Überblick

- Von C nach Java
- Background
- Konzepte des objektorientierten Programmierens
- Ein ausgearbeitetes Java-Programm als Beispiel
- Ein weiteres Java-Programm als Beispiel
- Typen und Subtypen
- Vom Design zum Programmieren (OOD zu OOP)
- Patterns in OOP
- Testen (von objektorientierten Programmen)

Ein ausgearbeitetes Java-Programm als Beispiel

- Aufgabenstellung
- Überlegungen
- Klassenhierarchie und Polymorphismus
- Klassen
- Main-Methode

Aufgabenstellung

- OO Programm zur Verwaltung von Fahrzeugen für ein Geschäft
- Fahrzeuge: Autos und Fahrräder
- Attribute von Fahrzeugen: Name und Preis
- Zusätzliches Attribut von Autos: Kennzeichen
- Zusätzliches Attribut von Fahrrädern: Typ
- OO Kapselung
- Räderanzahl für ein Fahrzeug bestimmen
- Gesamtanzahl von Rädern ermitteln
- Teuerstes Fahrzeug ermitteln

Überlegungen

- Fokus auf Objekten anstatt Prozeduren
- OO Programm (in Java) besteht aus Objekt-Klassen.
- Klassenhierarchie als Gerüst
- Polymorphismus zum Bestimmen der Räderanzahl
- Zugriffs-Methoden für OO Kapselung

Klassenhierarchie und Polymorphismus

```
public abstract class Fahrzeug {
    public abstract int anzahlRaeder();
}

public class Auto extends Fahrzeug {
    public int anzahlRaeder() {
        return 4;
    }
}

public class Fahrrad extends Fahrzeug {
    public int anzahlRaeder() {
        return 2;
    }
}
```


Klassen – Fahrzeug

```
public abstract class Fahrzeug {
    private String name = null;
    private double preis = 0;
    //Konstruktor für die Klasse Fahrzeug:
    //Jedem Fahrzeug wird beim Instanziieren ein Name zugewiesen.
    public Fahrzeug(String name) {
        this.name = name;
    }
    public void setPreis(double preis) {
        this.preis = preis;
    }
    public double getPreis() {
        return this.preis;
    }
    public abstract int anzahlRaeder();
}
```

Klassen – Auto

```
public class Auto extends Fahrzeug {
    private String kennzeichen = null;
    //Konstruktor für die Klasse Auto:
    public Auto(String name, String kennzeichen) {
        super(name);
        this.kennzeichen = kennzeichen;
    }
    public int anzahlRaeder() {
        return 4;
    }
}
```


Klassen – Fahrrad

```
public class Fahrrad extends Fahrzeug {
    private String typ = null;
    //Konstruktor für die Klasse Fahrrad:
    public Fahrrad(String name, String typ) {
        super(name);
        this.typ = typ;
    }
    public int anzahlRaeder() {
        return 2;
    }
}
```

Klassen – Geschaeft

```
public class Geschaeft {
    private Liste fahrzeugListe;
    //Konstruktor für die Klasse Geschaeft:
    public Geschaeft() {
        Erzeuge leere Liste
        Erzeuge Fahrzeug-Instanzen und trage sie in
        Liste ein
    }
    private int anzahlRaeder() {
        Ermittle Anzahl der Raeder aller Fahrzeuge im Geschaeft
    }
    private Fahrzeug teuerstesFahrzeug() {
        Ermittle das teuerste Fahrzeug im Geschaeft
    }
    Main-Methode
}
```


Main-Methode

```
//Main-Methode in Klasse Geschaeft:
//Wird beim Aufruf des Programms gestartet.
//Im Stringarray args werden die Kommando-Zeilen-Parameter
//übergeben.
public static void main(String[] args) {
        Geschaeft autohausmaier = new Geschaeft();
        int gesamtanzahlraeder = autohausmaier.anzahlRaeder();
        Fahrzeug teuerstesfz = autohausmaier.teuerstesFahrzeug();
}
```