Lernzettel

Objektorientierte Analyse und Design: Klassen, Objekte, Attribute und Methoden

Universität:	Technische Universität Berlin
Kurs/Modul:	Einführung in die Informatik
Erstellungsdatum:	September 19, 2025

Lernzettel: Objektorientierte Analyse und Design: Klassen, Objekte, Attribute und Methoden

(1) Grundbegriffe.
Eine Klasse ist der Bauplan für Objekte. Sie definiert Attribute (Daten) und Methoden (Verhalten). Ein Objekt ist eine konkrete Instanz einer Klasse. Attribute speichern den Zustand, Methoden implementieren das Verhalten eines Objekts.

(2) Aufbau einer Klasse.
Attribute speichern Zustände (z. B. Name, Alter). Methoden implementieren Verhalten (z. B. getName, setName, berechneAlter). Sichtbarkeiten steuern den Zugriff:

- public: öffentlich zugänglich
- protected: innerhalb der Klasse und abgeleiteten Klassen sichtbar
- private: nur innerhalb der Klasse sichtbar

(3) Instanz vs Klassen-Mitglieder.

- Instanzattribute und Instanzmethoden gehören zu Objekten und können pro Objekt verschieden sein.
- Klassenmitglieder (z. B. statische/ Klassenmethoden oder Klassenattribute) gehören zur Klasse selbst und sind objektreduziert zugänglich.

(4) Kapselung und Abstraktion.

- Kapselung: Details der Implementierung verstecken; Zugriff über öffentliche Schnittstellen.
- Abstraktion: nur relevante Eigenschaften sichtbar; komplexe Details werden verborgen.

(5) Vererbung und Polymorphie.

- Vererbung erlaubt es, Eigenschaften (Attribute) und Verhalten (Methoden) einer Oberklasse zu übernehmen.
- Konstruktoren initialisieren den Zustand einer Instanz; Aufrufe von Oberklassen-Konstruktoren sind üblich.
- Polymorphie bedeutet, dass Objekte verschiedener Unterklassen über eine gemeinsame Basisklasse referenziert werden können.
- Abstrakte Klassen dienen als Vorlagen und können nicht direkt instanziiert werden.
- Schnittstellen definieren Verträge, ohne konkrete Implementierung.

(6) Design-Grundprinzipien.

- geringe Kopplung, hohe Kohäsion
- klare Verantwortlichkeiten, übersichtliche Schnittstellen
- Wiederverwendbarkeit und Erweiterbarkeit

(7) UML-Klassen-Diagramm (Kurzüberblick).

- Klassenname mit Attributen und Methoden (mit Sichtbarkeiten).
- Beziehungen: Vererbung (Pfeil von Unterklasse zu Oberklasse), Assoziation, Aggregation, Komposition.
- Notationen für Sichtbarkeiten: + public, - private, # protected.

(8) Beispiel: einfache Klasse und Nutzung (C++-Stil).

Im Folgenden sehen Sie eine kleine Klasse "Person" mit privaten Attributen und öffentlichen Zugriffsmethoden.

```
class Person {
private:
    std::string name;
    int alter;
public:
    Person(const std::string& n, int a) : name(n), alter(a) {}
    std::string getName() const { return name; }
    int getAlter() const { return alter; }
    void setName(const std::string& n) { name = n; }
    void setAlter(int a) { alter = a; }
};
```

Beispiel-Verwendung:

```
Person p("Alice", 30);
std::cout << p.getName() << " ist " << p.getAlter() << " Jahre alt.\n";
```

(9) Weiterführender Java-ähnlicher Stil (Kurzüberblick).

```
public class Person {
    private String name;
    private int alter;

    public Person(String name, int alter) {
        this.name = name;
        this.alter = alter;
    }
```

```
    public String getName() { return name; }
    public int getAlter() { return alter; }
    public void setName(String name) { this.name = name; }
    public void setAlter(int alter) { this.alter = alter; }
}
```