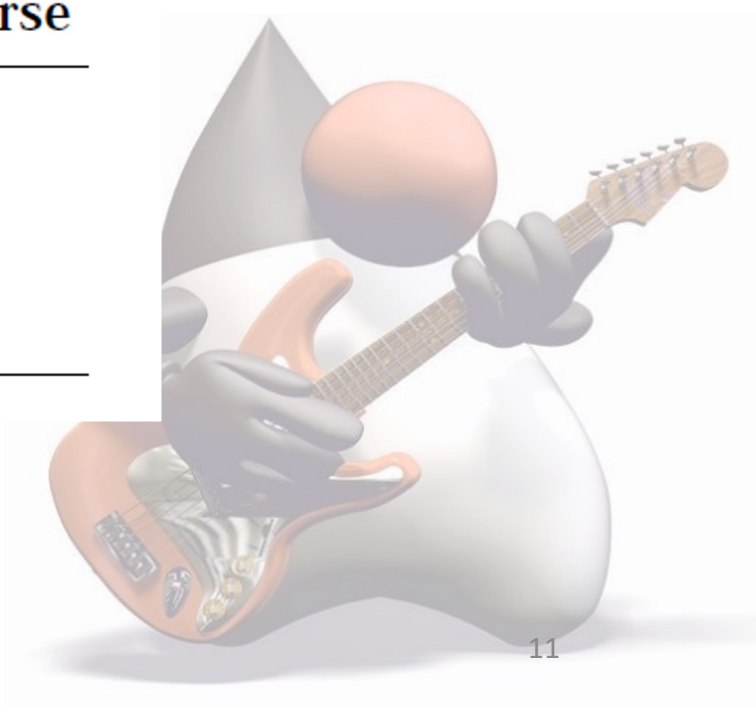
Eine Subklasse kann einer Superklassen Referenzvariable zugewiesen werden



Access Modifiers

- auf private Attribute kann eine Subklasse nicht zugreifen
- Subklassen können auf protected Variablen zugreifen

Modifier	Same Class	Same Package	Subclass	Universe
private	Yes			
default	Yes	Yes		
protected	Yes	Yes	Yes	
public	Yes	Yes	Yes	Yes



Konstruktoren

```
public CulturalHuman(){
}
```

Ist eigentlich …

```
public CulturalHuman(){
        super();
}
```

- super() ruft den Konstruktor der Superklasse auf (Object)
- der super Konstruktor muss IMMER aufgerufen werden, daher muss man den Aufruf super() nicht explizit angeben
- Konstruktoren werden nicht vererbt.

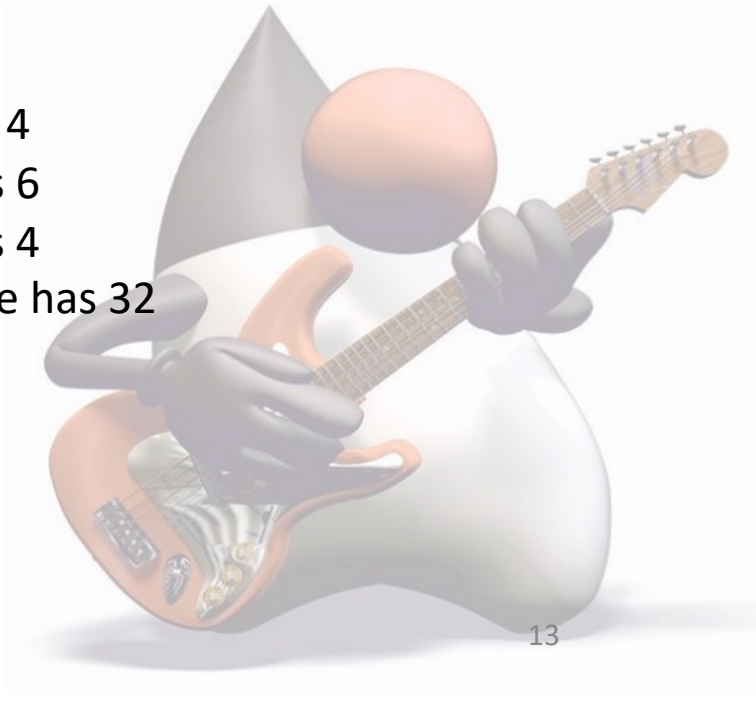
Hidden Fields Problematik

```
public class Vehicle {

    int seats = 4;
    int passengers;

    public Vehicle() {
    }

    public Vehicle(int initialSeats) {
        this.seats = initialSeats;
    }

    public void add1Passenger() {
        if (hasFreeSeat()) {
            passengers += 1;
        } else {
            printMessage();
        }
    }

    public boolean hasFreeSeat() {
        return passengers < seats;
    }

    public void printMessage() {
        System.out.println("No free seat found in the Vehicle");
    }
```

```
public class Bus extends Vehicle{

        int seats = 32;

        public Bus(){

        }

        public Bus(int seats){
            this.seats = seats;
        }

}
```

```
public static void main(String[] args) {
        Vehicle car = new Vehicle();
        System.out.println("Vehicle car has " + car.seats);
        Vehicle van = new Vehicle(6);
        System.out.println("Vehicle van has " + van.seats);
        Vehicle bus = new Bus();
        System.out.println("Vehicle bus has " + bus.seats);
        Bus busType = new Bus();
        System.out.println("Vehicle busType has " + busType.seats);
    }
```

Ausgabe:
Vehicle car has 4
Vehicle van has 6
Vehicle bus has 4
Vehicle busType has 32



Vererbung

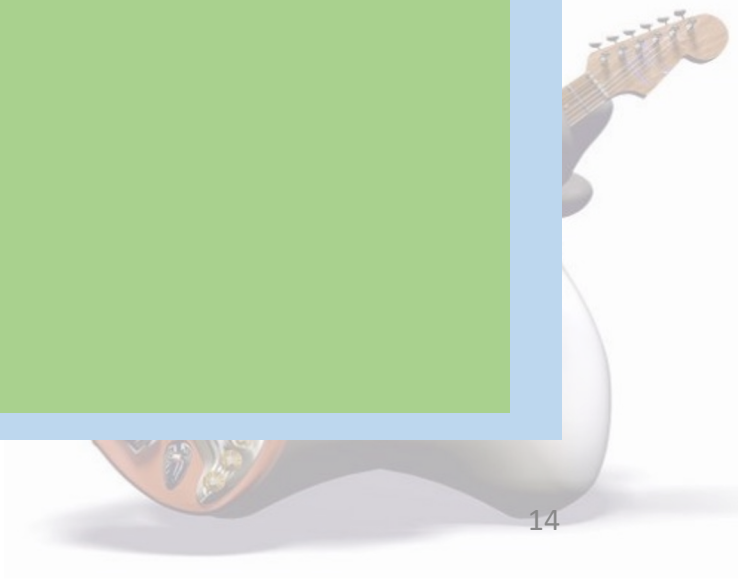
Eine Klasse kann mithilfe des Schlüsselwortes extends von einer anderen Klasse erben

Klasse die von einer anderen Klasse erben besitzen alle ihre Methoden und Attribute.

Können auch weiter spezialisiert werden

Attribute mit selben Namen werden in der Subklasse versteckt

Eine Subklasse kann einer Superklassen Referenz zugewiesen werden



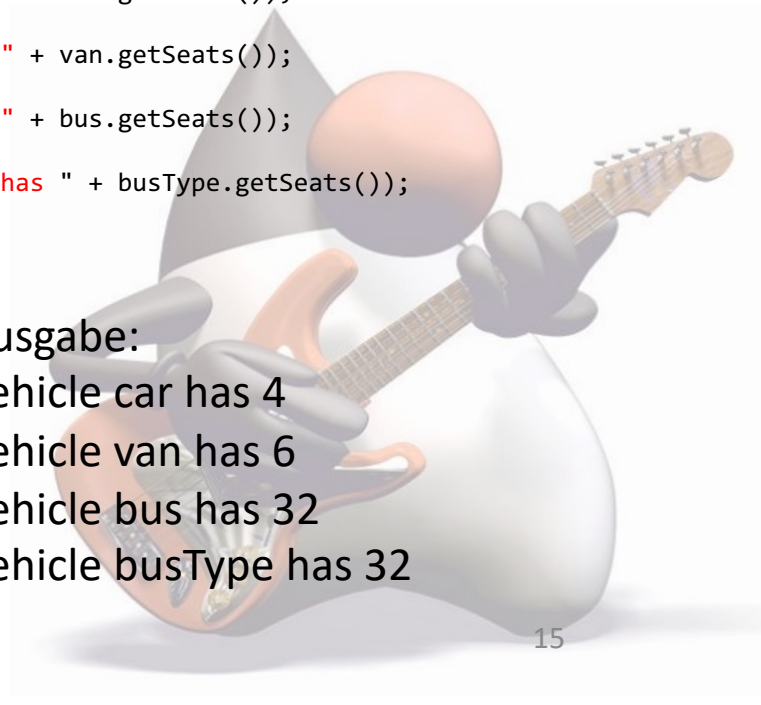
Why - getter and setter

```
public class Vehicle {

    int seats = 4;
    int passengers;

    public Vehicle() {
    }

    public Vehicle(int initialSeats) {
        this.seats = initialSeats;
    }

    public void add1Passenger() {
        if (hasFreeSeat()) {
            passengers += 1;
        } else {
            printMessage();
        }
    }

    public int getSeats(){
         return seats;
       }


    public boolean hasFreeSeat() {
        return passengers < seats;
    }

    public void printMessage() {
        System.out.println("No free seat found in the Vehicle");
    }
```

```
public class Bus extends Vehicle{

        int seats = 32;

        public int getSeats(){
          return seats;
        }
}
```

```
 public static void main(String[] args) {
        Vehicle car = new Vehicle();
        System.out.println("Vehicle car has " + car.getSeats());
        Vehicle van = new Vehicle(6);
        System.out.println("Vehicle van has " + van.getSeats());
        Vehicle bus = new Bus();
        System.out.println("Vehicle bus has " + bus.getSeats());
        Bus busType = new Bus();
        System.out.println("Vehicle busType has " + busType.getSeats());
    }
```

Ausgabe:
Vehicle car has 4
Vehicle van has 6
Vehicle bus has 32
Vehicle busType has 32



Vererbung

Eine Klasse kann mithilfe des Schlüsselwortes extends von einer anderen Klasse erben

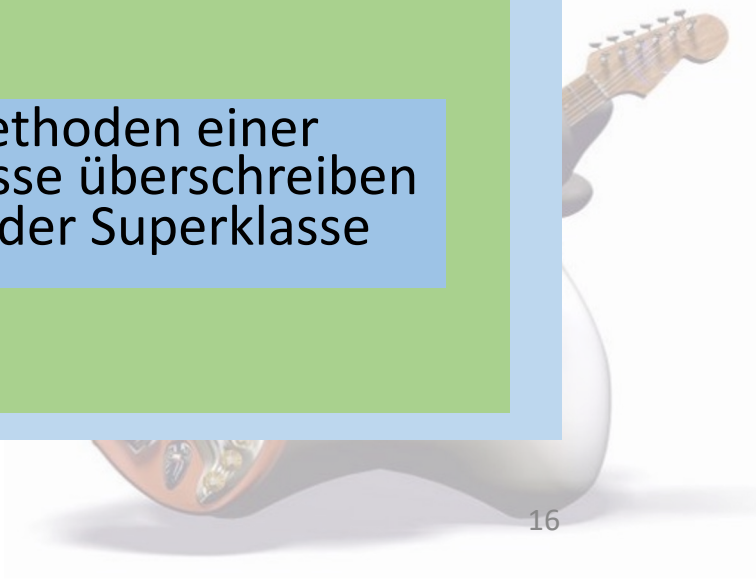
Klasse die von einer anderen Klasse erben besitzen alle ihre Methoden und Attribute.

Können auch weiter spezialisiert werden

Attribute mit selben Namen werden in der Subklasse versteckt

Eine Subklasse kann einer Superklassen Referenz zugewiesen werden

Methoden einer Subklasse überschreiben jene der Superklasse



Equals und HashCode

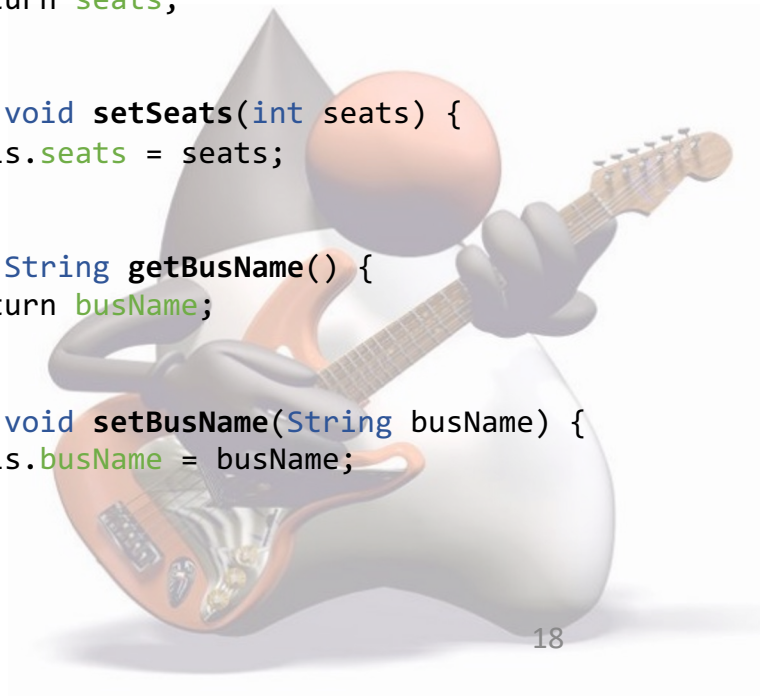
- Wann sind selbstdefinierte Objekte equals() == true ?

Equals und Hashcode

- Nehmen wir unsere Klasse Bus
  - Erstellen wir 2 Objekte
  - `Bus bus1 = new Bus(„MAN001“);`
  - `Bus bus2 = new Bus(„MAN001“);`
- If(bus1 == bus2) -> false; //selbe referenz?
- If(bus1.equals(bus2) -> false; //equals?

- Was müssen wir tun?

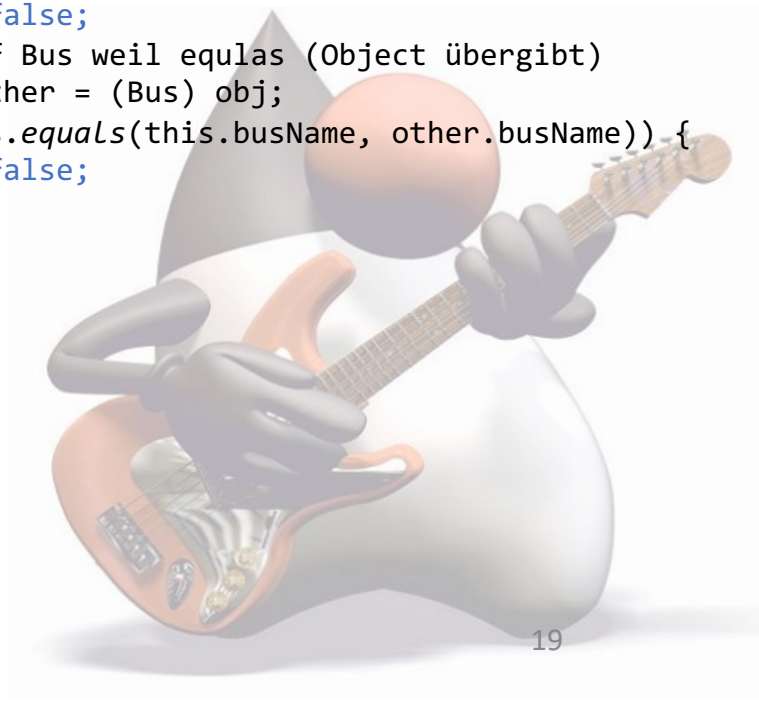
```
public class Bus extends Vehicle {

    int seats = 32;
    private String busName;

    public Bus() {
    }

    public Bus(int initialSeats) {
        super(initialSeats);
    }

    public Bus(String busName) {
        this.busName = busName;
    }

    public int getSeats() {
        return seats;
    }

    public void setSeats(int seats) {
        this.seats = seats;
    }

    public String getBusName() {
        return busName;
    }

    public void setBusName(String busName) {
        this.busName = busName;
    }
}
```



Equals und Hashcode

- Nehmen wir unsere Klasse Bus
  - Erstellen wir 2 Objekte
  - `Bus bus1 = new Bus(„MAN001“);`
  - `Bus bus2 = new Bus(„MAN001“);`
- If(bus1 == bus2) -> false; //selbe referenz?
- If(bus1.equals(bus2) -> true!; //equals?

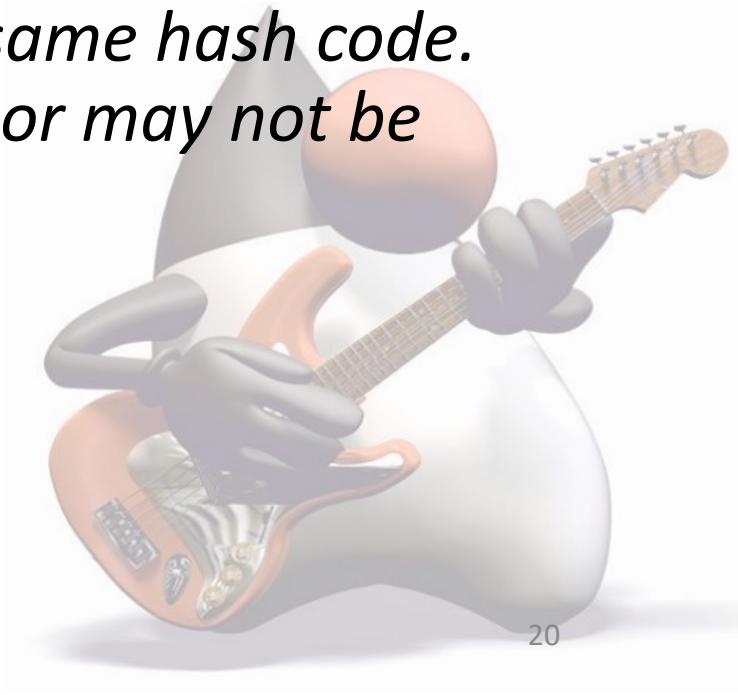
```
public class Bus extends Vehicle {

    int seats = 32;
    private String busName;
//constructors und getter / setter

 @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        } //cast auf Bus weil equlas (Object übergibt)
        final Bus other = (Bus) obj;
        if (!Objects.equals(this.busName, other.busName)) {
            return false;
        }
        return true;
    }

}
```



Wieso HashCode?

- Bestimmte Klassen aus dem Collections Framework (siehe Collections) benötige überschriebene hashCode für die korrekte und performante Funktionsweise. Stichwort: **Equals HashCode – Vertrag:**
- *The contract between equals() and hashCode() is:*
  1) If two objects are equal, then they must have the same hash code.
  2) If two objects have the same hash code, they may or may not be equal.

```
@Override
  public int hashCode() {
     int hash = 7;
     hash = 43 * hash + Objects.hashCode(this.busName);
     return hash;
  }
```

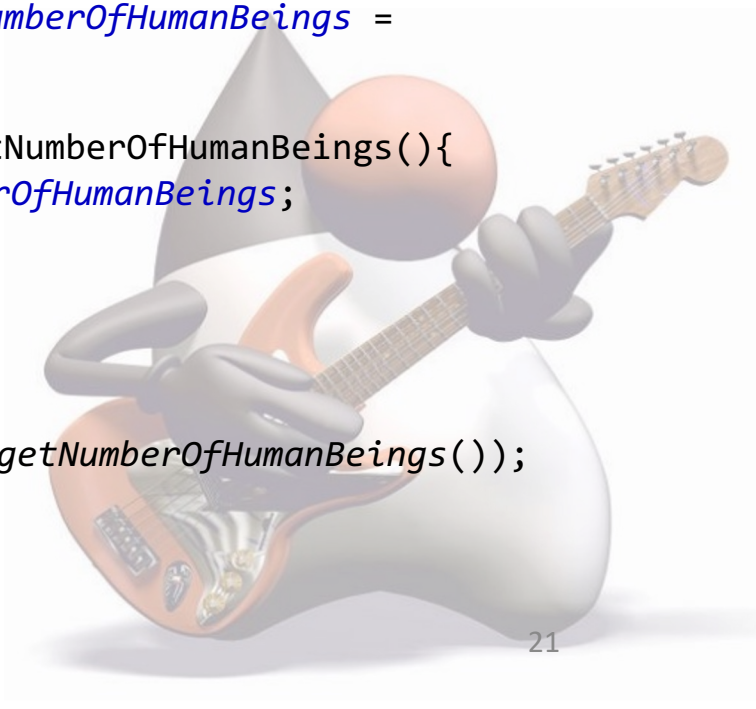


Statische Variablen

- Statische Variablen können über den Klassennamen angesteuert werden. `System.out.println(HomoSapiens.numberOfHumanBeings);`
- Es gibt auch statische Methoden

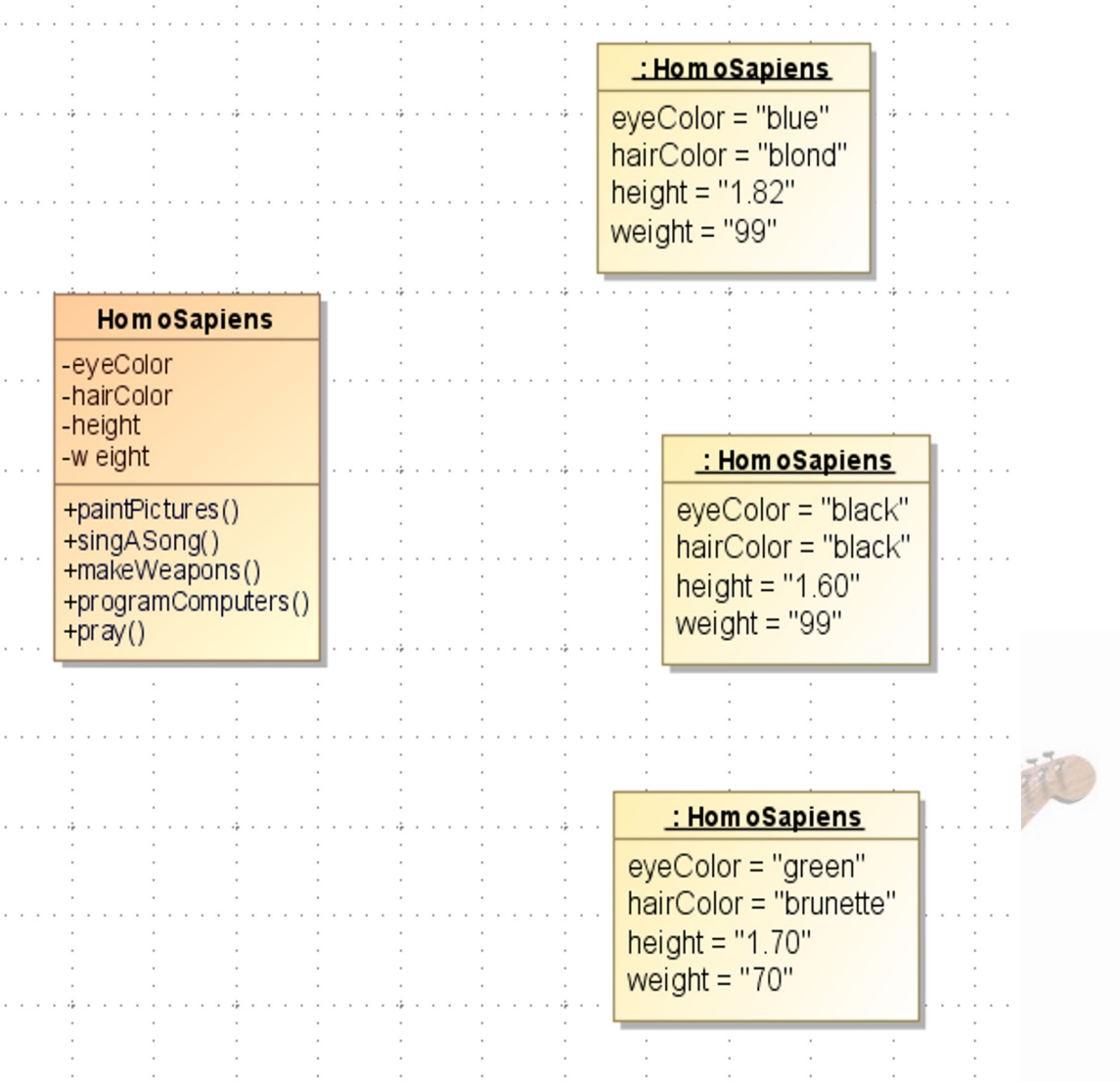
```
public class HomoSapiens extends CulturalHuman{
        private static long numberOfHumanBeings =
7876765123L;

        public static long getNumberOfHumanBeings(){
                return numberOfHumanBeings;
        }

        // ...
}
...
System.out.println(HomoSapiens.getNumberOfHumanBeings());
```

- Statische Methoden können aber nur statische Attribute verwenden.



Statische Attribute und Methoden

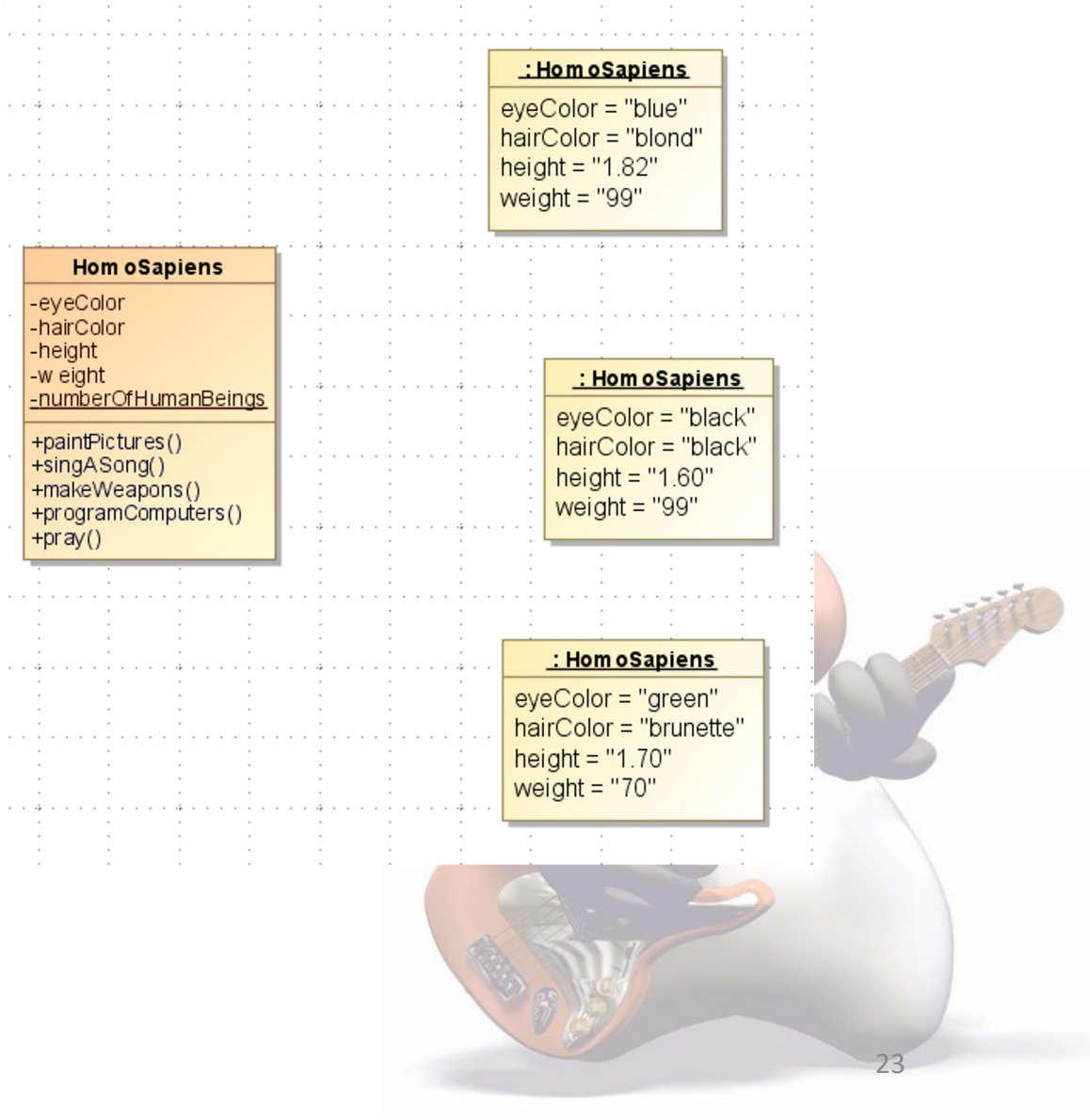
- Attribute werden im Objekt gespeichert
- Manchmal möchte man aber nur einen Wert speichern, nicht einen Wert pro Objekt



Lösung

- Wir können Attribute in der Klasse speichern

```
public class HomoSapiens extends CulturalHuman{
        public static long numberOfHumanBeings = 7876765123L;

        // ...

}
```



Übung 8

- Implementiere eine Klasse SavingAccounts als eine Subklasse von Account
- SavingAccount soll ein Attribut interestRate bekommen
- SavingAccount soll die Methode withdraw von Account überschreiben.