Aufgabestellung: Projekt Lebenswesen

In diesem Projekt werden Sie eine Klassenhierarchie für Lebenswesen erstellen. Ziel ist es, Konzepte wie **Packages**, **Vererbung**, **Sichtbarkeiten**, **Getter und Setter**, sowie **Methodenüberschreibung** (Override) zu verstehen und anzuwenden.

Ziele

- **Verständnis von Sichtbarkeiten:** Verwenden Sie `public`, `protected`, `private` und `default` für unterschiedliche Zugriffsszenarien.
- **Vererbung und Polymorphismus:** Implementieren Sie eine Klassenhierarchie mit gemeinsamen und spezifischen Methoden.
- **Getter und Setter:** Kapseln Sie private Eigenschaften und ermöglichen Sie den Zugriff über Getter und Setter.
- **Packages:** Strukturieren Sie Ihr Projekt in verschiedenen Paketen.

Aufgaben

1. Erstellen Sie die Oberklasse `Lebewesen`

- Die Klasse soll gemeinsame Eigenschaften wie `name` und `alter` enthalten.
- Methoden:
  - `bewegen()` – Eine generische Methode, die später überschrieben werden kann.

---

2. Erstellen Sie die Unterklassen `Tier` und `Pflanze`

- **`Tier`:**
  - Soll eine abstrakte Klasse sein.
  - Eigenschaften:
    - `nahrung` (z. B. Fleisch, Pflanzen).
  - Methoden:
    - `fressen()` – Gibt aus, was das Tier frisst.
    - Abstrakte Methode `lautGeben()` – Muss in Unterklassen implementiert werden.
- **`Pflanze`:**
  - Überschreibt die Methode `bewegen()`, um das Wachstum anzuzeigen.

---

3. Erstellen Sie spezifische Klassen

- **`Mensch`:**
  - Eigenschaften:
    - `sprache` (z. B. Deutsch, Englisch).
  - Überschreibt `bewegen()` – Gibt aus, dass der Mensch läuft und spricht.
- **`Katze`:**

- Erbt von `Tier`.
- Implementiert die Methode `lautGeben()` – Gibt z. B. "miaut" aus.

---

4. Implementieren Sie eine Hauptklasse

- Erstellen Sie eine Klasse `Main`, in der Sie:
  - Objekte von `Pflanze`, `Mensch` und `Katze` instanziieren.
  - Methoden wie `bewegen()`, `fressen()` und spezifische Methoden aufrufen.

Erweiterung

1. Fügen Sie eine weitere Tierklasse, z. B. `Hund`, hinzu.
   - Implementieren Sie spezifische Methoden wie `lautGeben()` und `spielen()`.
2. Experimentieren Sie mit den Sichtbarkeiten:
   - Verwenden Sie `protected` für Eigenschaften, die Unterklassen benötigen.
   - Verwenden Sie `private` für Eigenschaften, die nur innerhalb der Klasse zugänglich sein sollen.
   - Nutzen Sie Getter und Setter, um den Zugriff zu kontrollieren.
3. Nutzen Sie Polymorphismus:
   - Erstellen Sie eine Liste von `Lebewesen` und rufen Sie verschiedene Methoden auf, um das Verhalten der Unterklassen zu testen.

Abgabe

- Eine funktionierende Klassenhierarchie mit allen oben genannten Anforderungen.
- Eine klar strukturierte Package-Organisation.
- Eine Hauptklasse mit vollständigem Testcode, der die Hierarchie demonstriert.