DIBA MEYSAMIAZAD

# ANIMAL SHELTER MANAGER

## OBJECT-ORIENTED PROGRAMMING PROJECT

DIBA MEYSAMIAZAD
1081211

June 2019

## Table of Contents

## Table of Figures

## Introduction

This project is a simple manager for the data of an animal shelter. It allows the management of containers of objects that represent different types of animals typically found in an animal shelter: Cats, dogs and birds.

The manager uses database files to represent containers and gives the user the possibility to create and save new files, open previously created files and apply changes to old or new files. These actions represent the definition of a new container, and the insertion or removal of elements from containers previously defined. Furthermore, searching on container objects based on their different attributes is possible.

## Classes and Hierarchy

The logical class hierarchy is as shown in the image below:

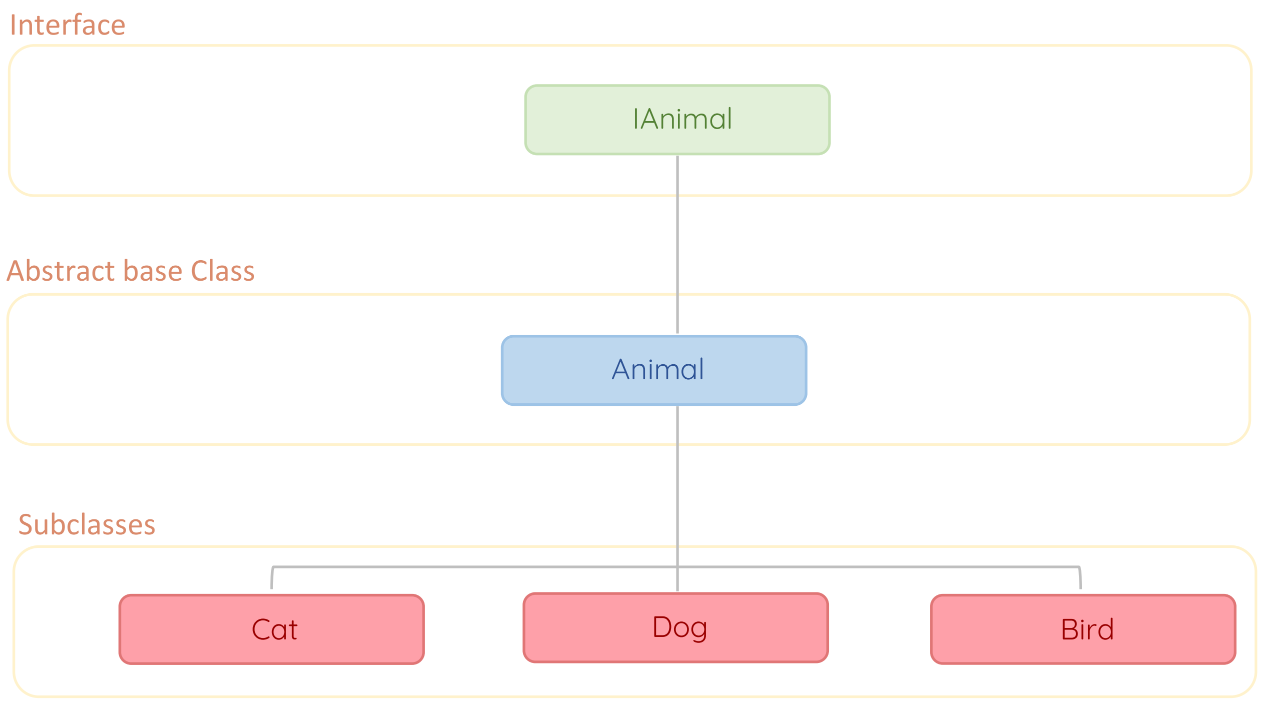


*Figure 1: Class hierarchy*

“IAnimal” is an abstract base class for the hierarchy that contains only virtual methods, i.e. an interface. “Animal” is an abstract base class and contains 6 data members representing properties in common between all shelter animals: ID, name, age, breed, adoption status and gender.

The class “Animal” has three subclasses representing three types of animals typically found in animal shelters: cats, dogs, and birds. Each of these subclasses has a data member representing the kind of vaccines typically administered to that certain type of animal; furthermore, the bird subclass has an additional data member that represents the bird species.

In addition to the above-mentioned hierarchy, the model includes a container template implemented using a dynamic array which is later used for holding deep pointers to objects of type “Animal”.

A few of the methods used in the hierarchy are:

- Different getter methods each of which return a specific attribute of the hierarchy.
- virtual std::string type() const = 0: returns a string that indicates the type of animal the object represents.
- virtual std::string vaccines() const = 0: returns the vaccines each animal should take based on its type. Each animal class has a different data member that represents the most typical vaccines that type of animal should receive.
- virtual double costOfCare() const = 0: calculates and returns the average cost of care for each animal using a static data member of the animal class as a base and adding a sum depending on the type and age of the animal each object represents.
- The class "Bird" has an additional data member representing its species. std::string species() const acts as a getter for this attribute.

## Architecture

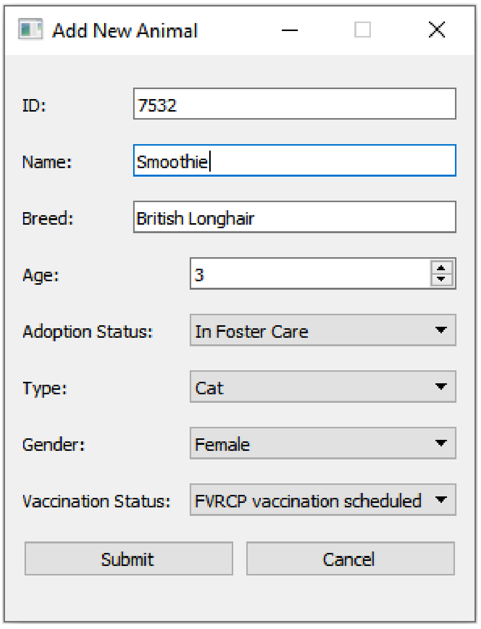
This project uses the Model-View architectural pattern, trying to use the view classes provided by Qt in a suitable way to represent the data available in the project.

## Example of Use

The main controls are found in the toolbar on the left. The main ones are: open file, save file, add item, delete item, view details, edit item, and search.

### Adding a new animal

Adding a new animal is quite intuitive. All the fields in the insertion form must be filled. The ID number must be a four-digit number. The "Vaccination Status" field becomes active only after having chosen the type of animal, as the values depend on the type of animal we are inserting. If the animal is a bird, a new "Species" field appears for the insertion of the species of the bird.

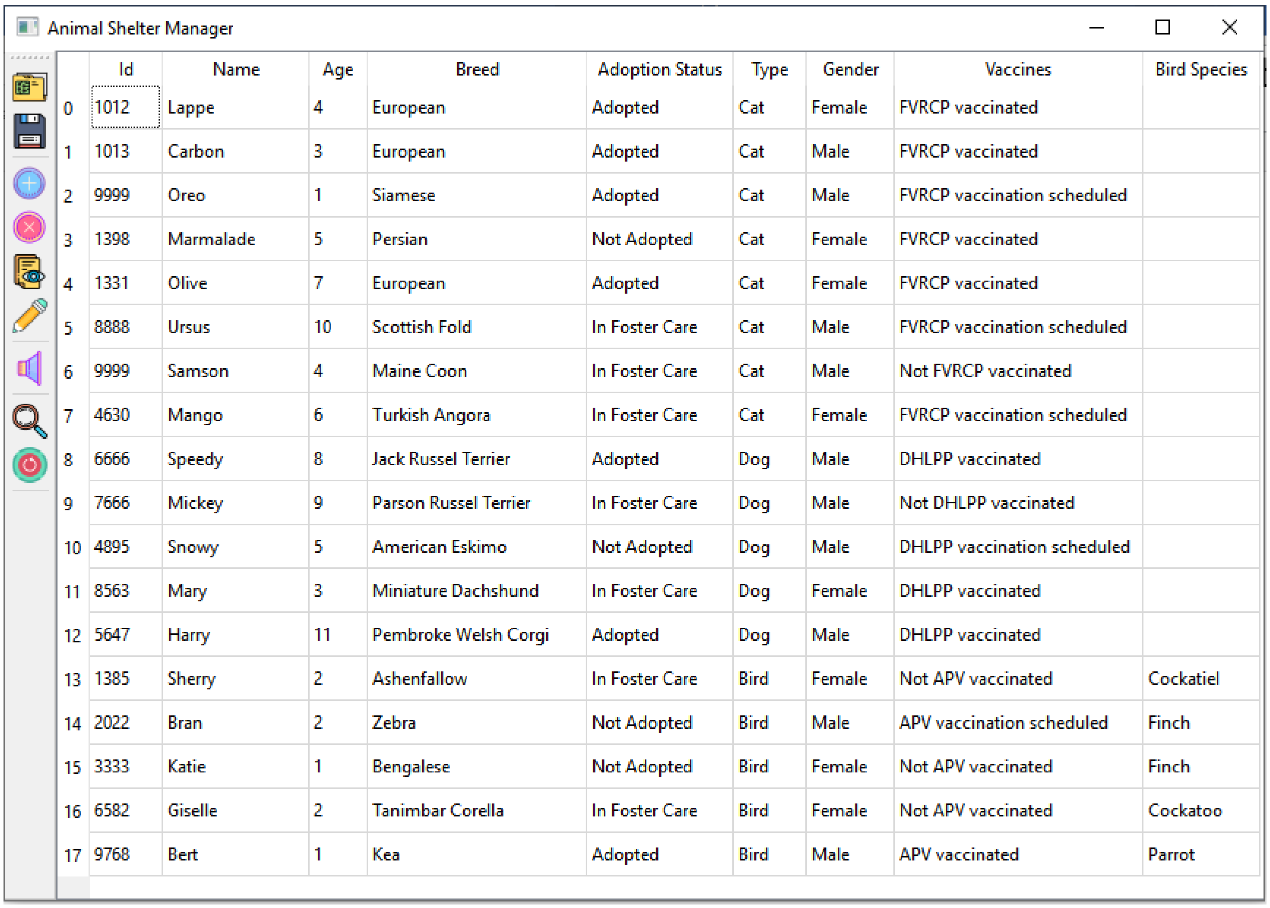


| | |
|---|---|
| ID: | 7532 |
| Name: | Smoothie |
| Breed: | British Longhair |
| Age: | 3 |
| Adoption Status: | In Foster Care |
| Type: | Cat |
| Gender: | Female |
| Vaccination Status: | FVRCP vaccination scheduled |

*Figure 2: Adding a new item*

### Viewing an item

A general view of the items is seen within the main window itself. A subclass of "QTableView" called "TableView" has been defined in order to hold the data in the main window (Figure 3). Viewing a single item in more detail is possible by clicking the "View details" button on the left toolbar. As can be seen in figure 4, in this way we can also see the average monthly cost of the caretaking of a certain animal for the shelter, calculated polymorphically by the costOfCare() function, as mentioned above.



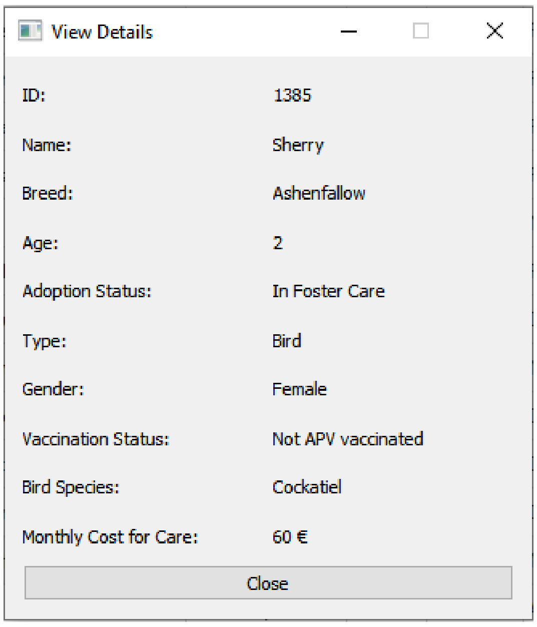
Animal Shelter Manager

| | Id | Name | Age | Breed | Adoption Status | Type | Gender | Vaccines | Bird Species |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1012 | Lappe | 4 | European | Adopted | Cat | Female | FVRCP vaccinated | |
| 1 | 1013 | Carbon | 3 | European | Adopted | Cat | Male | FVRCP vaccinated | |
| 2 | 9999 | Oreo | 1 | Siamese | Adopted | Cat | Male | FVRCP vaccination scheduled | |
| 3 | 1398 | Marmalade | 5 | Persian | Not Adopted | Cat | Female | FVRCP vaccinated | |
| 4 | 1331 | Olive | 7 | European | Adopted | Cat | Female | FVRCP vaccinated | |
| 5 | 8888 | Ursus | 10 | Scottish Fold | In Foster Care | Cat | Male | FVRCP vaccination scheduled | |
| 6 | 9999 | Samson | 4 | Maine Coon | In Foster Care | Cat | Male | Not FVRCP vaccinated | |
| 7 | 4630 | Mango | 6 | Turkish Angora | In Foster Care | Cat | Female | FVRCP vaccination scheduled | |
| 8 | 6666 | Speedy | 8 | Jack Russel Terrier | Adopted | Dog | Male | DHLPP vaccinated | |
| 9 | 7666 | Mickey | 9 | Parson Russel Terrier | In Foster Care | Dog | Male | Not DHLPP vaccinated | |
| 10 | 4895 | Snowy | 5 | American Eskimo | Not Adopted | Dog | Male | DHLPP vaccination scheduled | |
| 11 | 8563 | Mary | 3 | Miniature Dachshund | In Foster Care | Dog | Female | DHLPP vaccinated | |
| 12 | 5647 | Harry | 11 | Pembroke Welsh Corgi | Adopted | Dog | Male | DHLPP vaccinated | |
| 13 | 1385 | Sherry | 2 | Ashenfallow | In Foster Care | Bird | Female | Not APV vaccinated | Cockatiel |
| 14 | 2022 | Bran | 2 | Zebra | Not Adopted | Bird | Male | APV vaccination scheduled | Finch |
| 15 | 3333 | Katie | 1 | Bengalese | Not Adopted | Bird | Female | Not APV vaccinated | Finch |
| 16 | 6582 | Giselle | 2 | Tanimbar Corella | In Foster Care | Bird | Female | Not APV vaccinated | Cockatoo |
| 17 | 9768 | Bert | 1 | Kea | Adopted | Bird | Male | APV vaccinated | Parrot |

*Figure 3: Main window view of items*

## Searching for an item

By clicking the search button on the left toolbar, a dockable search box appears at the bottom of the screen (Figure 4). Items can be searched for by any of their attributes, or any subset of their attributes, thanks to the use of “QSortFilterProxyModel”. Searching is done in real-time, so no search button has been implemented. A subclass of the Qt class “QSortFilterProxyModel” has been implemented to adapt it to a search by a fixed string: “QSortFilterProxyModel” does not consider fixed strings, but instead works based on inclusion. That, interpreted in terms of the application at hand, means e.g. searching for the string “m” in the name field yields all the results including the letter “m” somewhere in their name attribute. This property of “QSortFilterProxyModel” has been used as is for the search fields for ID, name, breed and bird species; but for the rest of the fields, the newly defined subclass called “SortFilterProxyModel” has been used in order to yield only results that are a complete match.

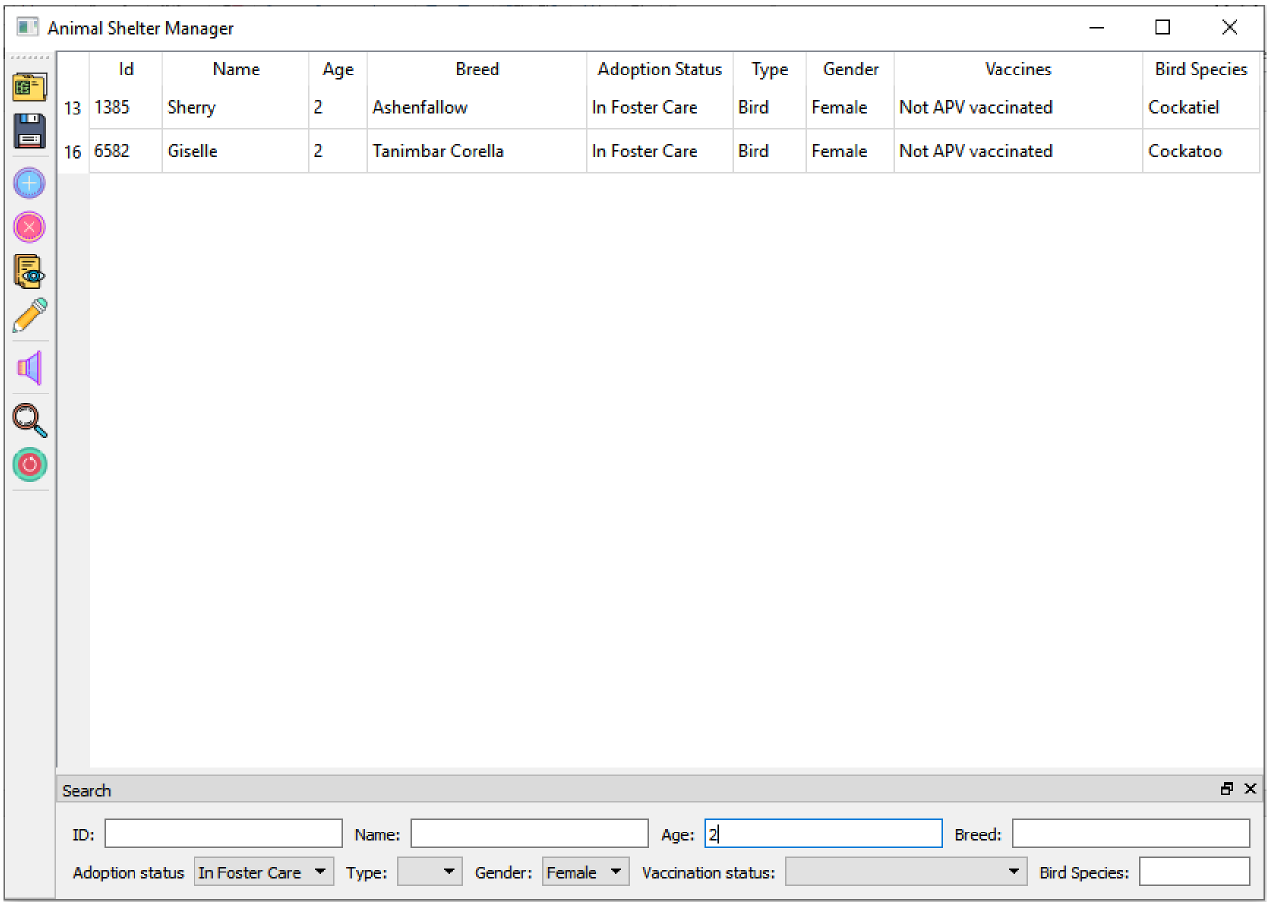


View Details

| | |
|---|---|
| ID: | 1385 |
| Name: | Sherry |
| Breed: | Ashenfallow |
| Age: | 2 |
| Adoption Status: | In Foster Care |
| Type: | Bird |
| Gender: | Female |
| Vaccination Status: | Not APV vaccinated |
| Bird Species: | Cockatiel |
| Monthly Cost for Care: | 60 € |

Close

*Figure 4: Viewing the details of an item*


Animal Shelter Manager

| | Id | Name | Age | Breed | Adoption Status | Type | Gender | Vaccines | Bird Species |
|---|---|---|---|---|---|---|---|---|---|
| 13 | 1385 | Sherry | 2 | Ashenfallow | In Foster Care | Bird | Female | Not APV vaccinated | Cockatiel |
| 16 | 6582 | Giselle | 2 | Tanimbar Corella | In Foster Care | Bird | Female | Not APV vaccinated | Cockatoo |

Search

ID: Name: Age: 2 Breed:

Adoption status: In Foster Care Type: Gender: Female Vaccination status: Bird Species:

*Figure 5: Sample search results*

## Qt Library

A subclass of the Qt class "QTableView" called "TableView" has been defined to hold the main view of the application. This view depends upon a custom "QSortFilterProxyModel" that acts as a bridge to access the model created by the class "modeladapter" which is a subclass of the Qt class "QAbstractTableModel". "QSortFilterProxyModel" class also lets us search the objects representing shelter animals, based on any subset of their attributes, and the results will be correctly shown in the "TableView" mentioned before.

## Serialization

In order to make the saving and opening of files representing containers possible, The JSON file format has been used as a standalone serialization mechanism. The reason for this choice is mainly the fact of JSON being very intuitive and easy for humans to interpret while also being simpler and taking up less space in comparison to XML. Moreover, as the Qt documentation puts it, *"the JSON support in Qt provides an easy to use C++ API to parse, modify and save JSON data."*

## Compilation

The file qmake project file “Proj.pro” has been added to the files turned in, as it will be different from the file produced by the command qmake -project. The reason for this difference is the use of some C++11 syntax and keywords.

Besides the project file, a sample database file called “SampleDB.json” has been added, which can be used to test and run the program.

## Hours of Work

| Analysis of the project and requirements | 2h |
|---|---|
| Studying the model design | 3h |
| Studying the GUI design | 3h |
| Learning the Qt Library | 12h |
| Coding the model | 8h |
| Coding the GUI | 15h |
| Testing and debugging | 9h |
| Writing the essay | 3h |
| Total | 55h |