CSE 2050 – Programming in a Second Language

Fall 2023

Homework 4: OOP

Total Points: 30

Date Assigned: Friday, Oct 27, 2023 **Due Date:** Sunday, Nov 5, 2023

Instructions: Please submit your work on Canvas as a zipped file named `cse2050_yourname_hw4.zip`. Make sure to include a main.py file as the driver for testing your program and organize all classes into separate files.

Key Concepts Demonstrated

- Writing programs based on the Objected Oriented Programming (OOP) Paradigm
  - Re-purposing code that displays images in a UI to display items about dogs
  - Writing code that uses objects that interact with each other
  - Utilizing Python's special OOP methods to display object summaries

1. (30 points) **K9 Viewer**

Given a JSON file named dog_breeds.json (locate it on Canvas) that contains images and descriptions of various dog breeds, complete the following:

Write an OOP program in Python by updating the code on slide 124 of the lecture slides for Week 9-11 to display in the EzGraphics UI data regarding the first 12 dog breeds in the file instead of the sample images given. Your program should have the following classes (shown in orange color below) and methods. All classes should have a constructor that initializes instance variables.

Dog - (8pts)
- get_breed_name() - gets the breed name of a dog
- get_description() - gets the description of a dog breed
- set_image(url) - downloads an image from the web and saves it as a GIF file.
- get_image() returns the name of an image file for saving or displaying in the UI (e.g.: appenzeller.gif. Note that saved images must have the same name as the image in the URL)

DogDataProcessor - (8pts)
- load_dogs(json_file, n=12) - reads the json file and creates *n* Dog objects and saves them to a list.
- get_dogs() - returns a list of Dog objects
- tabulate_dogs() - displays a tabulation (Breed, Description) of all dogs in the list (for description, display the first sentence)

DogUI - (10pts)
__init__() - initializes variables such as gaps, window height, width, and calls setup_window() to set up the window
- setup_window() - sets up the canvas, window, and assigns a window title
- layout_ui(dogs) - takes in a list of dog objects and displays a random image from the list of images and calls `show_dog_breed()` to display the breed name below the image on the Canvas as shown in the image overleaf. Make sure to add margins around your images.
You must display an ANSI and ASCII-based progress-bar in the command line when downloading images before the UI loads.
- show_dog_breed(dog, x, y) - displays the dog breed as text in the UI

main() method - (4pts)
- creates an instance of the DogDataProcessor that reads the JSON file and stores at least the first 50 Dog objects in a list
- creates an instance of the DogUI and displays data and images in it based on the dogs in the DogDataProcessor (see image overleaf)
- calls the tabulate_recipes() method to tabulate all dogs in the list

Hints

1. Read pages 210 - 222 of the textbook
2. Based on experimentation, the ezgraphics module supports only GIF images. Use the PIL module to convert the JPG images to GIF images using the code below. Scale your images to a reasonable size (e.g. 200px).

```
from PIL import Image
scaled_width = 200
img = Image.open('your_image.jpg')
percent_width = (scaled_width / float(img.size[0]))
h_size = int((float(img.size[1]) * float(percent_width)))
img = img.resize((scaled_width, h_size), Image.ANTIALIAS)
img.save('resized_image.gif') # save the img as a GIF for loading in ezgraphics
```


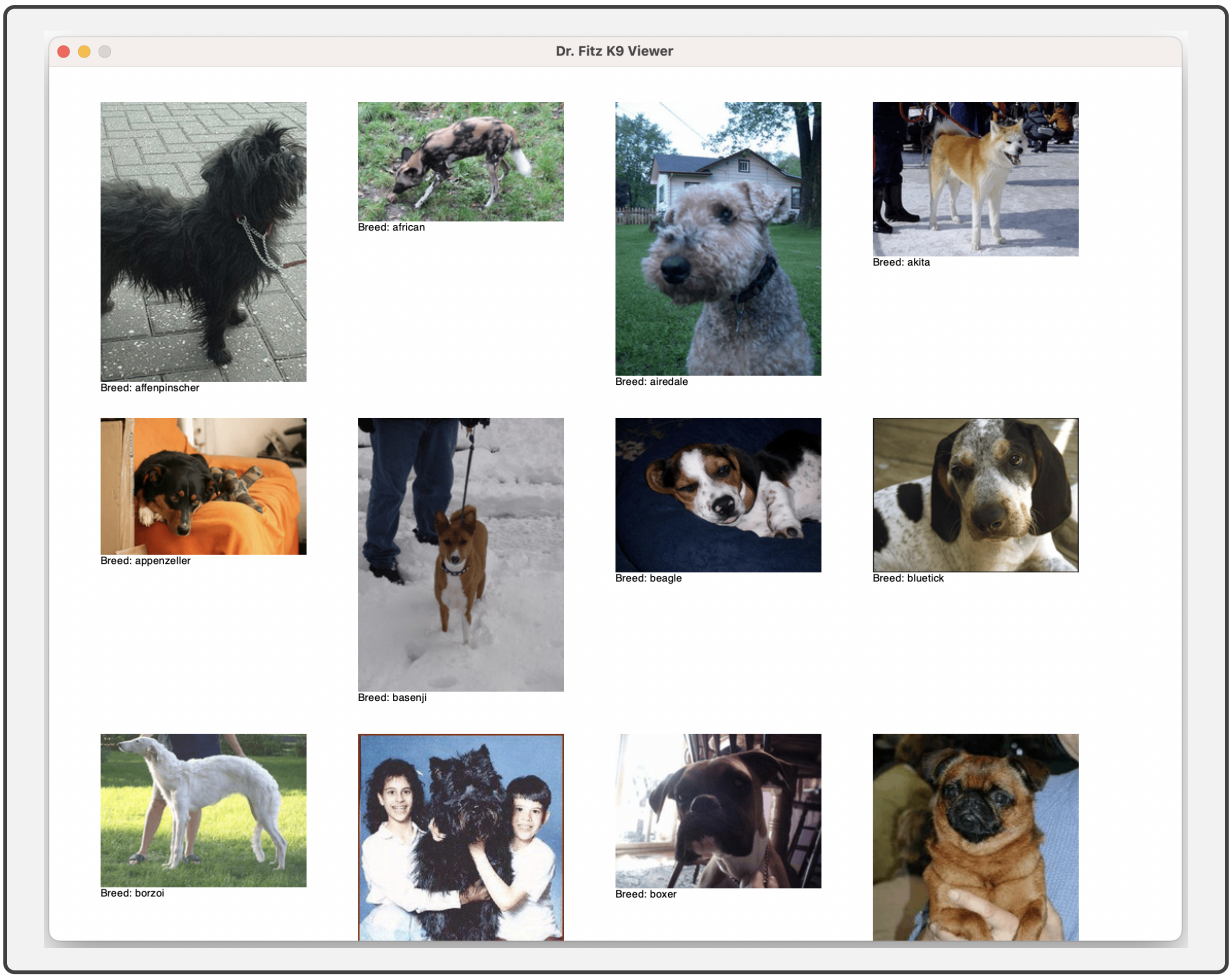

Figure 1: Example UI for the K9 Viewer