Projekt «Virtuelles Haustier»

Erstelle eine WPF-Anwendung, in der ein virtuelles Haustier (z. B. ein Hund oder eine Katze) gepflegt wird. Das Haustier hat verschiedene Eigenschaften und Methoden, die gepflegt, verändert und überprüft werden können. Es reagiert auf Eingaben des Benutzers, wie Fütterung, Spazierengehen oder Schlafen, und gibt visuelle Rückmeldungen.

Ziel
Das virtuelle Haustier soll über eine grafische Oberfläche gesteuert werden, wobei objektorientierte Prinzipien wie Kapselung, Vererbung und Polymorphie verwendet werden. Gleichzeitig wird das Design dokumentiert und das Projekt auf Qualität geprüft.

1. Objektorientiertes Design

- **Kapselung**
  Das Haustier wird durch eine Klasse Haustier modelliert, die private Eigenschaften wie Name, Energie, Hunger, Stimmung und Gesundheit enthält. Diese Eigenschaften dürfen nur über öffentliche Methoden verändert werden.

- **Vererbung**
  Die Klasse Haustier wird durch spezialisierte Klassen wie Hund und Katze erweitert. Diese erben die Grundfunktionalität der Haustier-Klasse, können aber zusätzliche Methoden oder Eigenschaften haben, die für den jeweiligen Haustier-Typ spezifisch sind.

- **Polymorphie**
  Die Methode ReagiereAufAktion wird in den abgeleiteten Klassen Hund und Katze unterschiedlich implementiert, sodass je nach Haustier-Typ eine andere Reaktion auf Benutzeraktionen erfolgt.

- **Abstraktion**
  Die abstrakte Klasse Haustier definiert Methoden wie Füttern(), Spaziergang(), Schlafen(), während die konkreten Klassen Hund und Katze diese Methoden mit spezifischem Verhalten umsetzen.

2. Modellierung und Dokumentation

- **CRC-Karten**
  Jede Klasse (z. B. Haustier, Hund, Katze) erhält eine CRC-Karte (Class-Responsibility-Collaborator), um die Verantwortlichkeiten und Interaktionen der Klassen zu verstehen.

- **UML-Diagramme**
  Die Klassenstruktur und die Beziehungen zwischen den Klassen werden in einem UML-Diagramm dokumentiert.

- **Javadoc**
  Jede Methode und Klasse wird mit einer kurzen Beschreibung und Kommentaren versehen, die den Zweck und die Funktionsweise erklären.

3. Implementierung
Erstelle eine WPF-Anwendung mit folgenden Elementen:

- **Interface**
  - IHaustierAktionen
- **Klassen**
  - Haustier (basierend auf Interface und abstracts)

  - Hund und Katze (spezialisierte Klassen)

- **Eigenschaften**
  - Name
  - Energie
  - Hunger
  - Stimmung
  - Gesundheit

- **Methoden**
  - Fuettern()
  - Spaziergang()
  - Schlafen()
  - ReagiereAufAktion()

- **WPF-Elemente**
  - Buttons, um das Haustier zu füttern, spazieren zu führen oder schlafen zu legen.
  - Labels, um die Eigenschaften des Haustiers anzuzeigen (z. B. Energie, Stimmung).
  - Eine visuelle Darstellung des Haustiers (z. B. ein Bild oder ein Symbol, das sich je nach Zustand des Haustiers ändert).

4. Testen und Qualitätssicherung
Erstelle Unit-Tests für die Klassen und Methoden, um die Korrektheit und Qualität des Codes sicherzustellen:

- **Unit-Test für Fuettern():** Überprüfen, ob die Energie des Haustiers nach dem Füttern steigt.
- **Unit-Test für Spaziergang():** Überprüfen, ob die Stimmung des Haustiers nach einem Spaziergang besser wird.
- **Unit-Test für Schlafen():** Überprüfen, ob die Energie des Haustiers nach dem Schlafen wiederhergestellt wird.

Zusätzliche Bonuspunkte: Die WPF-Anwendung kann um lustige Animationen oder zufällige Ereignisse erweitert werden, wie ein Haustier, das auf einmal einen «Unfall» hat oder müde wird und schlafen möchte.