Projekt zaliczeniowy

Programowanie obiektowe
Rok akademicki 2024/2025

Autorzy:
Adam Pojda
Kamila Nowak
Aleksandra Włosińska

Klinika Weterynaryjna

Projekt przedstawia system zarządzania kliniką weterynaryjną. Głównym celem było stworzenie aplikacji umożliwiającej zarządzanie danymi lekarzy, zwierząt oraz wizyt, z zapewnieniem funkcjonalności takich jak sortowanie wizyt, dodawanie nowych danych oraz serializacja i deserializacja danych do pliku XML. System oferuje graficzny interfejs użytkownika.

Podział ról

Aleksandra Włosińska: Implementacja klas w kodzie programu.

Kamila Nowak: testowanie poprawności działania klas oraz funkcjonalności projektu.(testy jednostkowe)

Adam Pojda: Projekt i implementacja graficznego interfejsu użytkownika (GUI).

Opis klas

Klasa Osoba

- **Rola:** Klasa abstrakcyjna reprezentująca osoby w systemie. Umożliwia dziedziczenie wspólnych właściwości (imię i nazwisko) przez klasy Lekarz.
- **Modyfikatory dostępu:**
    - Właściwości Imie i Nazwisko są prywatne i tylko do odczytu, co zapewnia ich bezpieczeństwo.

- **Uzasadnienie:** Centralny punkt dziedziczenia dla klas reprezentujących osoby, co zwiększa czytelność i reużywalność kodu, w przyszłości wraz z rozwojem kodu można dodać więcej klas opierających się na niej.

Klasa Lekarz

- **Rola:** Reprezentuje lekarza weterynarii.
- **Modyfikatory dostępu:**
  - Właściwość Specjalizacja jest prywatna i tylko do odczytu, aby zapobiec nieautoryzowanym zmianom.
- **Funkcje:**
  - Metoda Opis() pozwala na czytelne przedstawienie danych lekarza.
- **Uzasadnienie:** Dzięki tej klasie możliwe jest zarządzanie listą lekarzy oraz przypisywanie ich do wizyt.

Klasa Zwierze

- **Rola:** Reprezentuje zwierzęta obsługiwane w klinice.
- **Modyfikatory dostępu:**
  - Właściwości Imie, Wiek, i Rasa są prywatne i tylko do odczytu, co zapobiega zmianom po ich inicjalizacji.
  - Wyjątek ZlyWiekException zabezpiecza przed wprowadzeniem ujemnego wieku.
- **Funkcje:**
  - Clone() umożliwia tworzenie kopii obiektu zwierzęcia.
- **Uzasadnienie:** Klasa umożliwia bezpieczne zarządzanie danymi zwierząt w systemie.

Klasa Wizyta

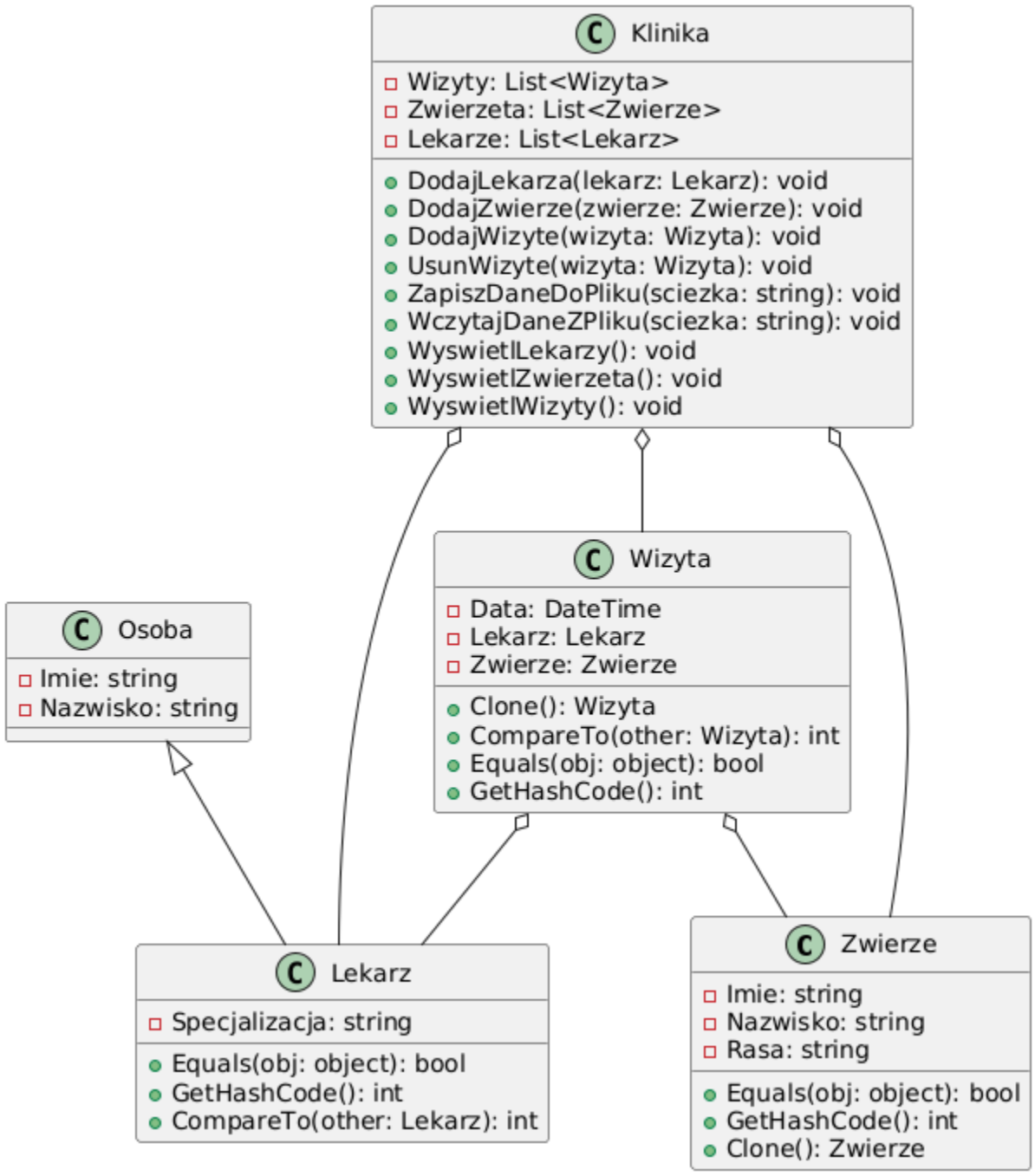
- **Rola:** Reprezentuje wizyty w klinice.
- **Modyfikatory dostępu:**
  - Właściwości Data, Lekarz, Zwierze są prywatne i tylko do odczytu, co zapewnia spójność danych.
- **Funkcje:**
  - CompareTo() umożliwia sortowanie wizyt według daty.
  - Equals() pozwala na porównywanie wizyt.
- **Uzasadnienie:** Kluczowa klasa do zarządzania harmonogramem kliniki.

Klasa Klinika

- **Rola:** Główna klasa zarządzająca danymi systemu.
- **Modyfikatory dostępu:**
  - Listy Lekarze, Zwierzeta i Wizyty są prywatne, co chroni dane przed przypadkowymi modyfikacjami.

- **Funkcje:**
  - Zarządzanie danymi (dodawanie/usuwanie wizyt, lekarzy i zwierząt).
  - Obsługa zdarzeń informujących o zmianach w systemie.
  - Serializacja i deserializacja danych.
- **Uzasadnienie:** Klasa integruje wszystkie elementy systemu, zapewniając przejrzystą strukturę i centralne zarządzanie.

Diagram klas



Opis funkcjonalności

1. **Zarządzanie danymi lekarzy:**
    - Klasa Lekarz oraz metody w Klinika umożliwiają dodawanie, wyświetlanie i zarządzanie listą lekarzy.
2. **Zarządzanie danymi zwierząt:**
    - Klasa Zwierze pozwala na dodawanie nowych zwierząt do bazy danych kliniki.
3. **Obsługa wizyt:**
    - Klasa Wizyta oraz funkcje zarządzania w Klinika umożliwiają tworzenie, usuwanie i sortowanie wizyt.
4. **Serializacja danych:**
    - Funkcja ZapiszDaneDoPliku pozwala na zapisanie stanu kliniki do pliku XML, a WczytajDaneZPliku umożliwia ich odczyt.
5. **Zdarzenia:**
    - Obsługa zdarzeń takich jak dodanie lekarza czy wizyty informuje użytkownika o zmianach w systemie.
6. **Testowanie aplikacji:**

Testowanie aplikacji było jednym z kluczowych etapów projektu, mającym na celu weryfikację poprawności działania zaimplementowanych funkcji. Testy jednostkowe zostały stworzone przy użyciu frameworka xUnit w języku C#.
Przykłady testów:

- **Klasa Zwierze:**
    - Poprawne tworzenie obiektów (np. psa rasy Labrador o imieniu „Burek” w wieku 3 lat).
    - Obsługa wyjątków – sprawdzono, czy próba przypisania wieku ujemnego rzuca odpowiedni wyjątek ZlyWiekException.
- **Klasa Lekarz:**
    - Sprawdzono poprawność formatowania danych lekarza przy pomocy metody ToString().
- **Klasa Osoba:**
    - Przetestowano poprawne przypisanie imienia i nazwiska do obiektu osoby.

7. **Interfejs graficzny:**

Graficzny interfejs użytkownika został zaprojektowany z myślą o intuicyjności obsługi. Menu główne zawiera opcje zarządzania wizytami, lekarzami i zwierzętami oraz możliwość wyjścia z aplikacji. Ekran zarządzania wizytami umożliwia przeglądanie, dodawanie, usuwanie oraz sortowanie wizyt.
Przykłady widoków GUI:

- Menu główne aplikacji:

- Zarządzanie wizytami: