

Schronisko

Magdalena Baran, Maja Panasiewicz

1 Podział pracy

1. Magdalena Baran

Opracowanie podstawowej logiki działania programu, zorganizowanej w klasy z wykorzystaniem konstruktorów oraz dziedziczenia. Zaimplementowanie metod specjalnych, obsługa wyjątków, a także mechanizmy odczytu i zapisu danych do pliku.

2. Maja Panasiewicz

Stworzenie czytelnego i przyjaznego dla użytkownika interfejsu graficznego. Dodatkowo dodane filtrowanie danych według gatunku. Wprowadzenie drobnych poprawek istniejących już metod, tak aby współgrały z interfejsem.

Sprawozdanie zostało przygotowane wspólnie, w równym podziale (50/50) przez obie osoby.

2 Opis projektu

Schronisko to aplikacja służąca do zarządzania bazą zwierząt przebywających w danym ośrodku. Program umożliwia dodawanie nowych zwierząt do systemu, przeglądanie ich danych, edytowanie informacji, a także usuwanie zwierząt z bazy. Możliwe jest również wczytywanie danych z pliku oraz ich zapis w postaci listy.

Celem projektu jest ułatwienie pracownikom schroniska przechowywania, wyszukiwania oraz filtrowania danych. Program jest przyjazny dla użytkownika dzięki prostemu i intuicyjnemu interfejsowi graficznemu. Zapobiega też potencjalnym pomyłkom, prosząc o zapisywanie pliku przy np. zamykaniu okna aplikacji.

3 Struktura Programu

3.1 Klasa bazowa - Animal

```
1 class Animal:
2     def __init__(self, name, age, species):
3         self.name = name
4         self.age = age
5         self.species = species
6
7     def __str__(self):
8         return f"{self.name} ({self.species}, {self.age} lat)"
9
10    def __eq__(self, other):
11        return isinstance(other, Animal) and self.name == other.name and
            self.species == other.species
```

Klasa Animal jest klasą podstawową zawierająca atrybuty takie jak:

- **name** - imię zwierzęcia
- **age** - wiek zwierzęcia
- **species** - gatunek zwierzęcia
- **__init__(self, name, age, species)** — konstruktor klasy, który tworzy nowy obiekt Animal i przypisuje podane wartości do atrybutów.
- **__str__(self)** — metoda specjalna zwracająca czytelny opis obiektu jako łańcuch znaków, np. "Burek (pies, 5 lat)"
- **__eq__(self, other)** — metoda porównująca dwa obiekty Animal. Zwraca True, jeśli other jest również instancją klasy Animal i ma takie samo name oraz species. Wiek nie jest brany pod uwagę przy porównaniu.

3.2 Klasa dziedzicząca - Dog

```
1 class Dog(Animal):
2     def __init__(self, name, age, breed):
3         super().__init__(name, age, "Pies")
4         self.breed = breed
5
6     def __str__(self):
7         return f"Pies: {self.name}, rasa: {self.breed}, wiek: {self.age} lat"
```

Klasa Dog dziedziczy po klasie Animal jej właściwości i metody, ale rozszerza je o dodatkowe pole rasy psa. Klasa ta zawiera:

- Atrybuty dziedziczone:
 - **name** - imię zwierzęcia
 - **age** - wiek zwierzęcia
 - **species** - gatunek zwierzęcia
- Atrybuty dodatkowe:
 - **breed** - rasa psa
- Metody:
 - **__init__(self, name, age, breed)** - konstruktor klasy Dog, który wywołuje konstruktor klasy Animal i dodaje atrybut breed
 - **super().__init__(name, age, "Pies")** - Wywołuje konstruktor klasy bazowej Animal, przekazując imię, wiek oraz "Pies" jako gatunek zwierzęcia.
 - **__str__(self)** - metoda zwracająca opis tekstowy obiektu psa

3.3 Klasa dziedzicząca - Cat

```
1 class Cat(Animal):
2     def __init__(self, name, age, color):
3         super().__init__(name, age, "Kot")
4         self.color = color
5
6     def __str__(self):
7         return f"Kot: {self.name}, kolor: {self.color}, wiek: {self
            .age} lat"
```

Klasa Cat dziedziczy po klasie Animal jej właściwości i metody ale rozszerza je o dodatkowe pola koloru kota. Klasa ta zawiera:

- Atrybuty dziedziczone:
 - **name** - imię zwierzęcia
 - **age** - wiek zwierzęcia
 - **species** - gatunek zwierzęcia
- Atrybuty dodatkowe:
 - **color** - kolor kota
- Metody:
 - **__init__(self, name, age, color)** - konstruktor klasy Cat, który wywołuje konstruktor klasy Animal i dodaje atrybut color
 - **super().__init__(name, age, "Kot")** - Wywołuje konstruktor klasy bazowej Animal, przekazując imię, wiek oraz "Kot" jako gatunek zwierzęcia.
 - **__str__(self)** - metoda zwracająca opis tekstowy obiektu kot

3.4 Klasa Shelter

```
1 class Shelter:
2     def __init__(self):
3         self.animals = []
4         self.current_file = None
5
6
7     def add_animal(self, animal):
8         self.animals.append(animal)
9
10    def remove_animal(self, name):
11        self.animals = [a for a in self.animals if a.name != name]
12
13    def list_animals(self):
14        return self.animals
```

```

15
16     def save_to_file(self, filename=None):
17         if filename is None:
18             filename = self.current_file
19         if filename is None:
20             return False
21         try:
22             with open(filename, 'w', encoding='utf-8') as f:
23                 for a in self.animals:
24                     f.write(str(a) + '\n')
25         except Exception as e:
26             print(f"Blad zapisu: {e}")
27
28     def get_all_animals(self):
29         return self.animals
30
31     def load_data_from_file(self, filename):
32         self.animals = []
33         self.current_file = filename
34         try:
35             with open(filename, "r", encoding="utf-8") as file:
36                 lines = file.readlines()
37                 for line in lines:
38                     line = line.strip()
39                     if line.startswith("Kot"):
40
41                         name = line.split(":", 1)[1].split(",")[0].strip()
42                         color = line.split("kolor:", 1)[1].split(",")[0].strip()
43                         age = int(line.split("wiek:", 1)[1].split("lat")[0].strip())
44                         self.add_animal(Cat(name, age, color))
45                     elif line.startswith("Pies"):
46                         name = line.split(":", 1)[1].split(",")[0].strip()
47                         breed = line.split("rasa:", 1)[1].split(",")[0].strip()
48                         age = int(line.split("wiek:", 1)[1].split("lat")[0].strip())
49                         self.add_animal(Dog(name, age, breed))
50         except FileNotFoundError:
51             print("Plik nie istnieje. Tworzymy nowy plik.")
52
53     def open_file(self):
54         if self.animals:
55             answer = messagebox.askyesnocancel("Otwórz nowy plik",

```

```

56         "Czy chcesz zapisać dane przed otwarciem nowego pliku?"
57     )
58     if answer is None:
59         return
60     if answer:
61         if self.current_file:
62             self.save_to_file()
63         else:
64             file_path = filedialog.asksaveasfilename(
65                 defaultextension=".txt",
66                 filetypes=[("Pliki tekstowe", "*.txt")])
67             if not file_path:
68                 return pliku
69             self.save_to_file(file_path)
70
71     filename = filedialog.askopenfilename(filetypes=[("Pliki
72     tekstowe", "*.txt")])
73     if filename:
74         self.load_data_from_file(filename)
75         self.current_file = filename

```

Klasa Shelter reprezentuje schronisko oraz umożliwia zarządzanie listą zwierząt, zapisywanie i wczytywanie danych z plików oraz interakcję z użytkownikiem za pomocą GUI. Elementy klasy Shelter:

- **__init__(self)** - Konstruktor
- **self.animals** - Lista przechowująca zwierzęta
- **self.current_file** - Ścieżka do aktualnie używanego pliku
- **add_animal(self, animal)** - Metoda dodająca zwierzę do listy self.animals
- **remove_animal(self, name)** - Metoda usuwająca zwierzę o podanym imieniu z listy zwierząt.
- **list_animals(self)** - Metoda zwracająca listę wszystkich zwierząt
- **save_to_file(self, filename = None)** - Zapisuje dane o zwierzętach do pliku tekstowego. Jeśli nie podano filename to zapisuje do pliku self.current_file.
- **load_data_from_file(self, filename)** - Wczytuje dane z pliku tekstowego oraz czyści listę self.animals. Funkcja odczytuje każdą linię z pliku i analizuje czy linia zaczyna się od słowa "Pies" czy "Kot". Jeśli plik nie istnieje to funkcja informuje o tym, jednak nie przerywa działania.
- **open_file(self)** - Funkcja obsługuje otwieranie nowego pliku przez GUI. Jeśli istnieją dane w self.animals to program pyta użytkownika, czy zapisać dane. W momencie, gdy użytkownik kliknie przycisk anuluj - operacja zostaje przerwana, w przeciwnym razie pojawi się okno do wyboru lokalizacji. Ostatecznie funkcja zapisuje ścieżkę w self.current.file

3.5 Interfejs graficzny, czyli klasa ShelterApp

Dla czytelności podzielimy ją na mniejsze części.

3.5.1 Konstruktor

```
1 class ShelterApp:
2     def __init__(self, root):
3         self.root = root
4         self.root.title("Schronisko dla ązwierzt")
5         self.shelter = Shelter()
6         self.sort_ascending = True
7         self.animals = []
8
9         image = Image.open("pics/tlo.png")
10        bg_image = ImageTk.PhotoImage(image)
11        root.geometry(f"{image.width}x{image.height}")
12        root.resizable(False, False)
13        imgico = PhotoImage(file="pics/ikona.ico")
14        self.root.iconphoto(False, imgico)
15
16        self.canvas = tk.Canvas(root, width=image.width, height=image.
17            height)
18        self.canvas.pack(fill="both", expand=True)
19        self.canvas.create_image(0, 0, image=bg_image, anchor="nw")
20        self.bg_image = bg_image
21
22        menu_frame = tk.Frame(self.canvas, bg="")
23        menu_frame.pack(pady=20, padx=(0, 290))
24
25        btns = [
26            ("Dodaj psa", self.add_dog),
27            ("Dodaj kota", self.add_cat),
28            ("śWywietl wszystkie", self.show_all),
29            ("ńUsu po imieniu", self.remove_animal),
30            ("Zapisz do pliku", self.save_to_file),
31            ("Otwórz plik", self.shelter.open_file)]
32
33        for i, (label, command) in enumerate(btns):
34            tk.Button(menu_frame, text=label, command=command, width=20,
35                font=("Comic Sans MS", 10), bg="#c1d6ec", fg="#103051") \
36                .grid(row=i, column=0, pady=5)
37
38        self.root.protocol("WM_DELETE_WINDOW", self.on_exit)
```

Konstruktor klasy ShelterApp tworzy i konfiguruje graficzny interfejs użytkownika aplikacji do zarządzania schroniskiem dla zwierząt, ustawiając tło, ikonę, przyciski oraz ramkę menu. Dodatkowo inicjalizuje obiekty i zmienne potrzebne do działania aplikacji, takie jak lista zwierząt i instancja klasy Shelter.

Kolejno wymienione elementy konstruktora:

- Podstawowa konfiguracja okna, oraz inicjalizacja obiektów i zmiennych
- Ustawienie tła i ikonki
- Canvas zawierający tło
- Menu z przyciskami
- Zamykanie okna aplikacji

3.5.2 Metoda Add_dog

```
1      def add_dog(self):
2          add_window = tk.Toplevel(self.root)
3          add_window.title("Dodaj psa")
4
5          add_window.geometry("200x250")
6          add_window.resizable(False, False)
7          add_window.configure(bg="#f8d4e9")
8
9
10         tk.Label(add_window, text="Imię:", font=("Comic Sans MS", 10), bg="#f8d4e9", fg="#993355").pack(pady=5)
11         name_entry = tk.Entry(add_window)
12         name_entry.pack(pady=5)
13
14         tk.Label(add_window, text="Wiek:", font=("Comic Sans MS", 10), bg="#f8d4e9", fg="#993355").pack(pady=5)
15         age_entry = tk.Entry(add_window)
16         age_entry.pack(pady=5)
17
18         tk.Label(add_window, text="Rasa:", font=("Comic Sans MS", 10), bg="#f8d4e9", fg="#993355").pack(pady=5)
19         breed_entry = tk.Entry(add_window)
20         breed_entry.pack(pady=5)
21
22         def submit_dog():
23             name = name_entry.get()
24             age = age_entry.get()
25             breed = breed_entry.get()
26
27             if not name or not age or not breed:
28                 messagebox.showwarning("Uwaga", "Wszystkie pola muszą być wypełnione.")
29             return
30
```

```

31     if not name.isalnum():
32         messagebox.showwarning("!Bd", "Imi nie moe zawiera znaków
           specjalnych.")
33     return
34
35     try:
36         age = int(age)
37     except ValueError:
38         messagebox.showwarning("!Bd", "Wiek musi być aliczb.")
39     return
40
41     if age <= 0:
42         messagebox.showwarning("!Bd", "Wiek musi być większy niż 0.")
43     return
44
45     self.shelter.add_animal(Dog(name, age, breed))
46     messagebox.showinfo("Dodano", f"Dodano psa: {name}")
47     add_window.destroy()
48
49     submit_button = tk.Button(add_window, text="Dodaj psa", command=
           submit_dog, font=("Comic Sans MS", 10, "bold"), bg="#f5bede", fg="
           #76233f")
50     submit_button.pack(pady=10)

```

Funkcja `add_dog` tworzy nowe okno dialogowe, w którym użytkownik może wprowadzić dane psa do dodania do schroniska, takie jak imię, wiek i rasa. Po wprowadzeniu danych i ich walidacji, pies zostaje zapisany w systemie, a użytkownik otrzymuje potwierdzenie. **Elementy metody `add_dog`:**

- Utworzenie nowego okna (Toplevel) z tytułem, rozmiarem i kolorem tła
- Dodanie pól tekstowych do wprowadzania danych psa (imię, wiek, rasa)
- Obsługa przycisku „Dodaj psa”, który:
 - sprawdza poprawność danych (czy pola są wypełnione, poprawny wiek, brak znaków specjalnych w imieniu)
 - tworzy obiekt klasy `Dog` i dodaje go do schroniska za pomocą metody `add_animal`
 - wyświetla komunikaty informujące o sukcesie lub błędach
 - zamyka okno po dodaniu psa
- Przycisk końcowy wywołujący proces dodawania psa

3.5.3 Metoda `Add_cat`

```

1     def add_cat(self):
2         add_window = tk.Toplevel(self.root)
3         add_window.title("Dodaj kota")
4

```



```

5 add_window.geometry("200x250")
6 add_window.resizable(False, False)
7 add_window.configure(bg="#f6cfa1")
8
9 tk.Label(add_window, text="Imię:", font=("Comic Sans MS", 10), bg="#
    f6cfa1", fg="#913f13").pack(pady=5)
10 name_entry = tk.Entry(add_window)
11 name_entry.pack(pady=5)
12
13 tk.Label(add_window, text="Wiek:", font=("Comic Sans MS", 10), bg="#
    f6cfa1", fg="#913f13").pack(pady=5)
14 age_entry = tk.Entry(add_window)
15 age_entry.pack(pady=5)
16
17 tk.Label(add_window, text="Kolor:", font=("Comic Sans MS", 10), bg="#
    f6cfa1", fg="#913f13").pack(pady=5)
18 color_entry = tk.Entry(add_window)
19 color_entry.pack(pady=5)
20
21 def submit_cat():
22     name = name_entry.get()
23     age = age_entry.get()
24     color = color_entry.get()
25
26     if not name or not age or not color:
27         messagebox.showwarning("Uwaga", "Wszystkie pola muszą być wypełnione.")
28     return
29
30     if not name.isalnum():
31         messagebox.showwarning("Uwaga", "Imię nie może zawierać znaków
        specjalnych.")
32     return
33
34     try:
35         age = int(age)
36     except ValueError:
37         messagebox.showwarning("Uwaga", "Wiek musi być liczbą.")
38     return
39
40     if age <= 0:
41         messagebox.showwarning("Uwaga", "Wiek musi być większy niż 0.")
42     return
43
44     self.shelter.add_animal(Cat(name, age, color))
45     messagebox.showinfo("Dodano", f"Dodano kota: {name}")
46     add_window.destroy()
47

```

```

48 submit_button = tk.Button(add_window, text="Dodaj kota", command=
    submit_cat, font=("Comic Sans MS", 10, "bold"), bg="#f1bd80", fg="
    #70310f")
49 submit_button.pack(pady=10)

```

Funkcja `add_cat` otwiera nowe okno dialogowe umożliwiające użytkownikowi dodanie kota do systemu schroniska poprzez wprowadzenie jego imienia, wieku i koloru. Po weryfikacji danych tworzony jest obiekt kota, który zostaje zapisany w schronisku, a użytkownik otrzymuje odpowiedni komunikat. **Elementy metody `add_cat`:**

- Utworzenie nowego okna (Toplevel) z tytułem, wymiarami oraz kolorystyką dostosowaną do tematu
- Dodanie pól tekstowych do wprowadzania danych kota (imię, wiek, kolor)
- Obsługa przycisku „Dodaj kota”, który:
 - sprawdza, czy wszystkie pola są wypełnione
 - weryfikuje poprawność imienia (czy nie zawiera znaków specjalnych)
 - konwertuje wiek na liczbę całkowitą i sprawdza, czy jest większy od zera
 - tworzy obiekt klasy `Cat` i dodaje go do systemu przy użyciu metody `add_animal`
 - wyświetla odpowiedni komunikat (sukces lub błąd) oraz zamyka okno po dodaniu
- Przycisk końcowy umożliwiający zatwierdzenie i dodanie kota

3.5.4 Metody `show_all`, `remove_animal`, `save_to_file`

```

1  def show_all(self):
2      animals = self.shelter.list_animals()
3      self.show_in_new_window("Wszystkie zwierzęta", animals)
4
5  def remove_animal(self):
6      remove_window = tk.Toplevel(self.root)
7      remove_window.title("Usuń zwierzę")
8      remove_window.geometry("200x100")
9      remove_window.configure(bg="#f8b5b5")
10     remove_window.resizable(False, False)
11
12     tk.Label(remove_window, text="Imię:", font=("Comic Sans MS", 10), bg="#
        f8b5b5", fg="#8f1313").pack(pady=5)
13     name_entry = tk.Entry(remove_window)
14     name_entry.pack(pady=5)
15
16     def submit_remove():
17         name = name_entry.get()
18         if name:
19             self.shelter.remove_animal(name)

```

```

20         messagebox.showinfo("Usunito", f"Usunito ęzvierz o imieniu: {name}"
21         ")
22         remove_window.destroy()
23
24 tk.Button(remove_window, text="ńUsu", command=submit_remove,bg="#f19696
25         ", fg="#660a0a", font=("Comic Sans MS", 10, "bold")).pack(pady=3)
26
27 def save_to_file(self):
28     filename = filedialog.asksaveasfilename(defaultextension=".txt",
29     filetype=[("Text files", "*.txt")])
30     if filename:
31         self.shelter.save_to_file(filename)
32         messagebox.showinfo("Zapisano", f"Dane zapisane do: {filename}")

```

- **show_all** pobiera listę wszystkich zwierząt zarejestrowanych w schronisku i wyświetla je w nowym oknie.
 - Wywołanie metody list_animals() z obiektu shelter, zwracającej dane o wszystkich zwierzętach
 - Przekazanie pobranych danych do metody show_in_new_window, która prezentuje je w osobnym oknie z tytułem „Wszystkie zwierzęta”
- **remove_animal** otwiera małe okno, w którym użytkownik może podać imię zwierzęcia do usunięcia z systemu schroniska.
 - Utworzenie nowego okna (Toplevel) z tytułem, rozmiarem i kolorystyką
 - Pole tekstowe do wpisania imienia zwierzęcia
 - Przycisk „Usuń”, który:
 - * odczytuje wprowadzone imię
 - * wywołuje metodę remove_animal na obiekcie shelter, przekazując imię
 - * wyświetla komunikat o usunięciu i zamyka okno
- **save_to_file** pozwala użytkownikowi zapisać dane schroniska do pliku tekstowego wybranego z poziomu okna dialogowego.
 - Otwiera okno wyboru pliku do zapisu (z domyślnym rozszerzeniem .txt)
 - Jeśli użytkownik wybierze plik, wywołuje metodę save_to_file na obiekcie shelter, przekazując ścieżkę
 - Wyświetla komunikat potwierdzający pomyślne zapisanie danych

3.5.5 Metoda refresh_table

```
1 def refresh_table(self, frame, animals=None, title="", sort_by=None,
2   parent=None, canvas=None, species_filter="Wszystkie"):
3     if animals is None:
4       animals = self.shelter.get_all_animals()
5
6     if species_filter != "Wszystkie":
7       cls = Dog if species_filter == "Psy" else Cat
8       animals = [a for a in animals if isinstance(a, cls)]
9
10    if sort_by == "age":
11      animals = sorted(animals, key=lambda x: x.age, reverse=not self.
12        sort_ascending)
13
14    for w in frame.winfo_children(): w.destroy()
15    self.show_in_new_window(title, animals, reuse_frame=frame, sort_by=
16      sort_by,
17      parent_window=parent, canvas=canvas, species_filter=species_filter)
```

Funkcja refresh_table służy do odświeżenia widoku tabeli zawierającej dane zwierząt w zależności od wybranych filtrów i kryteriów sortowania. Działa dynamicznie na przekazanym kontenerze graficznym, aktualizując jego zawartość. **Elementy metody refresh_table:**

- Pobranie listy zwierząt z obiektu shelter, jeśli nie została ona przekazana jako argument
- Filtrowanie zwierząt według wybranego gatunku (Psy, Koty lub Wszystkie)
- Sortowanie według wieku, jeśli użytkownik wybrał taką opcję — rosnąco lub malejąco w zależności od stanu self.sort_ascending
- Usunięcie wszystkich elementów z przekazanego frame, by przygotować miejsce na nową zawartość
- Ponowne wyświetlenie odświeżonej listy zwierząt w tym samym frame za pomocą metody show_in_new_window, z uwzględnieniem filtrów i opcji

3.5.6 Metoda show_in_new_window

```
1 def show_in_new_window(self, title, animals, reuse_frame=None,
2   sort_by=None, parent_window=None, canvas=None,
3   species_filter="Wszystkie"):
4     if reuse_frame and parent_window and canvas:
5       frame = reuse_frame
6     else:
7       parent_window = tk.Toplevel(self.root)
8       parent_window.title(title)
9       parent_window.geometry(f"470x{self.root.winfo_height()}")
10      parent_window.resizable(False, False)
```

```

11
12 canvas = tk.Canvas(parent_window, bg="#e6f4e6")
13 scrollbar = tk.Scrollbar(parent_window, orient="vertical", command=
    canvas.yview)
14 canvas.configure(yscrollcommand=scrollbar.set)
15 frame = tk.Frame(canvas, bg="#e6f4e6")
16 canvas.create_window((0, 0), window=frame, anchor="nw")
17 scrollbar.pack(side="right", fill="y")
18 canvas.pack(side="left", fill="both", expand=True)
19
20 def limited_mouse_wheel(event):
21     if event.delta > 0 and canvas.yview()[0] <= 0:
22         return
23     canvas.yview_scroll(int(-1 * (event.delta / 120)), "units")
24
25 parent_window.bind_all("<MouseWheel>", limited_mouse_wheel)
26
27 headers = ["Gatunek", "ęImi", "Wiek", "Rasa/Kolor"]
28 for col, h in enumerate(headers):
29     if h == "Wiek":
30         def toggle_sort():
31             self.sort_ascending = not self.sort_ascending
32             self.refresh_table(frame, title=title, sort_by="age",
33                 parent=parent_window, canvas=canvas, species_filter=species_filter)
34
35         btn = tk.Button(frame, text=f"Wiek {'(łstrzaka w górę)' if self.
            sort_ascending else '(łstrzaka w dół)'}",
36             font=("Comic Sans MS", 11, "bold"),
37             bg="#b2d8b2", fg="#0e6c0e", relief="ridge", borderwidth=1, padx=10,
            pady=6,
38             command=toggle_sort)
39         btn.grid(row=0, column=col, sticky="nsew", padx=1, pady=1)
40
41     elif h == "Gatunek":
42         next_filter = {"Wszystkie": "Psy", "Psy": "Koty", "Koty": "Wszystkie"}[
            species_filter]
43         btn = tk.Button(frame, text=species_filter, font=("Comic Sans MS", 11,
            "bold"),
44             bg="#b2d8b2", fg="#0e6c0e", relief="ridge", borderwidth=1, padx=10,
            pady=6,
45             command=lambda: self.refresh_table(frame, title=title,
46                 parent=parent_window, canvas=canvas,
47                 species_filter=next_filter)
48         )
49         btn.grid(row=0, column=col, sticky="nsew", padx=1, pady=1)
50     else:
51         tk.Label(frame, text=h, font=("Comic Sans MS", 11, "bold"), bg="#b2d8b2",
            fg="#0e6c0e",

```

```

52     relief="ridge", borderwidth=1, padx=10, pady=6).grid(row=0, column=col,
53         sticky="nsew", padx=1,
54         pady=1)
55
56     for row, animal in enumerate(animals, start=1):
57         values = ["(emotka psa)" if isinstance(animal, Dog) else "(emotka kota)
58             ", animal.name, animal.age,
59             animal.breed if isinstance(animal, Dog) else animal.color]
60         for col, val in enumerate(values):
61             tk.Label(frame, text=val, font=("Comic Sans MS", 10), bg="#d9f2d9", fg=
62                 "#0e6c0e",
63                 relief="ridge", borderwidth=1, padx=8, pady=4).grid(row=row, column=col
64                     , sticky="nsew", padx=1, pady=1)
65
66     frame.update_idletasks()
67     canvas.config(scrollregion=canvas.bbox("all"))

```

Funkcja `show_in_new_window` służy do prezentowania listy zwierząt w osobnym oknie lub odświeżonej ramce, umożliwiając sortowanie według wieku i filtrowanie po gatunku. Tworzy interaktywny widok tabeli z nagłówkami i danymi, zawierający m.in. przewijanie i dynamiczne przełączniki. **Elementy metody `show_in_new_table`:**

- Warunkowe wykorzystanie istniejącej ramki (`reuse_frame`) lub utworzenie nowego okna z Canvasem i paskiem przewijania
- Definicja własnej obsługi scrollowania myszką, umożliwiającą przewijanie listy tylko w odpowiednich warunkach
- Wyświetlenie nagłówków tabeli:
 - „Wiek” — jako przycisk umożliwiający sortowanie rosnąco/malejąco
 - „Gatunek” — jako przycisk umożliwiający filtrowanie między: Wszystkie, Psy, Koty
 - Pozostałe nagłówki jako etykiety
- Wypełnienie tabeli danymi zwierząt:
 - Oznaczenie gatunku ikoną (psa lub okna)
 - Wyświetlenie imienia, wieku oraz rasy (dla psa) lub koloru (dla kota)
- Aktualizacja obszaru przewijania w Canvasie po zakończeniu tworzenia widoku

3.5.7 Metoda `on_exit`

```

1     def on_exit(self):
2     def ask_save():
3         answer = messagebox.askyesnocancel("Zamknij", "Czy chcesz zapisać dane
4             przed wyjściem?")
5         if answer is None:
6             return False

```

```

6     if answer:
7     if self.shelter.current_file:
8     self.shelter.save_to_file()
9     return True
10    else:
11    file_path = filedialog.asksaveasfilename(defaultextension=".txt",
12    filetypes=[("Pliki tekstowe", "*.txt")])
13    if not file_path:
14    return ask_save()
15    self.shelter.save_to_file(file_path)
16    return True
17    else:
18    return True
19
20    if ask_save():
21    self.root.destroy()

```

Funkcja `on_exit` obsługuje zamykanie aplikacji, pytając użytkownika, czy chce zapisać dane przed wyjściem, i wykonując odpowiednie działania w zależności od jego decyzji. Dzięki temu zabezpiecza użytkownika przed przypadkową utratą danych. **Elementy metody `on_exit`:**

- Wywołanie okna dialogowego z pytaniem, czy zapisać dane — użytkownik może wybrać: Tak, Nie lub Anuluj
- Jeśli użytkownik wybierze „Tak”:
 - Dane zostają zapisane do bieżącego pliku, jeśli istnieje
 - W przeciwnym razie użytkownik zostaje poproszony o wskazanie pliku do zapisu
- Jeśli użytkownik wybierze „Nie” — aplikacja zostaje zamknięta bez zapisu
- Jeśli wybierze „Anuluj” — proces zamykania zostaje przerwany
- W przypadku potwierdzenia (Tak lub Nie), aplikacja zostaje zamknięta (`self.root.destroy()`)

3.5.8 Uruchomienie GUI

```

1     if __name__ == "__main__":
2     root = tk.Tk()
3     app = ShelterApp(root)
4     root.mainloop()

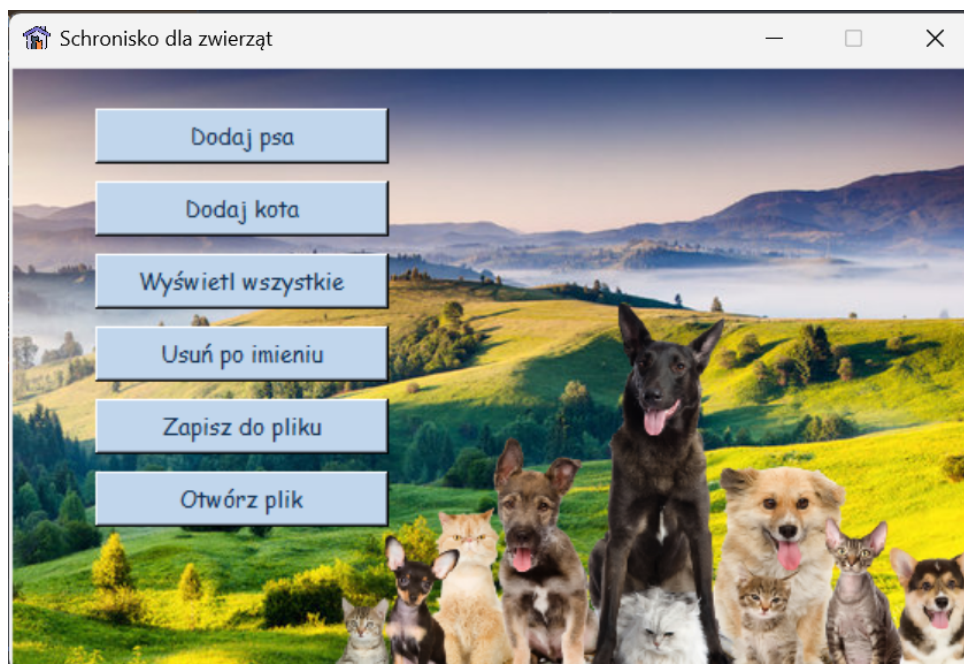
```

Ta część tworzy główne okno programu oraz instancję klasy `ShelterApp`, a następnie uruchamia główną pętlę aplikacji

- `tk.Tk()` tworzy nam główne okno
- `ShelterApp(root)` to inicjalizacja aplikacji
- `root.mainloop()` odpowiada za uruchomienie GUI

4 Działanie programu

Po włączeniu aplikacji wyskoczy nam okienko z uroczym menu.



Rysunek 1: Główne menu aplikacji

4.1 Dodaj psa

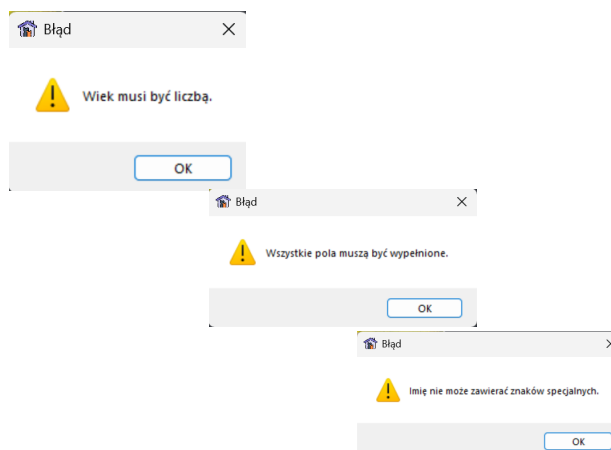
Po kliknięciu opcji dodaj psa zobaczymy małe okienko pozwalające nam dodać nowego psa do naszego schroniska. Należy pamiętać, że wprowadzone dane muszą być poprawne, tzn.

- Wiek musi być liczbą całkowitą
- Imię nie może zawierać znaków specjalnych
- Wszystkie pola muszą być wypełnione

Jeżeli wprowadzimy błędne dane, zobaczymy odpowiednie komunikaty, a zwierzę się nie doda do bazy.



Rysunek 2: Dodaj psa



Rysunek 3: Poszczególne komunikaty

4.2 Dodaj kota

Po kliknięciu opcji dodaj kota zobaczymy bardzo podobne okienko do tego, które wyskakiwało przy dodawaniu psa. Jeżeli wprowadzimy błędne dane, uzyskamy taki sam rezultat jak przy dodawaniu psa.



Rysunek 4: Dodaj kota

4.3 Wyświetl wszystkie

Opcja wyświetl wszystkie otwiera nam nowe okno z dużą tabelą jasno ukazującą wszystkie zwierzęta, które aktualnie mamy w schronisku.



Wszystkie	Imię	Wiek ▲	Rasa/Kolor
	Kiti	2	szary
	Doda	12	jamnik
	Borys	5	beagle
	Mru	4	czarny
	Max	3	labrador
	Luna	6	biały
	Bella	7	owczarek niemiecki
	Riko	1	rudy
	Zeus	9	rottweiler
	Puszek	2	biały z szarymi plamkami

Rysunek 5: Wyświetl wszystkie

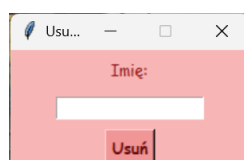


Koty	Imię	Wiek ▲	Rasa/Kolor
	Riko	1	rudy
	Simba	1	złoty
	Zosia	1	rudy
	Gacek	1	szaro-biały
	Rysiek	1	rudy
	Kiti	2	szary
	Puszek	2	biały z szarymi plamkami
	Rufus	2	rudy
	Kubuś	2	biały
	Figaro	3	czarno-biały

Rysunek 6: Posortowane zwierzęta

Z tego poziomu możemy filtrować zwierzęta, wyświetlając wszystkie, same psy albo same koty, a także sortować widoczne zwierzęta po wieku.

4.4 Usuń po imieniu

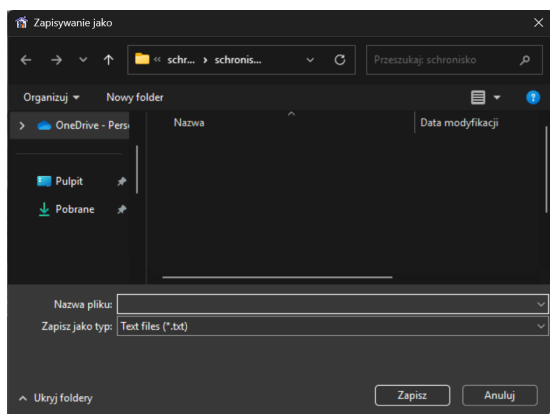


Rysunek 7: Okno usuwania zwierzęcia

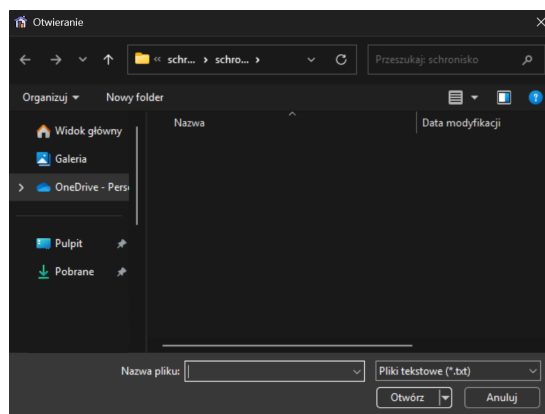
Po wpisaniu imienia zwierzęcia zostanie ono usunięte z bazy na której aktualnie pracujemy.

4.5 Zapisz do pliku, otwórz plik

Dwie ostatnie opcje to zapisywanie naszej pracy do pliku tekstowego oraz otwieranie pliku.

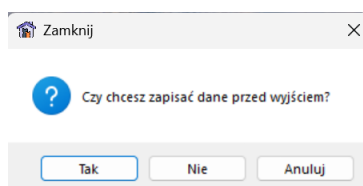


Rysunek 8: Zapisywanie do pliku



Rysunek 9: Otwieranie pliku

Dzięki temu nie musimy się bać, że nasza praca przepadnie, i będziemy mogli później do niej wrócić. Na prawdę nie musimy się o to martwić. Przed każdą możliwą utratą niezapisanych danych chroni nas okienko, które wyskakuje zarówno przy otwieraniu nowego pliku, jak i przy zamykaniu aplikacji.

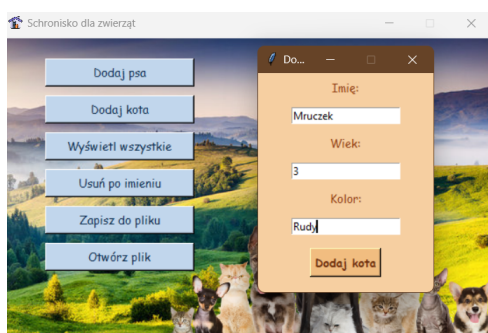


Rysunek 10: Komunikat

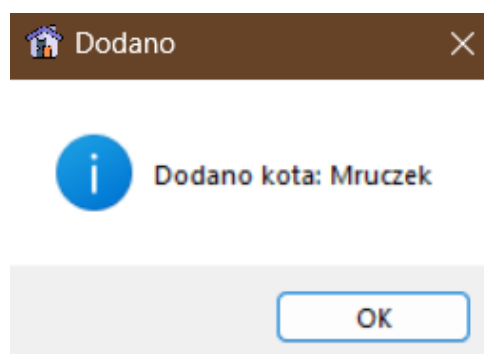
5 Przykładowe działanie programu

W przykładzie dodamy 3 koty i 2 psy oraz zapiszemy wprowadzone dane do pliku. W pierwszym kroku klikniemy na przycisk dodaj kota, wpisujemy jego dane i klikniemy znów na przycisk "Dodaj kota". Następnie pojawi się nam komunikat, że zapisano kota o podanym wcześniej imieniu. Po kliknięciu przycisku "OK" komunikat znika i zostaje samo menu aplikacji.

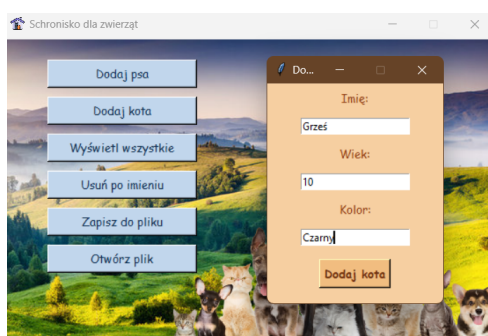
5.1 Dodanie kotów



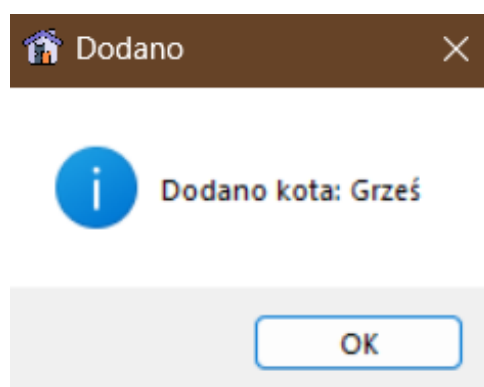
Rysunek 11: Dodanie pierwszego kota



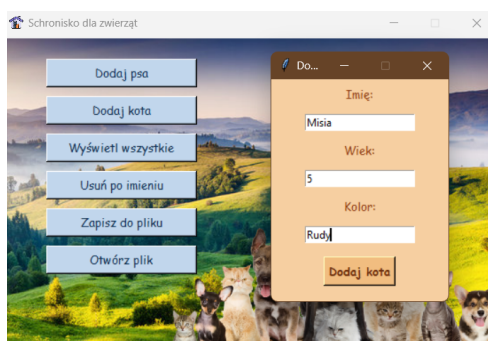
Rysunek 12: Komunikat o dodaniu



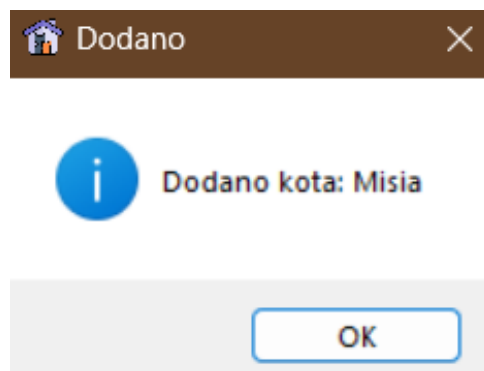
Rysunek 13: Dodanie drugiego kota



Rysunek 14: Komunikat o dodaniu

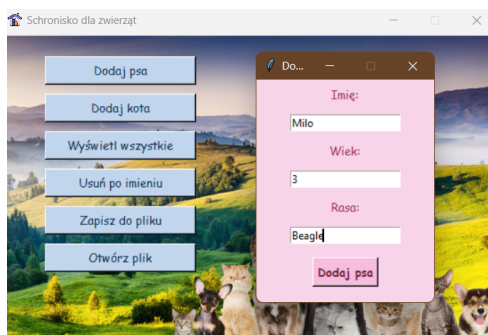


Rysunek 15: Dodanie trzeciego kota

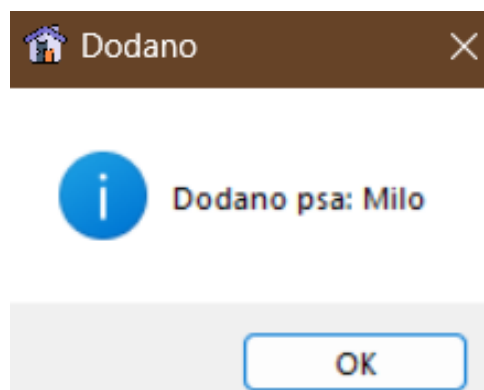


Rysunek 16: Komunikat o dodaniu

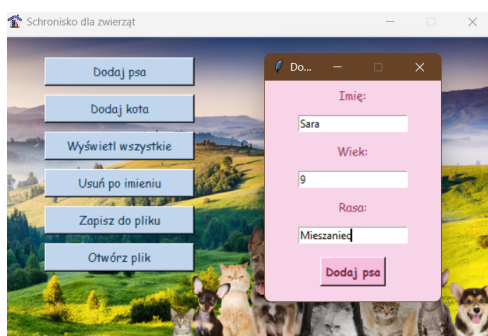
5.2 Dodanie psów



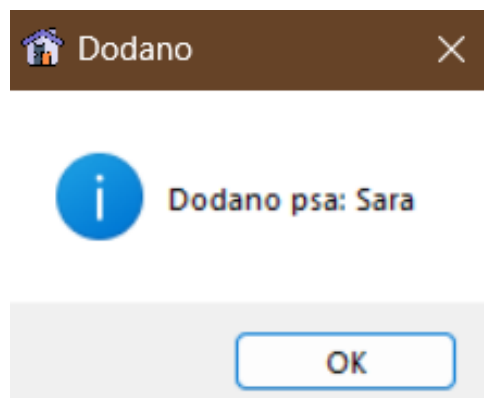
Rysunek 17: Dodanie pierwszego psa



Rysunek 18: Komunikat o dodaniu



Rysunek 19: Dodanie drugiego psa



Rysunek 20: Komunikat o dodaniu

5.3 Wyświetlenie wszystkich zwierząt

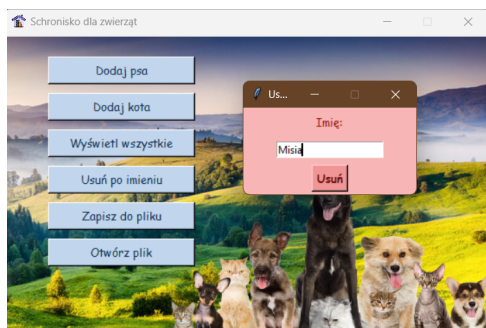
Wszystkie	Imię	Wiek ▾	Rasa/Kolor
	Grześ	10	Czarny
	Sara	9	Mieszaniec
	Misia	5	Rudy
	Milo	3	Beagle
	Mruczek	3	Rudy

Rysunek 21: Lista wszystkich zwierząt posortowana malejąco

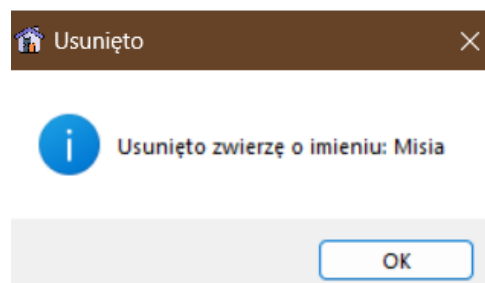
Wszystkie	Imię	Wiek ▲	Rasa/Kolor
	Milo	3	Beagle
	Mruczek	3	Rudy
	Misia	5	Rudy
	Sara	9	Mieszaniec
	Grześ	10	Czarny

Rysunek 22: Lista wszystkich zwierząt posortowana rosnąco

5.4 Usunięcie zwierzaka po imieniu



Rysunek 23: Usunięcie kota po imieniu

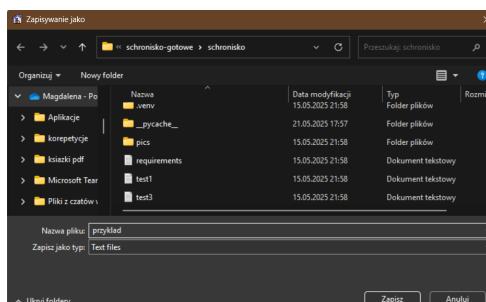


Rysunek 24: Komunikat o usunięciu

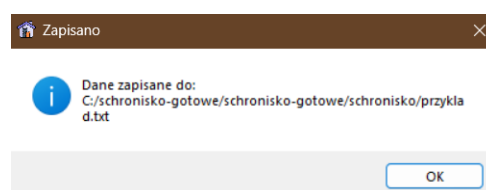
Wszystkie	Imię	Wiek [A]	Rasa/Kolor
	Grześ	10	Czarny
	Milo	3	Beagle
	Sara	9	Mieszaniec
	Mruczek	3	Rudy

Rysunek 25: Lista po usunięciu zwierzaka

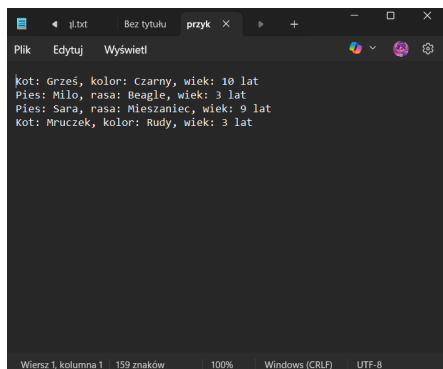
5.5 Zapisywanie listy do pliku



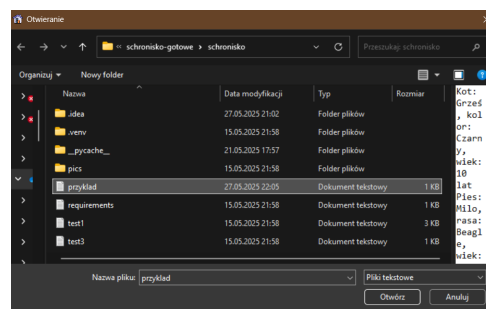
Rysunek 26: Zapisywanie listy



Rysunek 27: Komunikat o zapisaniu oraz lokalizacji pliku



Rysunek 28: Zapisywanie listy



Rysunek 29: Wczytanie pliku do aplikacji

6 Podsumowanie

Celem projektu było stworzenie wygodnej i przyjaznej dla użytkownika aplikacji, oferującej szeroki zakres funkcjonalności. Najważniejszym zadaniem programu jest ułatwienie pracownikom schroniska zarządzania danymi oraz korzystania z nich w intuicyjny sposób. Aplikacja umożliwia m.in. dodawanie i usuwanie zwierząt z bazy danych. W trakcie realizacji projektu wykorzystaliśmy wiedzę zdobytą podczas zajęć (m.in. tworzenie funkcji i klas w języku Python), a także umiejętności zdobyte samodzielnie, takie jak projektowanie zaawansowanego interfejsu graficznego.