# CS/SE 1337 – Homework 6 – Animal Shelter

**Dr. Doug DeGroot's C++ Classes**
**Due**: Sunday, May 2, 2020, by midnight (note, all Extra Credit programs due at this time, too!)

**How to submit**: Upload your entire project in a zip file to the eLearning site. The project must compile on either Codeblocks or MS Visual Studio 2015 (or later). Include your name in the file/project name, like so:

HW6-CS1337-Jimi-Hendrix.zip

Include all your project files plus screenshot or PDF of your program's entire output.

**Maximum Number of Points:** 100

**Note:** I have modified this homework to the best of my ability to make it doable in the shorter time period we have this semester. So I may have some inconsistencies in this document; I apologize if I do. Please let me know if you spot any problems.

**Objectives**

The goal of this homework is to get experience working with C++ classes, polymorphism, and inheritance. It might seem daunting at first blush, but really it is fairly simple. I have uploaded to the eLearning Codes/Classes-and-Pet folder a sample C++ code that you can study and start with. It's called Cats-and-Animals-2.cpp.

The larger goal for this homework is to write a small database system for an Animal Shelter. This is a no-kill shelter like the Operations Kindness shelter just west of the Addition Airport. You will accept animal "donations" to the shelter, but mostly only dogs and cats. (But yes, there may be the occasional hamster, rabbit, or other non-dog, non-cat animal donation.) At present, all we need to do is write the skeleton of a database that will track all the animals in the shelter. We will add animals to the database and print reports based on the database. (Clearly there is a lot more functionality we could add to this program, but for now, we will just do the above.)

**Approach:**

1. Use three classes for this project: Animal, Cat, and Dog. Cat and Dog will be subclasses of Animal.
2. Each animal in the database will have the following attributes (and maybe more, if I add them).
    a. Type (dog, cat, hamster, etc.)
    b. Name
    c. Age
    d. Weight
    e. Breed
    f. Color
    g. Health
3. There is a data file that you will need to read and parse to create the animal objects and set their data attributes in the database. An example file input will look *something* like this:

```
AnimalType,Name,Age,Weight,Breed,Color,Health,Sound
cat,Morris,9,3,mixed,yellow,good,meow
cat,Mittens,1,,Calico,brown and white,good,Mew mew
cat,Junior,1,2,Tabby,black,needs shots,Meow
chipmunk,Chippy,2,1,white and gold,good,sniff sniff
dog,Priss,,3,Heinz,white,good,bark
cat,Charcoal,1,2,Siamese,white and yellow,good
```

The data file will be a CSV (comma-separated-values) file saved from an Excel file; thus, it will be a plain text file with commas used as field separators. Note as in the above example that some of the values might be empty, meaning "unknown." Each line of the data file will represent one animal – a cat, a dog, or some other animal. Dogs and Cats will be created using their own Class definitions; other animals will simply be Animals and only animals. The final data file that you are to use will be uploaded (soon) to the eLearning site and called

**AnimalShelterData-03.csv**

4. You will need to define separate classes for Cats, Dogs, and Animals. Cats and Dogs will inherit from Animals.
5. Keep track of the number of animals created of each type; thus you will know how many cats have been created, how many dogs, and how many total animals (you don't have to count "other" animals since you can always compute that with

   Nbr "others" created = nbr Animals created – (nbr Dogs created + nbr Cats created)

   However, if you want to, you can indeed keep count of all "other" animals. You can even create more classed than just Dogs and Cats if you want to.
6. Use a separate **static variable** within each class to count the objects created of that type, and create a method to keep count of the number of objects and to assign a unique number to every newly created object. For example, there will be both a numberOfCats member and a myCatNumber member in the Cat class.
7. Create a .h file and a .cpp file for each of the three classes and link them into the main.cpp file.
8. Create and maintain three separate vectors of objects – one each for Animal, Cat, and Dog. Store every animal in the Animal vector (the Animal vector will store all three types of objects – animals, cats, and dogs).
9. In addition, keep a separate vector of Cats and one of Dogs. Store all cats in the Cat vector and all dogs in the Dog vector (that's in addition to storing them in the Animal vector).
10. Read the data input file line by line, collecting the animal attributes. Create an object of the appropriate type/s (animal, dog, cat) and set all the attributes to the values you read from the file.
11. For each class, create an Introduction method that will have the animal/cat/dog speak and then say its name, age, color, health, etc.
12. Once you've read the entire database and created all the objects and vectors, create four reports (yes, you can print these to the console and then PDF them, or you can print to a .txt file – your choice):
    a. Report 1: total number of animals created, number of cats created, number of dogs created
    b. Report 2: A Cat report. Have each cat in the Cat vector introduce itself (by "speaking", giving its name, age, etc.)
    c. Report 3: A Dog report: same as above but reporting all Dogs.
    d. Report 4: An Adoptable Animals report - All animals that are adoptable (health is good).

**Annotations for the code:**

1. The main function can be at either the beginning or the end of the program. I don't care which.
2. Add comments at the top of your program file to include your name, the name of the program, and any relevant notes on how your design works when executed.
3. Add a change log in your comments that list the major changes you make to your logic and when – nothing too terribly detailed, but a list of breadcrumbs that will remind you and others of what you've done as your program becomes more sophisticated and/or nearly complete.
4. Point out (in the comments at the top of your program) any special features or techniques you added using a comment saying something like "// Special Features:"

5. Comment your code effectively, as we discussed in class. Use descriptive variable names everywhere so that the code becomes as self-documenting as possible. Use additional commentary only to improve readability and comprehensibility by other people.
6. You absolutely MUST use consistent indentation and coding styles throughout the program. Failure to do so will result in a loss of three points.
7. If the program does not work at all, or works incorrectly, at least 15 points will be deducted.
8. Use *great* modularization techniques. -5 points for sloppy/obtuse code.
9. No late submissions will be accepted since this is the end of the semester and final grades must be submitted on time. Please meet the deadline.