# Aufgabenblatt 3 - Programmierung in Java

Vorpraktikum Informatik 2024

Letztes Update des Aufgabenblattes: 6. September 2024

## Einleitung

Willkommen zum Aufgabenblatt für das Programmieren in Java. In diesem Blatt findest du eine Reihe von Aufgaben, die dir helfen sollen, die Grundlagen von Java zu verstehen, insbesondere den Umgang mit Ein- und Ausgaben, Schleifen, Verzweigungen sowie Klassen und Methoden.

## Aufgabe 1: Ausgabe eines einfachen Texts

**Aufgabe:** Schreibe ein Java-Programm, das eine einfache Nachricht auf dem Bildschirm ausgibt.

a) Erstelle ein Programm, das die Nachricht „Hello, World!“ auf dem Bildschirm ausgibt.

b) Erweitere das Programm, indem es den Namen vom Benutzer einliest und anschließend „Hello, [Name]!“ ausgibt. Verwende dazu die Klasse `Scanner` für die Benutzereingabe.

**Hinweise:**

- Verwende die Methode `System.out.println()` für die Ausgabe.
- Die Klasse `Scanner` wird für das Einlesen von Benutzereingaben verwendet.

## Aufgabe 2: Einlesen und Ausgeben von Werten

**Aufgabe:** Schreibe ein Java-Programm, das Benutzereingaben einliest und diese verarbeitet.

a) Erstelle ein Programm, das eine ganze Zahl vom Benutzer einliest und diese auf dem Bildschirm ausgibt.

b) Erweitere das Programm so, dass zusätzlich eine zweite Zahl eingelesen wird. Berechne die Summe der beiden Zahlen und gib sie aus.

## Aufgabe 3: Verwendung von `for`-Schleifen

**Aufgabe:** Schreibe ein Java-Programm, das mithilfe einer `for`-Schleife wiederholte Aktionen durchführt.

a) Erstelle ein Programm, das die Zahlen von 1 bis 10 auf dem Bildschirm ausgibt.

b) Modifiziere das Programm, sodass die Quadrate der Zahlen von 1 bis 10 ausgegeben werden.

c) Erweitere das Programm, sodass nur die geraden Zahlen zwischen 1 und 20 ausgegeben werden.

## Aufgabe 4: Verwendung von `while`-Schleifen

**Aufgabe:** Schreibe ein Java-Programm, das eine `while`-Schleife verwendet, um Benutzereingaben zu verarbeiten.

a) Erstelle ein Programm, das den Benutzer wiederholt nach Zahlen fragt, bis die Zahl 0 eingegeben wird. Gib nach jeder Eingabe die aktuelle Zahl aus.

b) Erweitere das Programm, sodass die Summe aller eingegebenen Zahlen ausgegeben wird, nachdem die Schleife endet (nach Eingabe von 0).

## Aufgabe 5: Verwendung von `switch`-Anweisungen

**Aufgabe:** Schreibe ein Java-Programm, das eine `switch`-Anweisung verwendet, um verschiedene Aktionen basierend auf Benutzereingaben auszuführen.

a) Erstelle ein Programm, das den Benutzer nach einer Zahl zwischen 1 und 3 fragt und basierend auf der Eingabe eine Nachricht ausgibt (z.B. „Du hast 1 gewählt.“).

b) Füge einen `default`-Fall hinzu, der eine Fehlermeldung ausgibt, wenn die eingegebene Zahl nicht 1, 2 oder 3 ist.

c) Erweitere das Programm, sodass die Benutzer die Möglichkeit haben, eine Auswahl für eine einfache mathematische Operation (Addition, Subtraktion, Multiplikation) zu treffen und die Berechnung durchzuführen.

## Aufgabe 6: Verwendung von `continue` und `break`

**Aufgabe:** Schreibe ein Java-Programm, das die Schlüsselwörter `continue` und `break` in Schleifen verwendet.

a) Erstelle ein Programm, das eine `for`-Schleife verwendet, um die Zahlen von 1 bis 10 auszugeben, aber die Zahl 5 überspringt (verwende `continue`).

b) Erweitere das Programm, sodass die Schleife vorzeitig abbricht, sobald die Zahl 8 erreicht ist (verwende `break`).

c) Modifiziere das Programm, sodass die Schleife alle geraden Zahlen zwischen 1 und 20 ausgibt, aber die Zahl 12 überspringt.

**Hinweise:**

- `continue` überspringt den aktuellen Durchlauf der Schleife und setzt mit dem nächsten fort.
- `break` beendet die Schleife vollständig.

# Aufgabe 7: Einführung in Klassen und Objekte

**Aufgabe:** Schreibe ein Java-Programm, das eine einfache Klasse mit Attributen und Methoden erstellt. Implementiere dabei eine separate Klasse zum Erstellen und Testen von Objekten.

a) Erstelle eine Klasse `Hund`, die die Attribute `name` (String) und `alter` (int) enthält. Implementiere eine Methode `bellen()`, die „Wuff!“ auf dem Bildschirm ausgibt.

b) Erstelle eine separate Klasse `Main`, die eine `main()`-Methode enthält. In dieser Klasse sollen die Objekte der Klasse `Hund` erstellt werden. Weise den Objekten einen Namen und ein Alter zu und rufe die Methode `bellen()` auf.

c) Erweitere die Klasse `Hund` um eine Methode `spielen()`, die eine Nachricht ausgibt, z.B. „[Name] spielt im Garten!“. Rufe diese Methode in der Klasse `Main` ebenfalls auf.

# Aufgabe 8: Konstruktoren in Java

**Aufgabe:** Verwende Konstruktoren, um Objekte effizient zu erstellen, und teste die Konstruktoren in einer separaten Klasse.

a) Erweitere die Klasse `Hund` um einen Konstruktor, der die Attribute `name` und `alter` als Parameter übernimmt und initialisiert.

b) Erstelle eine separate Klasse `Main`, in der du im `main()`-Block neue Objekte der Klasse `Hund` mit Hilfe des Konstruktors erstellst. Teste, ob die Attribute korrekt gesetzt wurden.

c) Füge einen weiteren Konstruktor hinzu, der nur den `name` übernimmt und das `alter` standardmäßig auf 1 setzt. Teste diesen Konstruktor, indem du ein Objekt mit nur einem Parameter in der `Main`-Klasse erstellst.

d) Verwende das Schlüsselwort `this`, um auf die Instanzvariablen im Konstruktor zuzugreifen und die Konstruktoren zu verketteln. Teste die Implementierung in der `Main`-Klasse.

—

# Aufgabe 9: Getter und Setter Methoden

**Aufgabe:** Implementiere Getter- und Setter-Methoden, um auf die Attribute eines Objekts zuzugreifen und sie zu ändern.

a) Ergänze die Klasse `Hund` um Getter-Methoden für die Attribute `name` und `alter`. Teste diese Methoden in der `Main`-Klasse, indem du sie aufrufst und die Ergebnisse ausgibst.

b) Ergänze die Klasse `Hund` um Setter-Methoden für die Attribute `name` und `alter`. Verwende die Setter-Methoden, um die Werte der Attribute in der `Main`-Klasse zu ändern, und überprüfe die geänderten Werte.

c) Erweitere das Programm so, dass der Hund nicht älter als 20 Jahre und nicht jünger als 0 Jahre sein darf. Füge in den Setter-Methoden entsprechende Bedingungen ein und gib eine Fehlermeldung aus, wenn ungültige Werte gesetzt werden sollen.

d) Setze die Attribute `name` und `alter` als `private`, um die Kapselung sicherzustellen. Verwende `public` für die Getter- und Setter-Methoden, um den Zugriff von außen zu ermöglichen.

—

# Aufgabe 10: Erstellung und Verwaltung von mehreren Objekten

**Aufgabe:** Schreibe ein Java-Programm, in dem mehrere Objekte einer selbst erstellten Klasse erzeugt und verwaltet werden. Verwende dabei Konstruktoren und Methoden, um die Daten der Objekte zu verarbeiten.

Du entwickelst eine Anwendung zur Verwaltung einer kleinen Autosammlung.

a) Erstelle eine Klasse `Auto` mit den Attributen `marke` (String), `modell` (String), `baujahr` (int) und `kilometerstand` (int). Erstelle einen Konstruktor, der alle vier Attribute initialisiert. Füge eine Methode `druckeInfo()` hinzu, die alle Informationen des Autos auf dem Bildschirm ausgibt (Marke, Modell, Baujahr, Kilometerstand).

b) Erstelle in der separaten Klasse `Main` mindestens drei Objekte der Klasse `Auto` mit verschiedenen Marken, Modellen, Baujahren und Kilometerständen. Rufe für jedes Objekt die Methode `druckeInfo()` auf, um die Informationen der Autos auszugeben.

c) Ergänze die Klasse `Auto` um eine Methode `istOldtimer()`, die überprüft, ob das Auto älter als 30 Jahre ist. Gib eine entsprechende Nachricht zurück (z.B. „Das Auto `[Marke] [Modell]` ist ein Oldtimer“). Verwende diese Methode im `Main`-Block für jedes Auto und gib an, welche Autos Oldtimer sind.

d) Verwende Zugriffsmodifizierer (`private` für die Attribute und `public` für die Methoden), um die Sichtbarkeit der Attribute zu steuern und sicherzustellen, dass der Zugriff nur über Getter- und Setter-Methoden erfolgt.

# Aufgabe 11: Arbeiten mit Listen und Objekten

**Aufgabe:** Schreibe ein Java-Programm, in dem eine Liste von Objekten einer selbst erstellten Klasse verwaltet wird. Übe dabei das Erstellen, Hinzufügen, Entfernen und Auslesen von Objekten in einer Liste.

Du entwickelst eine Anwendung zur Verwaltung einer Sammlung von Büchern in einer Bibliothek.

a) Erstelle eine Klasse `Buch` mit den Attributen `titel` (String), `autor` (String), `isbn` (String) und `seitenzahl` (int). Erstelle einen Konstruktor, der alle Attribute initialisiert. Füge eine Methode `druckeInfo()` hinzu, die alle Informationen über das Buch auf dem Bildschirm ausgibt (Titel, Autor, ISBN, Seitenzahl).

b) Erstelle in einer separaten Klasse `Bibliothek` eine Liste vom Typ `Buch` (verwende `ArrayList<Buch>`). Füge mindestens drei verschiedene Bücher zur Liste hinzu, indem du die Methode `add()` der Liste verwendest.

c) Gib die Informationen zu allen Büchern in der Liste aus, indem du eine `for`-Schleife oder eine `for-each`-Schleife verwendest, die für jedes Buch in der Liste die Methode `druckeInfo()` aufruft.

d) Erweitere das Programm, sodass ein Buch aus der Liste entfernt werden kann (z.B. basierend auf der ISBN). Verwende die Methode `remove()` der Liste, um ein Buch zu entfernen.

e) Füge eine Methode `sucheBuch(String isbn)` zur Klasse `Main` hinzu, die ein Buch in der Liste anhand seiner ISBN sucht und die Informationen des gefundenen Buches ausgibt.

## Aufgabe 12: Interaktive Steuerung von Flüssigkeitsbehältern

Erstelle ein interaktives Programm, das zwei Flüssigkeitsbehälter steuert. Die Behälter heißen „first“ und „second“. Beide können jeweils bis zu 100 Liter Flüssigkeit fassen.

Das Programm bietet die Möglichkeit, Flüssigkeit hinzuzufügen, zu verschieben und zu entfernen. Die Befehle sind wie folgt definiert:

- `add amount`: Fügt dem ersten Behälter die angegebene Menge an Flüssigkeit hinzu. Die angegebene Menge muss als Ganzzahl (int) angegeben werden. Ein Behälter kann jedoch nicht mehr als 100 Liter Flüssigkeit aufnehmen. Alles, was darüber hinausgeht, geht verloren.

- `move amount`: Verschiebt die angegebene Menge Flüssigkeit vom ersten Behälter in den zweiten Behälter. Die Menge muss als Ganzzahl angegeben werden. Wenn mehr Flüssigkeit verschoben werden soll, als der erste Behälter enthält, wird nur die vorhandene Flüssigkeit verschoben. Der zweite Behälter kann nicht mehr als 100 Liter Flüssigkeit enthalten. Überschüssige Flüssigkeit geht verloren.

- `remove amount`: Entfernt die angegebene Menge Flüssigkeit aus dem zweiten Behälter. Wenn mehr Flüssigkeit entfernt werden soll, als im Behälter enthalten ist, wird die gesamte Flüssigkeit entfernt.

Nach jedem Befehl gibt das Programm den aktuellen Inhalt beider Behälter aus.

**Hinweis:** Um eine Zeichenkette zu zerlegen und den Befehl sowie die Menge zu extrahieren, kannst du folgenden Code verwenden:

```
String input = scan.nextLine();
String[] parts = input.split(" ");

String command = parts[0];
int amount = Integer.valueOf(parts[1]);
```

Ein „quit“-Befehl beendet das Programm.

# Bonusaufgaben

In den folgenden Bonusaufgaben sollen die zu erstellenden Klassen immer in einer separaten Klasse `Main` verwendet werden. Die Klasse `Main` enthält die `main()`-Methode, in der die Objekte der jeweiligen Klasse instanziiert und die Methoden der Objekte getestet werden.

## Bonusaufgabe 1: Summe von Objekteigenschaften

Erstelle eine Klasse `Produkt` mit den Attributen `name` (String) und `preis` (double). Lasse den Benutzer drei Produkte erstellen und deren Preise eingeben. Berechne die Summe der Preise und gib sie aus.

## Bonusaufgabe 2: Durchschnittswerte von Objekten

Erstelle eine Klasse `Student` mit den Attributen `name` (String) und `note` (double). Lasse den Benutzer mehrere `Student`-Objekte erstellen und gib den Durchschnitt der Noten aus.

## Bonusaufgabe 3: Umgekehrte Objektnamen

Erstelle eine Klasse `Person` mit dem Attribut `name` (String). Lasse den Benutzer eine `Person` erstellen und den Namen eingeben. Gib den Namen umgekehrt aus.

## Bonusaufgabe 4: Buchstabenhäufigkeit in Objektnamen

Erstelle eine Klasse `Film` mit dem Attribut `titel` (String). Lasse den Benutzer den Titel eingeben und zähle, wie oft ein bestimmter Buchstabe im Titel vorkommt. Der Buchstabe wird vom Benutzer ausgewählt.

## Bonusaufgabe 5: Zahlen in Worten (Objekte) umwandeln

Erstelle eine Klasse `ZahlInWorten` mit einer Methode `zahlInWort(int zahl)`, die eine Zahl in Worten ausgibt (z.B. 1 = ëins"). Lasse den Benutzer eine Zahl eingeben und nutze ein `ZahlInWorten`-Objekt, um die Zahl als Wort auszugeben.

## Bonusaufgabe 6: Primzahlen in Objekten überprüfen

Erstelle eine Klasse `Zahl` mit einem Attribut `wert` (int). Implementiere eine Methode `istPrim()`, die prüft, ob die Zahl eine Primzahl ist. Lasse den Benutzer ein `Zahl`-Objekt erstellen und prüfe, ob der Wert prim ist.

## Bonusaufgabe 7: Kleinste und größte Zahl von Objekten finden

Erstelle eine Klasse `Messwert` mit dem Attribut `wert` (int). Lasse den Benutzer fünf `Messwert`-Objekte erstellen und finde die kleinste und größte Zahl.

## Bonusaufgabe 8: Faktorielle Berechnung von Objekteigenschaften

Erstelle eine Klasse `Faktorial` mit einem Attribut `zahl` (int). Implementiere eine Methode `berechneFakultaet()`, die die Fakultät der Zahl berechnet und ausgibt.

## Bonusaufgabe 9: FizzBuzz mit Objekten

Erstelle eine Klasse `FizzBuzz` mit einer Methode `spiel()`, die für die Zahlen von 1 bis 100 nach folgenden Regeln arbeitet:

- Gib „Fizz“ aus, wenn die Zahl durch 3 teilbar ist.
- Gib „Buzz“ aus, wenn die Zahl durch 5 teilbar ist.
- Gib „FizzBuzz“ aus, wenn die Zahl sowohl durch 3 als auch durch 5 teilbar ist.
- Andernfalls gib die Zahl selbst aus.

Lasse den Benutzer ein `FizzBuzz`-Objekt erstellen und das Spiel starten.

## Bonusaufgabe 10: Multiplikationstabellen mit Objekten

Erstelle eine Klasse `Multiplikationstabelle` mit einem Attribut `zahl` (int). Implementiere eine Methode `druckeTabelle()`, die die Multiplikationstabelle der Zahl erstellt und ausgibt. Lasse den Benutzer eine Zahl eingeben und die Tabelle drucken.

## Bonusaufgabe 11: Zeichenfrequenz in Objekten analysieren

Erstelle eine Klasse `TextAnalyse` mit einem Attribut `text` (String). Implementiere eine Methode `analyse()`, die die Häufigkeit jedes Buchstabens im Text zählt. Lasse den Benutzer einen Text eingeben und führe die Analyse durch.

## Bonusaufgabe 12: Schreiben und Lesen von Objekten in Dateien

Erstelle eine Klasse `Person` mit den Attributen `name` (String) und `alter` (int). Lasse den Benutzer ein `Person`-Objekt erstellen und schreibe die Informationen in eine Datei. Lese die Datei anschließend aus und gib den Inhalt aus.

## Bonusaufgabe 13: Wörter in einer Datei zählen

Erstelle eine Klasse `TextDatei` mit einem Attribut `dateiPfad` (String). Implementiere eine Methode `zaehleWoerter()`, die die Anzahl der Wörter in der Datei zählt. Lasse den Benutzer den Pfad eingeben und zähle die Wörter.

## Bonusaufgabe 14: Linienweise Schreiben in eine Datei

Erstelle eine Klasse `Tagebuch`, die es ermöglicht, täglich eine neue Nachricht in eine Datei zu schreiben. Der Benutzer kann mehrfach hintereinander Nachrichten eingeben, die zeilenweise in der Datei gespeichert werden.

## Bonusaufgabe 15: Suche in einer Datei

Erstelle eine Klasse `LogSuche` mit einem Attribut `dateiPfad`. Implementiere eine Methode `sucheInDatei(String wort)`, die eine Datei durchsucht und die Zeilen ausgibt, die das gesuchte Wort enthalten.

## Bonusaufgabe 16: Mehrere Objekte in einer Datei speichern

Erstelle eine Klasse `Produkt` mit den Attributen `name` (String) und `preis` (double). Lasse den Benutzer mehrere `Produkt`-Objekte erstellen und speichere diese in einer Datei. Lese die Datei anschließend aus und gib die Informationen aus.

## Bonusaufgabe 17: Einfache CSV-Verarbeitung

Erstelle eine Klasse `CSVReader`, die eine CSV-Datei einliest und die Daten als Objekte speichert. Lasse den Benutzer die Datei auswählen und verarbeite die Inhalte, indem du die CSV-Zeilen in Objekte umwandelst und die Werte ausgibst.

## Bonusaufgabe 18: Temperaturkonverter mit Objekten

Erstelle eine Klasse `Temperatur` mit den Attributen `gradCelsius` und `gradFahrenheit` . Implementiere Methoden zur Umrechnung von Celsius in Fahrenheit und umgekehrt. Lasse den Benutzer Temperaturen eingeben und umrechnen.

## Bonusaufgabe 19: Bankkonten verwalten

Erstelle eine Klasse `Konto` mit den Attributen `kontonummer`, `inhaber` (String) und `guthaben` (double). Lasse den Benutzer mehrere Konten erstellen, Geld einzahlen und abheben. Gib das aktuelle Guthaben aus.

## Bonusaufgabe 20: Einkaufswagen mit Objekten

Erstelle eine Klasse `Produkt` und eine Klasse `Einkaufswagen`. Der `Einkaufswagen` soll mehrere `Produkt`-Objekte enthalten. Lasse den Benutzer Produkte in den Einkaufswagen legen und die Gesamtkosten berechnen.