Aufgabenblatt 4 - Objektorientierte Programmierung in Java

Vorpraktikum Informatik 2024

Letztes Update des Aufgabenblattes: 11. September 2024

Einleitung

Willkommen zum Aufgabenblatt zur Objektorientierten Programmierung in Java. In diesem Blatt findest du Aufgaben, die dir helfen, die Konzepte von Vererbung, abstrakten Klassen, Polymorphismus und Interfaces zu verstehen.

Aufgabe 1: Vererbung

Aufgabe: Implementiere Vererbung in einer Klassenstruktur und teste sie in einer separaten Klasse `Main`.

a) Erstelle eine Klasse `Fahrzeug` mit den Attributen `marke` und `geschwindigkeit` . Implementiere eine Methode `beschleunigen(int)`, die die Geschwindigkeit erhöht.

b) Erstelle eine abgeleitete Klasse `Auto`, die von `Fahrzeug` erbt und das Attribut `anzahlTueren` hinzufügt. Ergänze eine Methode `druckeInfo()`, die alle Informationen des Autos ausgibt.

c) Erstelle in der Klasse `Main` ein Objekt der Klasse `Auto`, beschleunige es und rufe die Methode `druckeInfo()` auf.

Aufgabe 2: Vererbung und Überschreiben

Aufgabe: Überschreibe Methoden in abgeleiteten Klassen.

a) Erstelle eine Klasse `Tier` mit einer Methode `lautGeben()`, die „Tier macht ein Geräusch“ ausgibt.

b) Erstelle eine Klasse `Hund`, die von `Tier` erbt, und überschreibe die Methode `lautGeben()` so, dass „Wuff!“ ausgegeben wird.

c) Erstelle in der Klasse `Main` ein `Tier`-Objekt und ein `Hund`-Objekt. Rufe `lautGeben()` auf beiden Objekten auf und überprüfe, wie die Methode je nach Objekttyp ausgeführt wird.

Aufgabe 3: Abstrakte Klassen

Aufgabe: Verwende abstrakte Klassen, um die gemeinsame Struktur mehrerer Klassen zu definieren.

a) Erstelle eine abstrakte Klasse `Form` mit einer abstrakten Methode `berechneFlaeche()` und einer konkreten Methode `druckeForm()`, die den Namen der Form ausgibt.

b) Erstelle zwei abgeleitete Klassen `Kreis` und `Rechteck`, die die Methode `berechneFlaeche()` implementieren.

c) Erstelle in der Klasse `Main` Objekte von `Kreis` und `Rechteck`, rufe die Methode `berechneFlaeche()` auf beiden Objekten auf und gib die Ergebnisse aus.

Aufgabe 4: Abstrakte Klassen und konkrete Methoden

Aufgabe: Verwende abstrakte Klassen, die sowohl abstrakte als auch konkrete Methoden enthalten.

a) Erstelle eine abstrakte Klasse `Musikinstrument` mit einer konkreten Methode `stimmen()`, die „Das Instrument wird gestimmt“ ausgibt, und einer abstrakten Methode `spielen()`.

b) Erstelle zwei abgeleitete Klassen `Gitarre` und `Klavier`, die die Methode `spielen()` implementieren. Gib in `spielen()` jeweils „Die Gitarre wird gespielt“ bzw. „Das Klavier wird gespielt“ aus. Die Methode `stimmen()` wird nicht überschrieben und verwendet die generische Nachricht der Oberklasse.

c) Erstelle in der Klasse `Main` Objekte von `Gitarre` und `Klavier`. Rufe auf beiden Objekten die Methoden `spielen()` und `stimmen()` auf.

Aufgabe 5: Polymorphismus

Aufgabe: Implementiere Polymorphismus und teste es anhand einer gemeinsamen Schnittstelle.

a) Erstelle eine Klasse `Mitarbeiter` mit einer Methode `arbeite()`, die „Mitarbeiter arbeitet“ ausgibt.

b) Erstelle zwei abgeleitete Klassen `Entwickler` und `Designer`, die die Methode `arbeite()` überschreiben und jeweils spezifische Ausgaben machen (z.B. „Entwickler programmiert“).

c) Erstelle in der Klasse `Main` eine Liste von `Mitarbeiter`-Objekten (z.B. eine `ArrayList<Mitarbeiter>`), füge `Entwickler` und `Designer` hinzu, und rufe die Methode `arbeite()` auf jedem Objekt in der Liste auf.

Aufgabe 6: Interfaces

Aufgabe: Implementiere und verwende Interfaces in einem Java-Programm.

a) Erstelle ein Interface `FahrzeugInterface` mit den Methoden `start()` und `stop()`.

b) Implementiere das Interface in den Klassen `Auto` und `Motorrad`, die beide die Methoden `start()` und `stop()` spezifisch umsetzen (z.B. „Auto startet“).

c) Erstelle in der Klasse `Main` Objekte von `Auto` und `Motorrad` und verwende die Methoden des Interfaces.

Aufgabe 7: Kombination von Interfaces und Polymorphismus

Aufgabe: Verwende Interfaces, um Polymorphismus zu demonstrieren.

a) Erstelle ein Interface `Geraet` mit den Methoden `einschalten()` und `ausschalten()`.

b) Implementiere das Interface in den Klassen `Fernseher` und `Radio`, die die Methoden spezifisch umsetzen (z.B. „Fernseher eingeschaltet“).

c) Erstelle in der Klasse `Main` eine Liste von `Geraet`-Objekten und rufe `einschalten()` auf jedem Objekt in der Liste auf.

Aufgabe 8: Benutzerabfragen und mathematische Berechnungen

Aufgabe: Lasse den Benutzer mathematische Operationen über ein Interface eingeben und verarbeite die Eingaben.

a) Erstelle ein Interface `Rechenoperation`, das eine Methode `berechne(int a, int b)` definiert.

b) Implementiere zwei Klassen `Addition` und `Multiplikation`, die das Interface umsetzen und die Methode `berechne()` jeweils für Addition und Multiplikation implementieren.

c) Lasse den Benutzer in der Klasse `Main` zwei Zahlen eingeben und auswählen, ob er addieren oder multiplizieren möchte. Gib das Ergebnis der Berechnung aus.

Aufgabe 9: Temperaturkonverter mit Benutzerabfragen

Aufgabe: Implementiere ein System, das Temperaturwerte zwischen Celsius und Fahrenheit konvertiert, basierend auf Benutzereingaben.

a) Erstelle eine abstrakte Klasse `Temperatur` mit einer Methode `konvertiere()`. Definiere zwei abgeleitete Klassen `Celsius` und `Fahrenheit`, die die Methode für die jeweilige Umrechnung implementieren.

b) Implementiere eine Methode, die es dem Benutzer erlaubt, einen Temperaturwert einzugeben und auszuwählen, ob dieser Wert in Celsius oder Fahrenheit konvertiert werden soll.

c) Lasse den Benutzer in der Klasse `Main` Temperaturwerte eingeben und die Konvertierung basierend auf seiner Wahl durchführen. Gib das Ergebnis aus.

Aufgabe 10: Fahrzeugverwaltung mit Vererbung und Polymorphismus

Aufgabe: Erstelle ein System zur Verwaltung einer Fahrzeugflotte mit Vererbung und Polymorphismus.

a) Erstelle eine abstrakte Klasse `Fahrzeug` mit den Attributen `marke` (String), `baujahr` (int) und `kilometerstand` (int). Definiere eine abstrakte Methode `fahre(int km)`, die den Kilometerstand um die übergebene Kilometerzahl erhöht. Implementiere eine konkrete Methode `druckeInfo()`, die die Fahrzeugdaten ausgibt.

b) Erstelle zwei abgeleitete Klassen `Auto` und `Motorrad`, die die Methode `fahre()` implementieren. Beim Auto werden 5 zusätzliche Kilometer pro Fahrt für Nebenstrecken hinzugefügt.

c) Erstelle in der Klasse `Main` eine Liste von `Fahrzeug`-Objekten (Autos und Motorräder) und rufe die Methode `fahre()` auf jedem Objekt in der Liste auf. Teste die Funktionalität mit verschiedenen Entfernungen.

Aufgabe 11: Nutzerverwaltung mit Dateien und Vererbung

Aufgabe: Implementiere ein System zur Verwaltung von Nutzern, wobei die Daten in eine Datei geschrieben und gelesen werden.

a) Erstelle eine Klasse `Nutzer` mit den Attributen `name` (String) und `alter` (int). Implementiere eine Methode `druckeInfo()`, die den Namen und das Alter ausgibt.

b) Erstelle eine abgeleitete Klasse `Admin`, die von `Nutzer` erbt und das zusätzliche Attribut `zugriffsrechte` (String) enthält. Implementiere eine Methode `druckeInfo()`, die alle Informationen inklusive der Zugriffsrechte ausgibt.

c) Lasse den Benutzer in der Klasse `Main` mehrere Nutzer und Admins erstellen und schreibe ihre Informationen in eine Datei. Implementiere auch die Möglichkeit, die Daten aus der Datei zu lesen und auszugeben.

Aufgabe 12: Buchverwaltung mit Interfaces und Dateioperationen

Aufgabe: Implementiere ein Interface zur Verwaltung einer Sammlung von Büchern und speichere die Daten in einer Datei.

a) Erstelle ein Interface `BuchVerwaltung` mit den Methoden `hinzufuegen(Buch)` und `alleBuecherAnzeigen()`.

b) Implementiere das Interface in einer Klasse `Bibliothek`, die eine Liste von `Buch`-Objekten verwaltet. Jedes `Buch` hat die Attribute `titel`, `autor` und `isbn`.

c) Lasse den Benutzer in der Klasse `Main` Bücher hinzufügen und die Liste der Bücher in eine Datei schreiben. Implementiere auch die Möglichkeit, die Liste aus der Datei zu lesen und anzuzeigen.

Bonusaufgaben

Bonusaufgabe 1: Sortieren von Objekten

Aufgabe: Implementiere eine Klasse zur Verwaltung von Objekten und sortiere sie nach bestimmten Kriterien.

a) Erstelle eine Klasse `Person` mit den Attributen `name` (String) und `alter` (int).

b) Implementiere eine Methode `sortiereNachAlter()`, die eine Liste von `Person`-Objekten nach Alter sortiert.

c) Lasse den Benutzer mehrere Personen hinzufügen und sortiere die Liste der Personen nach ihrem Alter.

Bonusaufgabe 2: Berechnung von Primzahlen mit Objekten

Aufgabe: Entwickle ein Programm zur Berechnung von Primzahlen und verwalte die Ergebnisse in Objekten.

a) Erstelle eine Klasse `Primzahl`, die eine Methode `berechne(int n)` enthält, um die ersten `n` Primzahlen zu berechnen.

b) Speichere die berechneten Primzahlen in einer Liste und implementiere eine Methode `zeigeAllePrimzahlen()`, um die Liste auszugeben.

c) Lasse den Benutzer die Anzahl der gewünschten Primzahlen eingeben und zeige die berechneten Primzahlen an.

Bonusaufgabe 3: Rekursive Berechnungen mit Objekten

Aufgabe: Implementiere ein rekursives Programm zur Berechnung des Faktorials und verpacke die Logik in einem objektorientierten Ansatz.

a) Erstelle eine Klasse `Faktorial`, die eine Methode `berechne(int n)` implementiert, die das Faktorial von `n` rekursiv berechnet.

b) Füge eine Methode `zeigeErgebnis()` hinzu, die das berechnete Faktorial anzeigt.

c) Lasse den Benutzer eine Zahl eingeben und zeige das Faktorial der Zahl an.

Bonusaufgabe 4: Simulation eines einfachen Warenkorbs

Aufgabe: Entwickle ein Programm, das den Warenkorb eines Online-Shops simuliert.

a) Erstelle eine Klasse `Produkt` mit den Attributen `name` (String) und `preis` (double).

b) Erstelle eine Klasse `Warenkorb`, die eine Liste von `Produkt`-Objekten enthält. Implementiere Methoden zum Hinzufügen und Entfernen von Produkten.

c) Lasse den Benutzer Produkte zum Warenkorb hinzufügen, entfernen und den Gesamtpreis des Warenkorbs berechnen.

Bonusaufgabe 5: Verwaltung einer Bücherbibliothek

Aufgabe: Entwickle eine Bibliothek zur Verwaltung von Büchern mit Such- und Sortierfunktionen.

a) Erstelle eine Klasse `Buch` mit den Attributen `titel` (String), `autor` (String) und `jahr` (int).

b) Erstelle eine Klasse `Bibliothek`, die eine Liste von `Buch`-Objekten enthält. Implementiere Methoden, um Bücher hinzuzufügen, zu suchen und nach Erscheinungsjahr zu sortieren.

c) Lasse den Benutzer die Bibliothek verwalten, Bücher suchen und die Liste der Bücher sortieren.

Bonusaufgabe 6: Tic-Tac-Toe Spiel mit OOP

Aufgabe: Entwickle ein Tic-Tac-Toe Spiel mit objektorientierter Programmierung.

a) Erstelle eine Klasse `Spieler` mit einem Attribut `name` (String), das den Namen des Spielers speichert.

b) Erstelle eine Klasse `TicTacToe`, die das Spielfeld verwaltet und Methoden zum Setzen von Zügen und Überprüfen des Gewinners implementiert.

c) Lasse zwei Spieler gegeneinander spielen und prüfe nach jedem Zug, ob ein Spieler gewonnen hat.

Bonusaufgabe 7: Graphenimplementierung mit Objekten

Aufgabe: Entwickle ein Programm zur Verwaltung eines ungerichteten Graphen mit Knoten und Kanten.

a) Erstelle eine Klasse `Knoten` mit einem Attribut `name` (String), das den Namen des Knotens speichert.

b) Erstelle eine Klasse `Graph`, die eine Liste von Knoten enthält und eine Methode `fuegeKanteHinzu(Knoten k1, Knoten k2)` implementiert, um zwei Knoten zu verbinden.

c) Lasse den Benutzer Knoten und Kanten hinzufügen und zeige den Graphen als Liste der Verbindungen an.

Bonusaufgabe 8: Brute-Force Passwortprüfung

Aufgabe: Entwickle ein Programm zur Brute-Force-Prüfung eines einfachen Passworts.

a) Erstelle eine Klasse `PasswortChecker`, die eine Methode `pruefePasswort(String passwort)` implementiert, um ein vorgegebenes Passwort zu testen.

b) Implementiere eine Methode `bruteForce(String passwort)`, die alle möglichen Kombinationen aus Buchstaben testet, um das Passwort zu finden.

c) Lasse den Benutzer ein Passwort eingeben und versuche, es mit der Brute-Force-Methode zu knacken.

Bonusaufgabe 9: Dateianalyse mit OOP

Aufgabe: Entwickle ein Programm, das eine Datei analysiert und Statistiken zu Wort- und Zeichenhäufigkeit liefert.

a) Erstelle eine Klasse `DateiAnalysator`, die eine Methode `analysiereDatei (String dateipfad)` implementiert, um eine Datei einzulesen.

b) Implementiere Methoden, um die Anzahl der Wörter und die Häufigkeit jedes Buchstabens in der Datei zu zählen.

c) Lasse den Benutzer eine Datei auswählen und zeige die Analyseergebnisse an.

Bonusaufgabe 10: Komplexe Zahlen mit OOP

Aufgabe: Entwickle ein Programm, das mit komplexen Zahlen arbeitet.

a) Erstelle eine Klasse `KomplexeZahl` mit den Attributen `realteil` (double) und `imaginaerteil` (double).

b) Implementiere Methoden zur Addition, Subtraktion und Multiplikation von komplexen Zahlen.

c) Lasse den Benutzer zwei komplexe Zahlen eingeben und die Ergebnisse der Operationen anzeigen.