Pet Adoption Shelter Final Report

Jacob Munoz

Description

A program that represents a functioning animal shelter. The purpose of the project was to create a standalone shelter management system. Users are able to browse all available pets and sort by animal type(dog, cat etc.) Users are able to put pets on hold for a visit or adopt any of the pets that are currently available for adoption. Users can also add pets they wish to be added to the adoption center. These requests must be reviewed and accepted/declined by an admin of the shelter. Admins can also manage the directories(add, remove, update) within the shelter including the master pet directory(contains all pets that have come into the shelter) and the pet drop-off request directory.

Features

ID	Requirement Title	Description	Priority
1	Admin Add Pet	Admins should be able to add pets to the directory.	Must have
2	Admin Remove Pet	Admins should be able to remove pets from the directory.	Must have
3	Admin Update Pet	Admins should be able to update pet information(weight, age, availability, etc.)	High
4	Admin Adoption Review	Admins should be able to accept/reject adoption requests(Update status to Adopted)	Must have
5	User Drop Off Request	Users should be able to add pets to directory for adoption	Must have
6	User View Available Pets	Users should be able to view pets that are up for adoption	Must have
7	User Reserve Pets	Users should be able to put pets on reserve to visit(Update status to On-Hold)	Must have
8	User Adoption Request	Users should be able to request to adopt pets(Update status to Adopted)	Must have
9	User Sort Search	Users should be able to sort available pets(Cats, dogs, etc.)	Should have
10	Admin Sign-up	People should be able to create an admin account(Trivial, no verification)	Stretch - Must have

Implemented:

1. Admin Add Pet[1]
2. Admin Remove Pet[2]
3. Admin Adoption Review[4]
4. User Drop-Off Request[5]
5. User View Available Pets[6]
6. User Reserve Pet[7]
7. User Adoption Request[8]
8. User Sort Search[9]

Not Implemented:

1. Admin Update Pet[3]
    a. Admins are able to update pet status however other information is not currently updateable.
2. Admin Sign-Up[10]
    a. Program currently has a single default admin object but no capability to create more.

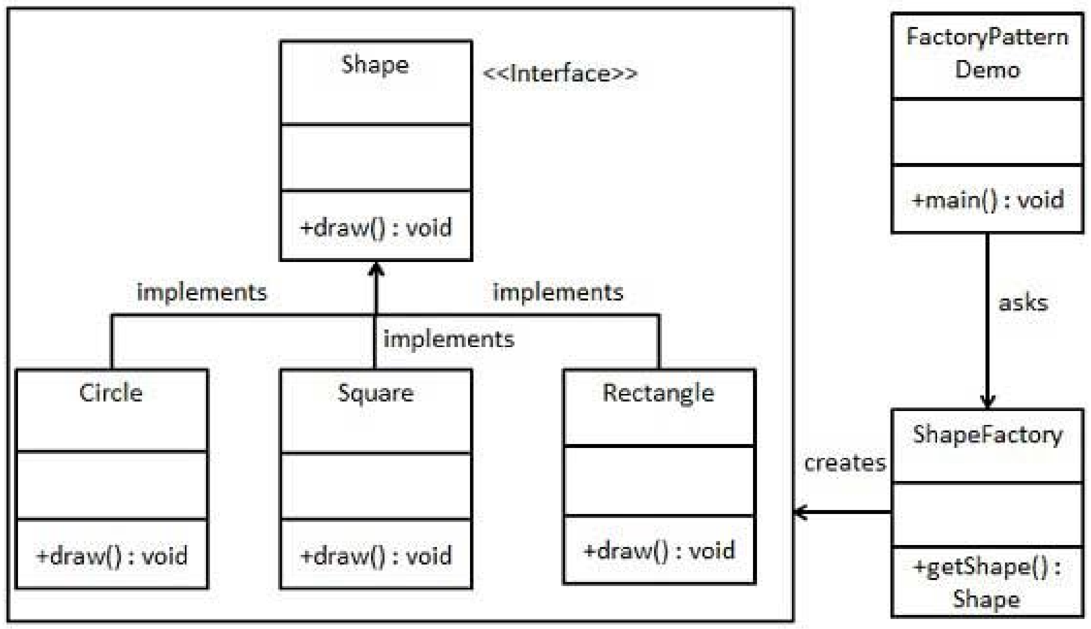
Class Diagram

I cannot find a way to display the entire diagram and make it viewable, this document simply is not big enough. Link: https://www.lucidchart.com/invitations/accept/7901e64a-ada1-4084-9a1d-41cf1de9ef83

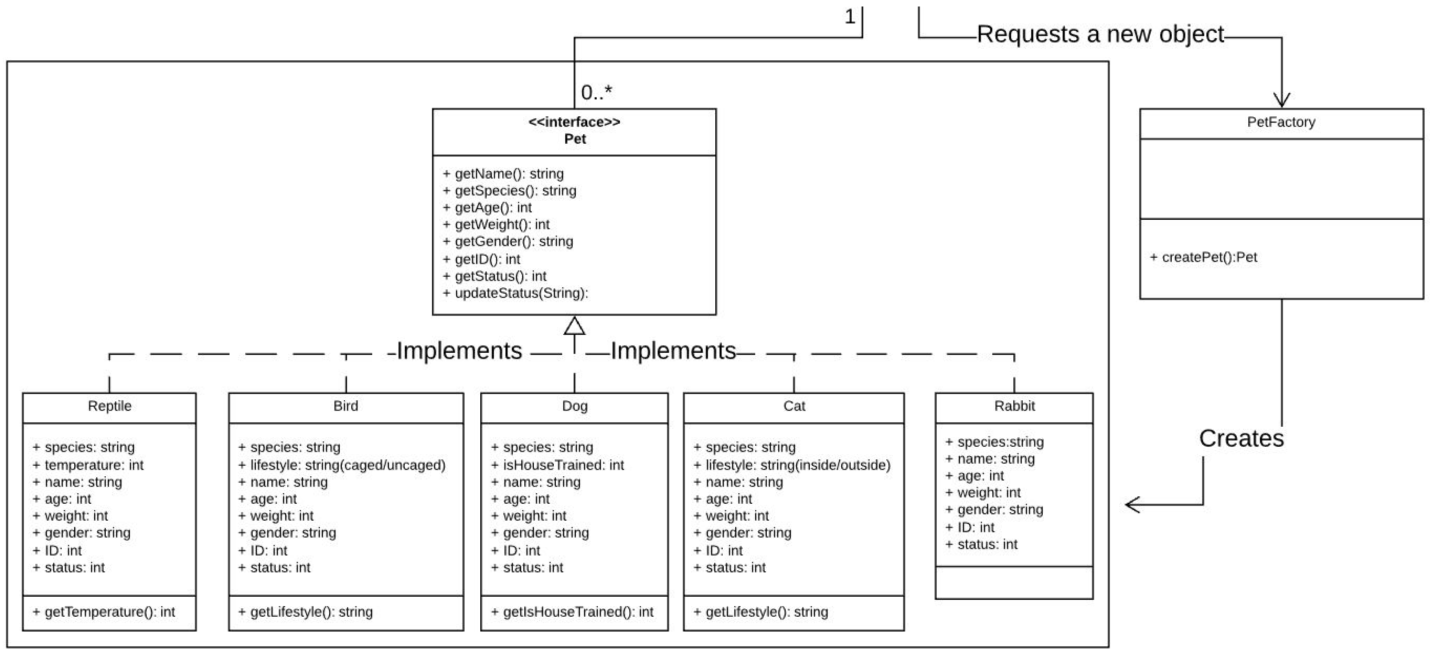
Many changes were made to the diagram as the project evolved and I learned however, the upfront planning made the initial implementation significantly easier. I have added a controller that handles the page interactions and flow of the program. The Shelter, Admin, and Customer classes were all heavily revised with additions to their class methods/attributes as the project progressed. The factory design pattern class was added to encapsulate the Pet classes/interface. Finally, some notation fixes and arrow corrections.

Design Pattern(Factory)

Example:



Implementation:



Reasoning:

I implemented the factory design pattern by creating an interface for Pet classes to implement. This can be extended to any number of pets you wish to add to the shelter. I then created a PetFactory class that implements createPet that will create the appropriate pet based on user inputted data(type of animal) along with the pet information provided. The factory returns the Pet object that can be placed in the appropriate directory by the system. I chose the factory design pattern because the creation of Pet objects is fundamental to the program. Without any pets the rest of program is useless. Having a class that can easily create any type of Pet object with unique information is extremely helpful. The creation of a Pet becomes trivial and the program can be easily extended because of the factory.

What I've Learned

Designing upfront makes the implementation significantly easier. Having put in the thought initially made it so that when I actually started writing code I already had a roadmap of what needed to be done. I also have a learned a lot about encapsulating methods and breaking up the work. Throughout my coding I would take the time to step back and refactor any methods that were performing more than one action. This proved to be extremely helpful when performing various actions because I could easily reuse code that had already been written. It also made the code easier to read and understand because there are a lot of small methods rather than a few monster methods. Additionally this project has made me more comfortable with object oriented programming as a whole. The idea of creating classes and using their attributes/methods in a system is something I am more confident in having completed this project. Overall I would say I benefited greatly from this semester project!