Einführung in die Objekt Orientierung Programmierung OOP

Was ist OOP?

Objektorientierte Programmierung (OOP) ist ein Programmierparadigma, das auf **Objekten** basiert, die **Daten** (Attribute) und **Funktionen** (Methoden) kombinieren. Es hilft, komplexe Systeme durch das Modellieren von realen Entitäten als Objekte zu strukturieren. Die vier Hauptprinzipien von OOP sind:

1. **Kapselung**: Daten und Funktionen werden in Objekten zusammengefasst, um den Zugriff zu kontrollieren.
2. **Vererbung**: Neue Klassen können von bestehenden Klassen erben und deren Eigenschaften/Funktionen wiederverwenden.
3. **Polymorphismus**: Objekte können verschiedene Formen annehmen, was flexible und dynamische Codeanpassungen ermöglicht.
4. **Abstraktion**: Nur wesentliche Informationen werden gezeigt, Details werden verborgen, um die Komplexität zu reduzieren.

OOP erleichtert die **Wiederverwendbarkeit**, **Wartbarkeit** und **Erweiterbarkeit** des Codes.

Vorgangsweise

Es gibt 23 Code-Versionen, von denen jede ein spezifisches Konzept erklärt. Dabei wird durch PDF-Datei erläutert, warum dieses Konzept wichtig ist und wie man es umsetzen kann.

Wichtige Hinweise

1. **Ziel des Tutorials**: Das Ziel dieses Tutorials ist es, den Teilnehmern nicht nur zu zeigen, wie das Konzept umgesetzt wird, sondern auch zu erklären, warum dieses Konzept wichtig ist, wann man es anwenden sollte, welche Vorteile und Merkmale es hat und wie man mögliche Nachteile vermeiden kann.

2. **Zielgruppe**: Dieses Tutorial richtet sich speziell an Dart-Entwickler, die bereits grundlegende Dart-Kenntnisse besitzen, wie das Schreiben einfacher Funktionen. Das allgemeine Konzept ist jedoch auch für Entwickler anderer Programmiersprachen verständlich.

3. **Schrittweise Vorgehensweise**: Dieses Tutorial richtet sich auch an Entwickler, die noch wenig Erfahrung mit OOP haben. Wenn Sie sich mit den ersten Versionen unterfordert fühlst, können Sie zur nächsten Version übergehen. Ich verspreche Ihnen, dass Sie wertvolle Informationen finden werden.

4. **Referenz für Code**: Dieses Tutorial kann auch als Code-Referenz dienen, zum Beispiel wenn Sie sich daran erinnern möchten, wie man einen Operator überlädt.

5. **Aufbau des Tutorials**: Das Tutorial besteht aus zwei Teilen: Code und PDF-Erklärung. Jede Version enthält eine PDF-Datei, die das jeweilige Konzept des Codes theoretisch erklärt und den Code im Detail verdeutlicht. Am Ende gibt es eine Aufgabe, die das Gelernte festigt.

6. **Lernaufgaben**: Jede Version enthält eine "Lernaufgaben"-Datei, in der der Teilnehmer Aufgaben findet, die in der PDF-Datei behandelt werden. Egal, wie gut man ein Konzept theoretisch versteht, man kann es nicht richtig anwenden und auswendig lernen, ohne praktische Erfahrung zu sammeln. Daher sind die Aufgaben sehr wichtig.

7. **Empfehlung zum Üben**: Es wird dringend empfohlen, mit dem Code zu experimentieren und zu testen. Nur so kann man wirklich lernen.

Versionenziele

V1: Verständnis des Klassenaufbaus, Definition und Verwendung von Attributen und Methoden.

V2: Grundlagen eines einfachen Konstruktors mit optionalen und unbenannten Parametern.

V3: Verständnis und Anwendung des `this`-Schlüsselworts und benannten Parametern.

V4: Einführung in Kapselung, private Attribute, Setter und Getter, sowie deren Verwendung.

V5: Verständnis der Aggregationsbeziehung zwischen Klassen.

V6: Verständnis der Kompositionsbeziehung zwischen Klassen.

V7: Einführung in Vererbung (Inheritance).

V8: Vererbung mit abstrakten Klassen, Eigenschaften und Nutzen abstrakter Klassen, sowie die Bedeutung und der Einsatz abstrakter Methoden.

V9: Implementierung und Unterschiede zur Vererbung, mit einem Beispiel einer Klasse, die Übersetzungsfunktionen implementiert.

V10: Verständnis der Mixins, ihrer Vor- und Nachteile und ihrer Einsatzmöglichkeiten.

V11: Einführung in Factory-Konstruktoren, deren Erstellung und Nutzen, mit Beispielen.

V12: Immutable-Klassen, deren Vorteile und Notwendigkeit der `copyWith`-Methode, sowie Aufbau und Nutzung.

V13: Objekte für das Internet nutzbar machen: JSON und warum Klassen die Fähigkeit zur JSON-Serialisierung brauchen. Grundlagen der `fromJson` und `toJson` Methoden mit einfachen Typen wie `int`, `double` und `String`.

V14: `fromJson` und `toJson` Methoden für Klassen mit Attributen, die komplexere Objekte enthalten.

V15: `fromJson` und `toJson` Methoden für Klassen mit Attributen wie einfachen Maps oder Listen.

V16: `fromJson` und `toJson` Methoden für Klassen mit Attributen, die Listen von Objekten enthalten.

V17: Operatoren überschreiben und überladen, Bedeutung des `hashCode` beim Vergleichsoperator, und Überschreiben von Methoden wie `toString`.

V18: Erweiterung eingebauter Dart-Klassen wie `List` und `int` mit zusätzlichen Methoden und Operatoren, Grenzen und Einschränkungen dabei.

V19: Einführung in Sealed-Klassen, ihre Erstellung und ihre Einsatzmöglichkeiten.

V20: Bedeutung und Nutzen von Kommentaren, Unterschiede zwischen zwei-Strich- und drei-Strich Kommentaren.

V21: Privat-Konstruktoren, ihre Nützlichkeit und Anwendungsbeispiele.

V22: Enums – Nutzen und Anwendung, die Wertigkeit und Funktionalität von Klassen-Enums, und wie man sie erstellt und verwendet.

V23: Statische Methoden und Attribute, ihre Eigenschaften, Erstellung und Anwendungsbeispiele.

Am Ende würde ich mich über Feedback und Verbesserungsvorschläge freuen.