Handout: Objektorientiertes Design in Java

Einführung in Klassen und Objekte

In der objektorientierten Programmierung (OOP) sind **Klassen** und **Objekte** die zentralen Konzepte. Eine Klasse ist ein **Bauplan**, der beschreibt, wie Objekte aufgebaut werden. Objekte sind **Instanzen** einer Klasse, die spezifische Daten und Verhalten kapseln.

Vorteile von Klassen und Objekten

- **Kapselung**: Daten und Methoden, die auf diese Daten zugreifen, sind in einer Einheit zusammengefasst.
- **Wiederverwendbarkeit**: Einmal definierte Klassen können für verschiedene Anwendungen verwendet werden.
- **Modularität**: Programmcode ist strukturierter und leichter wartbar.
- **Flexibilität und Erweiterbarkeit**: Klassen können durch Vererbung erweitert werden.

Aufbau von Klassen

Eine Klasse besteht aus:

1. **Attributen**: Variablen, die den Zustand eines Objekts beschreiben.
2. **Methoden**: Funktionen, die das Verhalten eines Objekts definieren.

Struktur einer Klasse

```
class Klassenname {
    // Attribute (Zustand)
    Datentyp attributName;

    // Methoden (Verhalten)
    Rückgabewert methodenName(Parameterliste) {
        // Methodenkörper
    }
}
```

Beispiel: Klasse `Auto`

```
class Auto {
    // Attribute
    String farbe;
    int maximaleGeschwindigkeit;
    int momentaneGeschwindigkeit;
    String lenkradAusrichtung;
```

```
    // Methoden
    void beschleunigen(int geschwindigkeit) {
        momentaneGeschwindigkeit += geschwindigkeit;
    }

    void bremsen(int geschwindigkeit) {
        momentaneGeschwindigkeit -= geschwindigkeit;
        if (momentaneGeschwindigkeit < 0) {
            momentaneGeschwindigkeit = 0;
        }
    }

    void lenken(String richtung) {
        lenkradAusrichtung = richtung;
    }
}
```

Objekte: Zugriff, Wertzuweisung und Gültigkeit

Ein **Objekt** ist eine Instanz einer Klasse. Es wird mit dem Schlüsselwort `new` erstellt.

Objekte erstellen

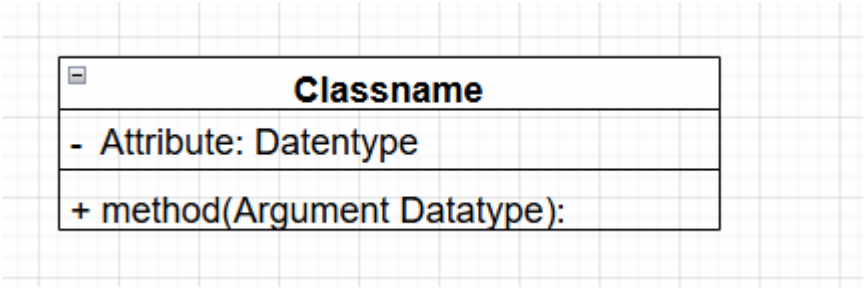
```
public class Main {
    public static void main(String[] args) {
        // Objekte der Klasse Auto erstellen
        Auto rotesAuto = new Auto();
        rotesAuto.farbe = "Rot";
        rotesAuto.maximaleGeschwindigkeit = 120;

        Auto gelbesAuto = new Auto();
        gelbesAuto.farbe = "Gelb";
        gelbesAuto.maximaleGeschwindigkeit = 110;

        // Attribute ändern
        rotesAuto.beschleunigen(50);
        System.out.println("Momentane Geschwindigkeit: " +
rotesAuto.momentaneGeschwindigkeit);
    }
}
```

Eigenschaften eines Objekts ändern

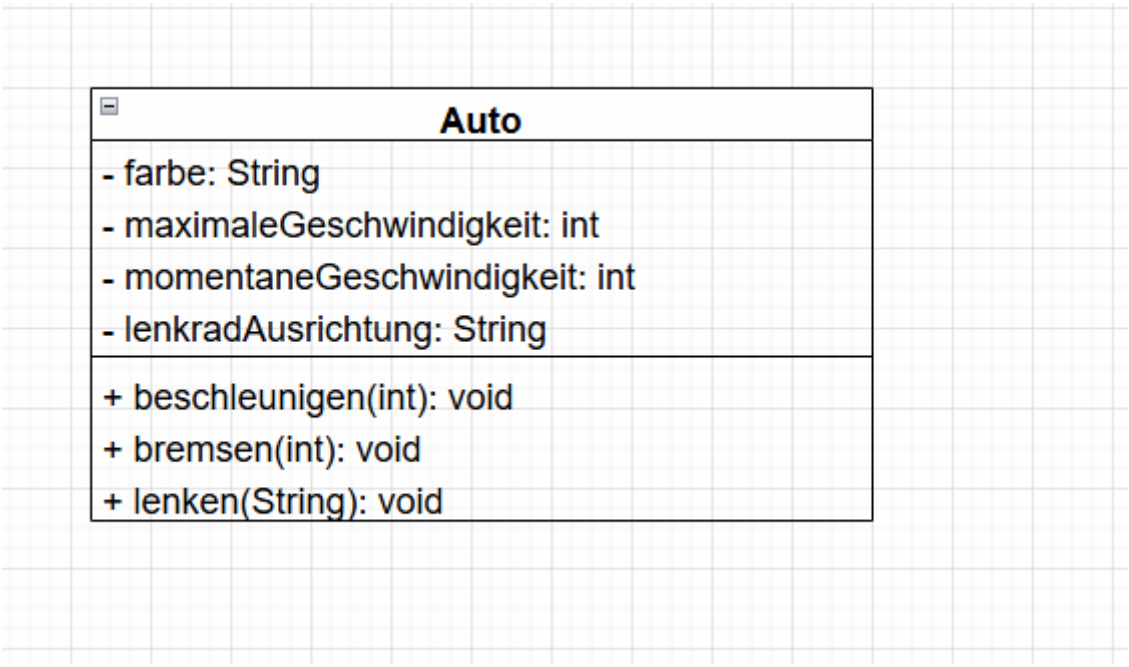
1. Attribute werden direkt oder über Methoden verändert.
2. Methoden werden aufgerufen, um das Verhalten eines Objekts zu definieren.

UML-Klassendiagramme

UML-Klassendiagramme sind grafische Darstellungen von Klassen und deren Beziehungen.



Beispiel: Klassendiagramm `Auto`



- **+**: öffentliche Elemente (`public`)
- **-**: private Elemente (`private`)

Erweiterung: Instanzen des Autos

Erstellen von fünf Autos mit unterschiedlichen Attributen:

```
public class Main {
    public static void main(String[] args) {
        Auto rotesAuto = new Auto();
        rotesAuto.farbe = "Rot";
        rotesAuto.maximaleGeschwindigkeit = 120;

        Auto gelbesAuto = new Auto();
        gelbesAuto.farbe = "Gelb";
        gelbesAuto.maximaleGeschwindigkeit = 110;

        Auto gruensAuto = new Auto();
        gruensAuto.farbe = "Grün";
        gruensAuto.maximaleGeschwindigkeit = 100;
```

```
        Auto blauesAuto = new Auto();
        blauesAuto.farbe = "Blau";
        blauesAuto.maximaleGeschwindigkeit = 110;

        Auto lilaAuto = new Auto();
        lilaAuto.farbe = "Lila";
        lilaAuto.maximaleGeschwindigkeit = 100;
    }
}
```

Zusammenfassung

Begriff	Beschreibung
Klasse	Ein Bauplan für Objekte. Beschreibt Zustand (Attribute) und Verhalten (Methoden).
Objekt	Eine Instanz einer Klasse mit spezifischen Werten für die Attribute.
Attribute	Variablen innerhalb einer Klasse, die den Zustand eines Objekts speichern.
Methoden	Funktionen innerhalb einer Klasse, die das Verhalten eines Objekts definieren.
UML-Klassendiagramm	Grafische Darstellung einer Klasse mit ihren Attributen und Methoden.

Vorteile von Klassen

1. **Modularität**: Erleichtert die Wartung und Erweiterung des Codes.
2. **Wiederverwendbarkeit**: Klassen können in verschiedenen Programmen genutzt werden.
3. **Datenkapselung**: Sorgt für besseren Schutz und Organisation der Daten.

Mit Klassen und Objekten wird die Programmierung in Java strukturierter, wiederverwendbarer und leichter wartbar.