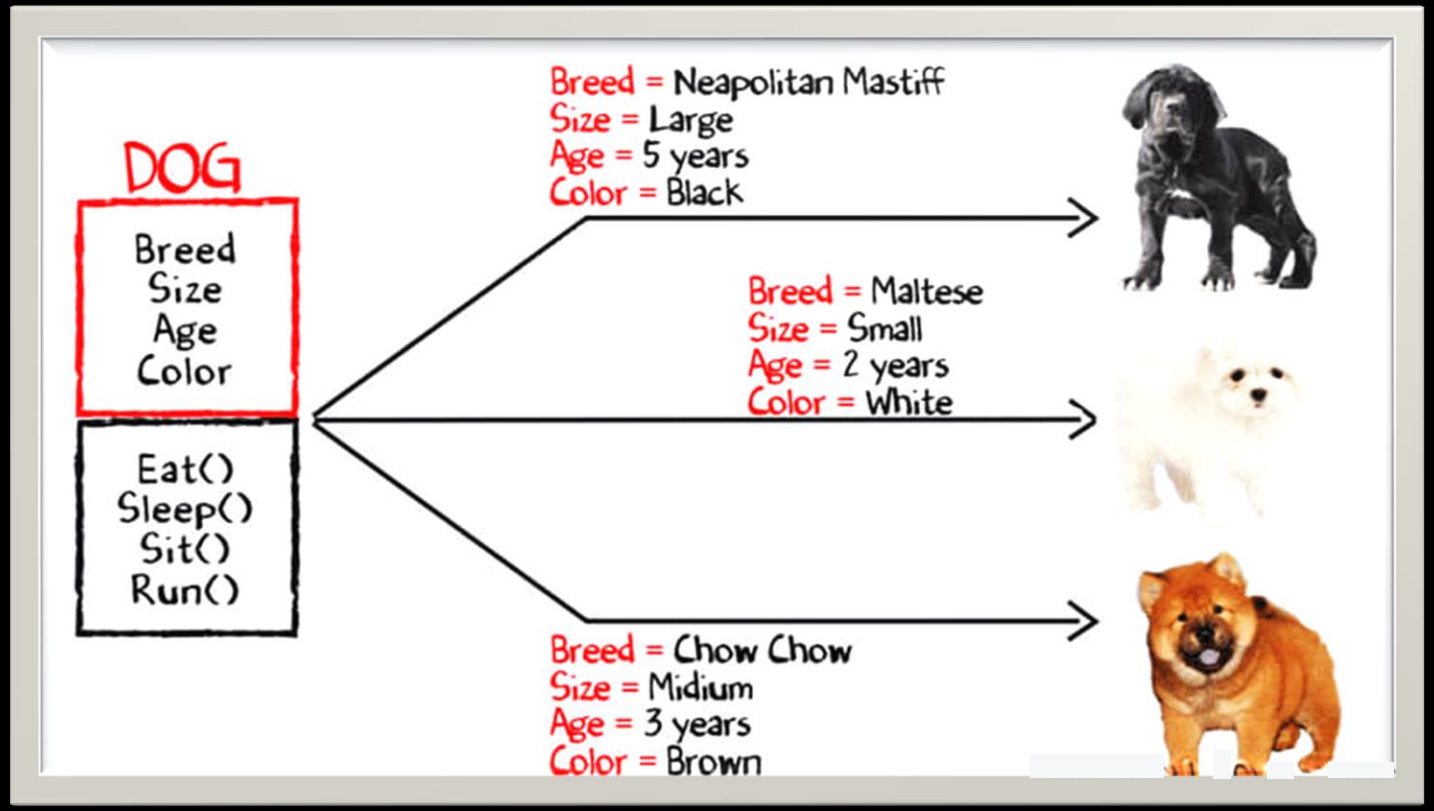
Informatik Q1 Abels

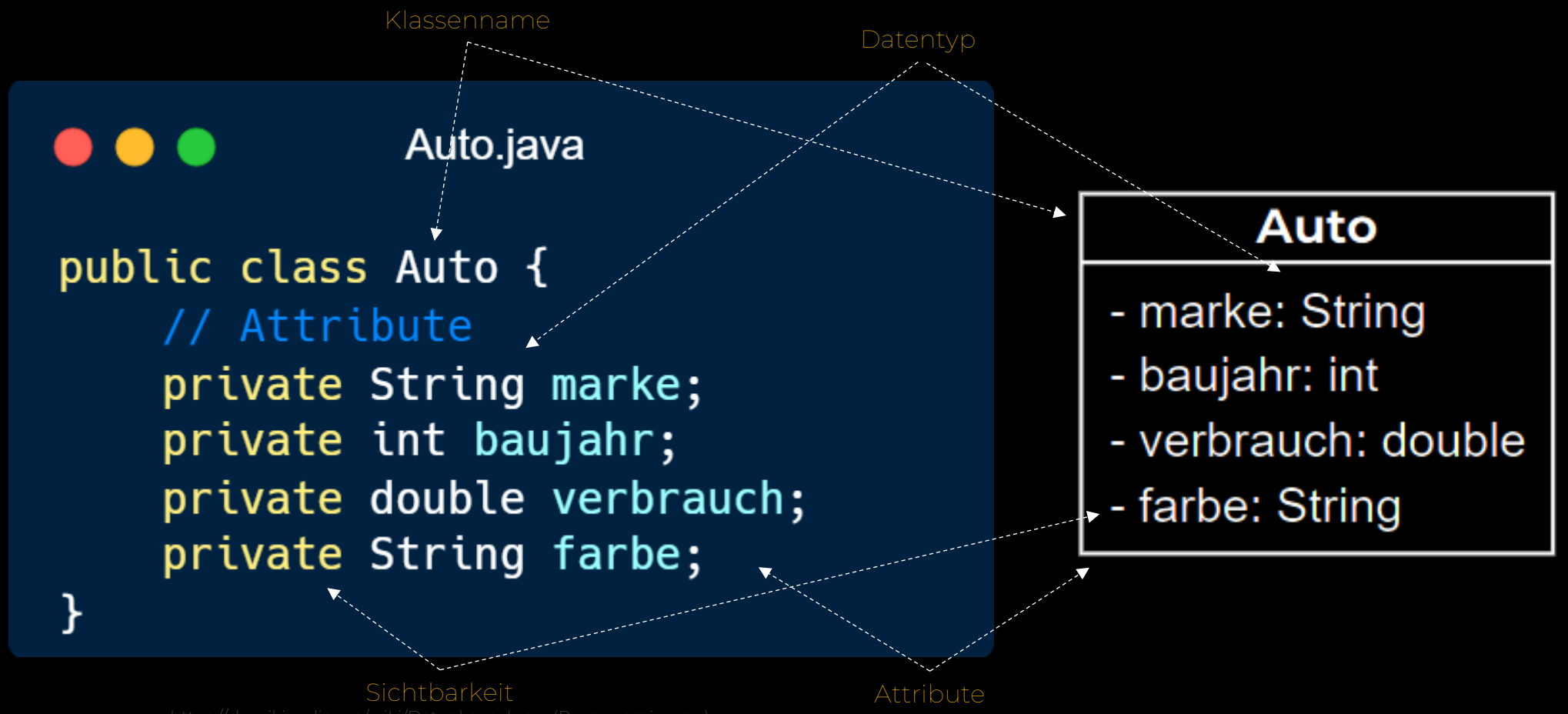
Klasse vs Objekt

Klasse vs Objekt



Klassen sind Vorlagen, aus denen Instanzen genannte Objekte zur Laufzeit erzeugt werden.

Klasse vs Objekt

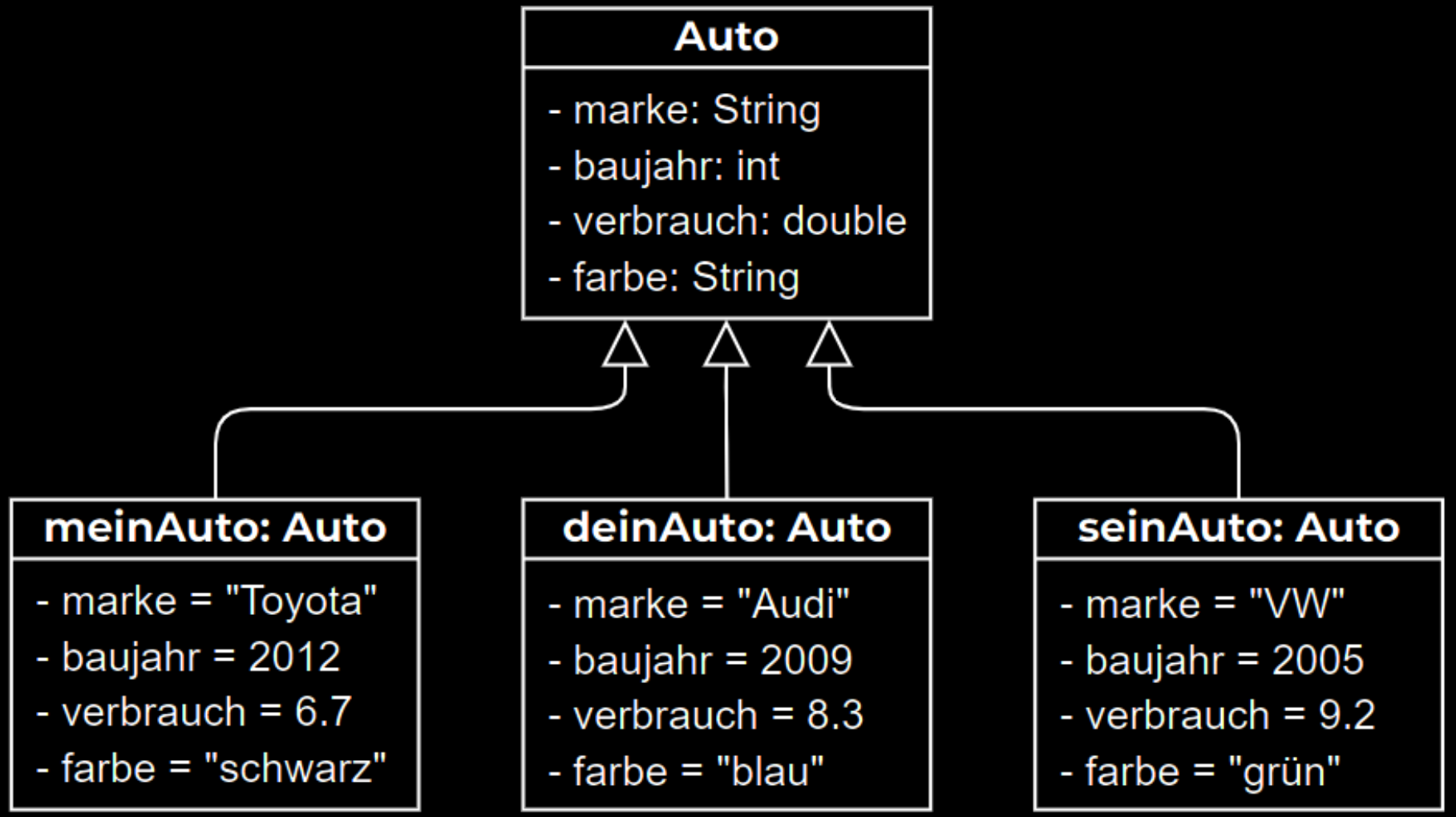


https://de.wikipedia.org/wiki/Datenkapselung_(Programmierung)

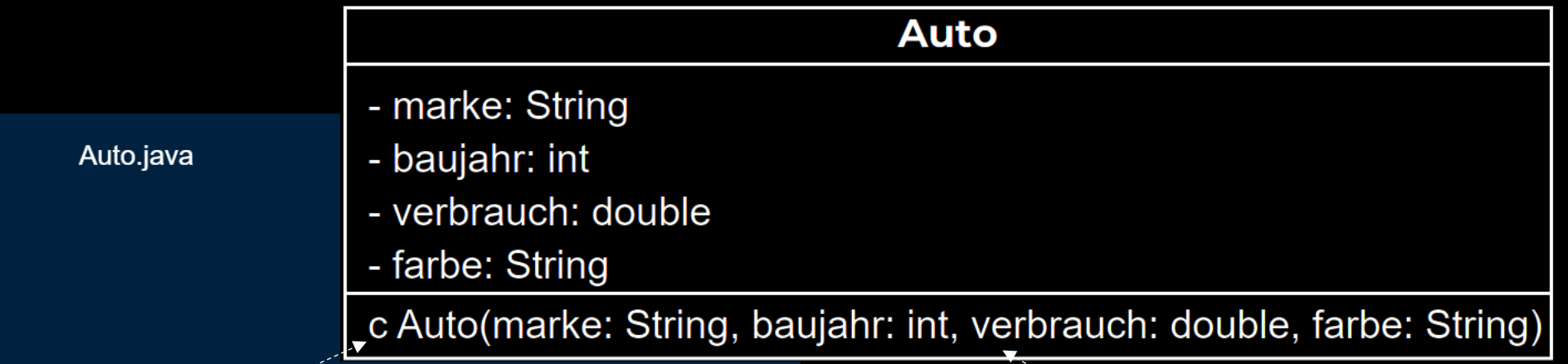
Quellcode

UML

Klasse vs Objekt



Konstruktor



Auto.java

```java
public class Auto {
    // Attribute
    private String marke;
    private int baujahr;
    private double verbrauch;
    private String farbe;

    // Konstruktor
    public Auto(String marke, int baujahr, double verbrauch, String farbe) {
        this.marke = marke;
        this.baujahr = baujahr;
        this.verbrauch = verbrauch;
        this.farbe = farbe;
    }
}
```

Constructor

Parameter zur Erzeugung
(Instanziierung) eines neuen Objekts

Bezug auf das aktuelle Objekt

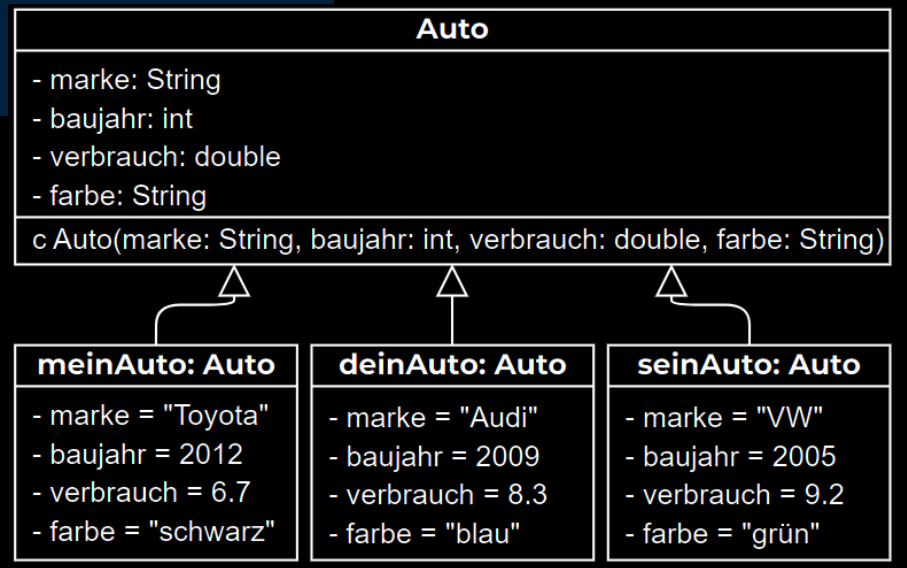
Hauptklasse

Autohaus.java

```java
public class Autohaus {
    public static void main(String[] args) {
        Auto meinAuto = new Auto(“Toyota”, 2012, 6.7, “schwarz”);
        Auto deinAuto = new Auto(“Audi”, 2009, 8.3, “blau”);
        Auto seinAuto = new Auto(“VW”, 2005, 9.2, “grün”);
    }
}
```

Auto nun als Datentyp

Aufruf des Constructors mit den gegebenen Parametern



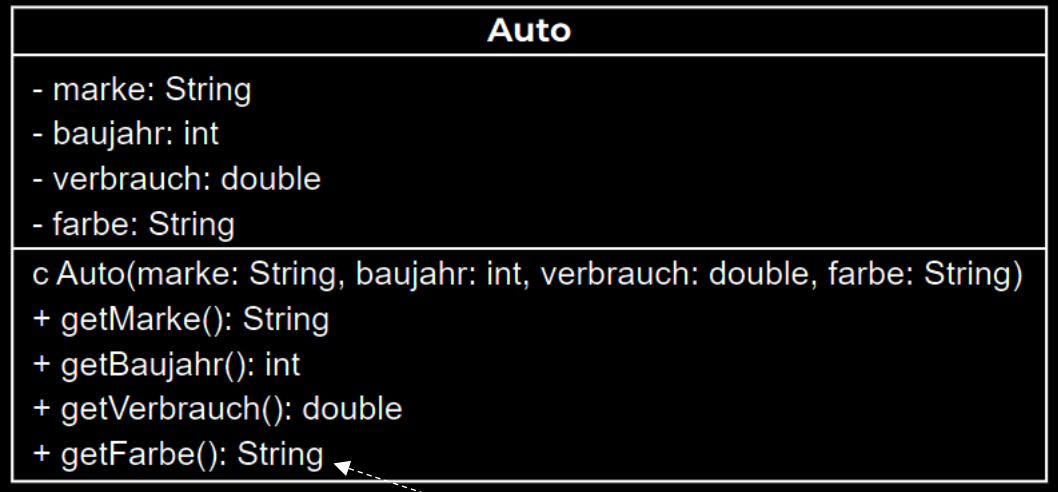
get-Methode

Auto.java

```java
public class Auto {
    // Attribute
    private String marke;
    private int baujahr;
    private double verbrauch;
    private String farbe;

    // Konstruktor
    public Auto(String marke, int baujahr, double verbrauch, String farbe) {
        this.marke = marke;
        this.baujahr = baujahr;
        this.verbrauch = verbrauch;
        this.farbe = farbe;
    }

    // Getter
    public String getMarke() {
        return this.marke;
    }

    public int getBaujahr() {
        return this.baujahr;
    }

    public double getVerbrauch() {
        return this.verbrauch;
    }

    public String getFarbe() {
        return this.farbe;
    }
}
```

Autohaus.java

```java
public class Autohaus {
    public static void main(String[] args) {
        Auto meinAuto = new Auto("Toyota", 2012, 6.7, "schwarz");
        Auto deinAuto = new Auto("Audi", 2009, 8.3, "blau");
        Auto seinAuto = new Auto("VW", 2005, 9.2, "grün");

        System.out.println(meinAuto.getMarke());
    }
}
```



set-Methode

Auto.java

```java
public class Auto {
    // Attribute
    private String marke;
    private int baujahr;
    private double verbrauch;
    private String farbe;

    // Konstruktor
    public Auto(String marke, int baujahr, double verbrauch, String farbe) {
        this.marke = marke;
        this.baujahr = baujahr;
        this.verbrauch = verbrauch;
        this.farbe = farbe;
    }

    // Getter & Setter
    public String getMarke() {
        return this.marke;
    }

    public void setMarke(String marke) {
        this.marke = marke;
    }

    public int getBaujahr() {
        return this.baujahr;
    }

    public void setBaujahr(int baujahr) {
        this.baujahr = baujahr;
    }

    public double getVerbrauch() {
        return this.verbrauch;
    }

    public void setVerbrauch(double verbrauch) {
        this.verbrauch = verbrauch;
    }

    public String getFarbe() {
        return this.farbe;
    }

    public void setFarbe(String farbe) {
        this.farbe = farbe;
    }
}
```

Autohaus.java

```java
public class Autohaus {
    public static void main(String[] args) {
        Auto meinAuto = new Auto("Toyota", 2012, 6.7, "schwarz");
        Auto deinAuto = new Auto("Audi", 2009, 8.3, "blau");
        Auto seinAuto = new Auto("VW", 2005, 9.2, "grün");

        System.out.println(meinAuto.getMarke());
        meinAuto.setMarke("Mercedes");
        System.out.println(meinAuto.getMarke());
    }
}
```

Übung 1

Gegeben sei eine Klasse namens **`Bus.java`**. Die Klasse Bus hat Attribute für den Namen des Fahrers und die Zahl der Gäste. Für beide Attribute besitzt sie get-/set-Methoden und einen entsprechenden Konstruktor. Außerdem hat sie zwei Methoden für das Zu- bzw. Aussteigen von `n` Personen.

a) Entwirf ein UML-Diagramm für die Klasse **`Bus`**.

b) Erweitere das UML-Diagramm um zwei Objekte `bus1` und `bus2`.

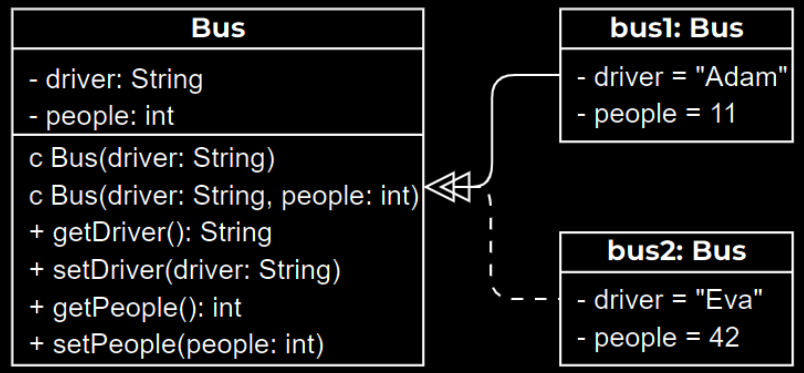
c) Implementiere die Klasse **`Bus`** in Java.

d) Implementiere die Klasse **`Verkehr`**, in der die beiden Busse `bus1` und `bus2` mit erfundenen Daten instanziiert werden.

e) Gib den Fahrer des Busses mit den meisten Gästen in der Console aus.

f) Befördere 5 Gäste vom Bus mit mehr Gästen zum Bus mit weniger Gästen.

Übung 1



Bus.java

```java
public class Bus {
    // Attribute
    private String driver;
    private int people;

    // Konstruktor
    public Bus (String driver) {
        this.driver = driver;
        this.people = 0;
    }

    public Bus (String driver, int people) {
        this.driver = driver;
        this.people = people;
    }

    // Getter & Setter
    public String getDriver () {
        return this.driver;
    }

    public void setDriver (String driver) {
        this.driver = driver;
    }

    public int getPeople () {
        return this.people;
    }

    public void setPeople (int people) {
        this.people = people;
    }

    // Methoden
    public void zusteigen (int n) {
        this.people = this.people + n;
    }

    public void aussteigen (int n) {
        this.people = this.people - n;
    }
}
```

Verkehr.java

```java
public class Verkehr {
    public static void main(String[] args) {
        // d)
        Bus bus1 = new Bus("Adam", 11);
        Bus bus2 = new Bus("Eva", 42);

        // e)
        if (bus1.getPeople() > bus2.getPeople()) {
            System.out.println(bus1.getDriver());
        } else {
            System.out.println(bus2.getDriver());
        }

        // f)
        if (bus1.getPeople() > bus2.getPeople()) {
            bus1.aussteigen(5);
            bus2.zusteigen(5);
        } else {
            bus2.aussteigen(5);
            bus1.zusteigen(5);
        }
    }
}
```

Übung 2

Dreieck
- a: double - b: double - c: double
c Dreieck(a: double, b: double, c: double) + getA(): double + setA(aNew: double) + getB(): double + setB(bNew: double) + getC(): double + setC(cNew: double) + area(): double

In einem neuen Ordner **Geometrie** wollen wir geometrische Objekte betrachten.

a) Erstelle eine Klasse **Dreieck** nach folgendem UML.

b) Implementiere die Funktion `public double area()`.

c) Erstelle eine Klasse **Main**, in der das Objekt **d1** vom Typ **Dreieck** mit willkürlichen Zahlen instanziiert und dessen Flächeninhalt berechnet sowie ausgegeben wird.

d) Erstelle eine Klasse **Punkt** mit den Parametern `double x` und `double y` inklusive Gettern, Settern und einem Konstruktor.

e) Erweitere die Klasse Punkt um die Funktion `public double distance(Punkt p)`.

f) Erstelle in der **Main**-Klasse ein weiteres **Dreieck**-Objekt **d2** durch Angabe dreier **Punkt**-Objekte und gib dessen Flächeninhalt aus. Was musst du dafür noch in der **Dreieck**-Klasse ergänzen?

Tagebucheintrag

Klasse vs Objekt

- UML
- Attribute
- Konstruktor
- Getter & Setter
- Sichtbarkeit / Geheimnisprinzip / Datenkapselung

Wochenübung

In einem neuen Ordner `Geometrie` wollen wir verschiedene geometrische Objekte betrachten.

a) Erstelle jeweils eine Klasse für die Objekte **`Dreieck`**, **`Zyliner`** und **`Kreis`**.

b) Erweitere die Klassen mit den Attributen, die diese Objekte eindeutig bestimmen.

c) Erweitere die Klassen mit den jeweiligen get-/set-Methoden und Konstruktoren.

d) Erweitere die Klassen mit Methoden zu Berechnung des Flächeninhalts bzw. Volumens.

e) Erweitere die Klassen mit weiteren Konstruktoren und Methoden.