**Dokumentation für das Tierheim-Verwaltungssystem**

**1. Einleitung**

Das Tierheim-Verwaltungssystem ist ein einfaches Konsolenprogramm, das es Benutzern ermöglicht, Tiere in einem Tierheim zu verwalten. Benutzer können Tiere hinzufügen, alle Tiere anzeigen und das Programm beenden.

**2. Architektur**

**2.1. Klassenübersicht**

* **ITier (Interface)**

Definiert eine Methode ILautGeben()

* **Tier (Basisklasse)**
  + Eigenschaften:
    - Name: Der Name des Tieres.
    - Alter: Das Alter des Tieres.
  + Konstruktor:
    - Nimmt name und alter als Parameter
  + Methoden:
    - LautGeben(): Gibt einen allgemeinen Laut des Tieres aus. Diese Methode kann von abgeleiteten Klassen überschrieben werden.
* **Hund (abgeleitete Klasse)**
  + Erbt von der Klasse Tier.
  + Eigenschaften:
    - Rasse: Die Rasse des Hundes.
  + Konstruktor:
    - Nimmt name, alter und rasse als Parameter
  + Methoden:
    - Überschreibt die LautGeben()-Methode, um einen spezifischen Laut für Hunde auszugeben.

**2.2. Programmlogik**

* **Hauptprogramm (Program-Klasse)**
  + Enthält die Main()-Methode
  + Eine Liste von Tier-Objekten wird erstellt, um die Tiere im Tierheim zu speichern.
  + Eine Endlosschleife ermöglicht Benutzereingaben für:
    - Hinzufügen eines Tieres
    - Anzeigen aller Tiere
    - Beenden des Programms
    - Speichern der Tierdaten in einer JSON-Datei

**3. Benutzeranweisungen**

**3.1. Programmstart**

Beim Starten des Programms wird der Benutzer mit einer Auswahl von Optionen begrüßt:

1. Tier hinzufügen
2. Alle Tiere anzeigen
3. Programm beenden
4. Tiere speichern

**3.2. Tier hinzufügen**

* Der Benutzer wird aufgefordert, den Namen und das Alter des Tieres einzugeben.
* Danach fragt das Programm, ob das Tier ein Hund ist.
* Falls der Benutzer "ja" wählt, wird nach der Rasse des Hundes gefragt.
* Das Tier (entweder als Hund oder als Tier) wird zur Liste hinzugefügt, und eine Bestätigung wird ausgegeben.

**3.3. Alle Tiere anzeigen**

* Bei Auswahl dieser Option wird eine Liste aller Tiere im Tierheim angezeigt.
* Für jedes Tier werden der Name, das Alter und der spezifische Laut ausgegeben.

**3.4. Programm beenden**

* Der Benutzer kann das Programm jederzeit beenden, indem er die entsprechende Option wählt.

**4. Programmierung**

**4.1 Ideenfindung**

Die Idee für dieses Tierheim-Verwaltungssystem entstand durch eine Anfrage an eine KI, die mir verschiedene Ansätze vorschlug. Nach der Evaluierung dieser Konzepte entschied ich mich, die Idee einer Zoo-Verwaltung anzupassen und den Fokus auf ein Tierheim zu legen. Diese Entscheidung erlaubte es mir, die Struktur und Logik so zu gestalten, dass sie den spezifischen Anforderungen eines Tierheims gerecht wird.

* 1. **Programmierschritte**
* Planung der Klassenstruktur: Zu Beginn überlegte ich, wie ich verschiedene Tierarten und deren spezifische Eigenschaften strukturieren könnte. Hierbei entwarf ich ein Interface ILautVomTier, das sicherstellt, dass jede Tierklasse die Methode LautGeben() implementiert.
* Im nächsten Schritt entwickelte ich die Basisklasse Tier, die die grundlegenden Eigenschaften wie Name und Alter enthält. Der Konstruktor dieser Klasse ermöglicht es, diese Eigenschaften bei der Erstellung eines Objekts zu initialisieren, was zu einer klaren und konsistenten Datenstruktur führt. Die Methode LautGeben() gibt einen allgemeinen Laut des Tieres aus und ist als virtuell gekennzeichnet, sodass sie in abgeleiteten Klassen überschrieben werden kann.
* Für die spezifischen Eigenschaften von Hunden erstellte ich die abgeleitete Klasse Hund, die von Tier erbt. Ich fügte die Eigenschaft Rasse hinzu, um zusätzliche Details zu speichern, die für Hunde relevant sind. In dieser Klasse wurde die Methode LautGeben() überschrieben, um den charakteristischen Laut eines Hundes auszugeben.
* Ein wichtiger Schritt war die Implementierung der Main()-Methode, in der ich ein Menüsystem entwarf, das die Benutzer durch die verschiedenen Optionen führt. Ich nutzte eine Endlosschleife, die es den Benutzern ermöglicht, mehrere Eingaben zu tätigen, ohne das Programm neu starten zu müssen.
* Ein weiterer wichtiger Aspekt war die Benutzerinteraktion. Ich wollte sicherstellen, dass die Anwendung benutzerfreundlich ist. Zu diesem Zweck implementierte eine if-Anweisung in der Endlosschleife im Hauptprogramm, um eine neue Zeile nach erfolgreichen Eingaben zu implementieren. Hierfür benötigte ich einen Boolian.
* Ausserdem stellte ich dann fest, dass es wichtig war, die Eingabe zu überprüfen, um sicherzustellen, dass sie tatsächlich eine Zahl ist. Um dies zu verbessern, hätte ich int.TryParse() verwenden können, um Eingabefehler zu vermeiden. Stattdessen entschied ich mich, die Eingabe in int.Parse() zu belassen und die Benutzer zu instruieren, die Eingabe zu wiederholen, falls ein Fehler auftritt.
* Die Option zum Hinzufügen eines Tieres verlangt vom Benutzer die Eingabe des Namens, Alters und die Abfrage, ob das Tier ein Hund ist. Ich implementierte eine Eingabeverarbeitung, die sicherstellt, dass die Eingaben korrekt sind. Bei der Altersabfrage verwendete ich int.Parse(), um sicherzustellen, dass die Eingabe eine Ganzzahl ist. Nach der Eingabe wird ein neues Tierobjekt, entweder vom Typ Hund oder der Basisklasse Tier, erstellt und zur Liste hinzugefügt.
* Die Unterscheidung zwischen Hunden und anderen Tieren stellte eine weitere Herausforderung dar. Ich implementierte eine Abfrage, die den Benutzer fragte, ob das Tier ein Hund ist. Je nach Antwort wurde entweder ein Hund-Objekt oder ein Tier-Objekt erstellt. Um die Benutzerfreundlichkeit zu erhöhen, stellte ich sicher, dass die Eingaben in Kleinbuchstaben umgewandelt wurden, um Tippfehler zu vermeiden (mit .ToLower).
* Ich wollte die Anwendung so erweitern, dass sie auch andere Tierarten unterstützt. Dies erforderte eine gründliche Überlegung, wie ich weitere abgeleitete Klassen für andere Tiere, wie Katzen oder Vögel, implementieren könnte. Ich entschied mich zunächst, es einfach zu halten und die Implementierung auf Hunde zu beschränken, um die Basis des Projekts stabil zu halten, bevor ich weitere Tierarten hinzufüge.
* Um die Flexibilität des Systems zu erhöhen, fügte ich die Möglichkeit hinzu, die aktuellen Tierdaten zu speichern, ohne das Programm beenden zu müssen. Diese Funktion ermöglicht es den Benutzern, ihre Eingaben jederzeit zu speichern.
* Um die Benutzeroberfläche meines Tierheim-Verwaltungssystems zu verbessern, implementierte ich die Funktion zum Leeren der Konsole nach jedem Prozess. Dazu verwendete ich die Console.Clear()-Methode, die die Konsole von vorherigen Ausgaben befreit und somit eine übersichtlichere Benutzererfahrung bietet. Ich platzierte diese Methode strategisch nach jeder erfolgreichen Eingabe oder Aktion im Hauptmenü, um den Benutzern ein frisches Interface zu präsentieren.

Randnotiz: Ich habe die Programmierung mit Hilfe von Freunden und KI erstellt.

**4.3 Speichern und Laden in JSON**

Um die Funktionalität meines Tierheim-Verwaltungssystems zu erweitern, implementierte ich eine Methode zum Speichern der Tierdaten in einer JSON-Datei. Dabei nutzte ich die System.Text.Json-Bibliothek, um die Liste der Tiere in einen JSON-String zu serialisieren und in einer Datei namens tiere.json zu speichern. Nun hatte ich das Problem, dass ich die abgespeicherten Tiere nicht abrufen konnte. Hierfür habe ich am Anfang meines Programms eine Methode implementiert, die die Daten der Tiere lädt. Ich fügte eine neue Option im Hauptmenü hinzu, die es den Benutzern ermöglicht, die Daten beim Start des Programms zu laden.