Lab 12

Comp 11 - Summer Session — The Animal Kingdom

12.1 Description

Kingdom, Phylum, Class, Order, Family, Genus, Species. Or perhaps you better remember a mnemonic: Keep Ponds Clean Or Frogs Get Sick. What this is describing is a taxonomy for how to classify and organize animals in the animal kingdom.

In this lab we will be using inheritance to build a hierarchy of our own. One hierarchy that is demonstrated for you in the sample code, is with "Cat". You will add an additional one with "Dog" and then 3 different Dog species.

Our objectives are the following:

- Utilize Inheritance and have a Dog class that inherits from animal.
- Then three different dog species(GermanShepard, Retriever, Husky, etc.) of your choice must inherit from the Dog class and implement the proper methods.
- Each of the dogs should make a different noise so they can be distinguished.
- Proper usage of virtual should be used, for when you want to enforce what must be implemented.
- Proper usage of private/public should be used, such that you encapsulate data that should and should not be shared amongst inherited classes.

12.2 Files

You may use the following code to help get you started. If you find it easier, you may (and are encouraged) to break this project into separate files, just remember to submit them all!

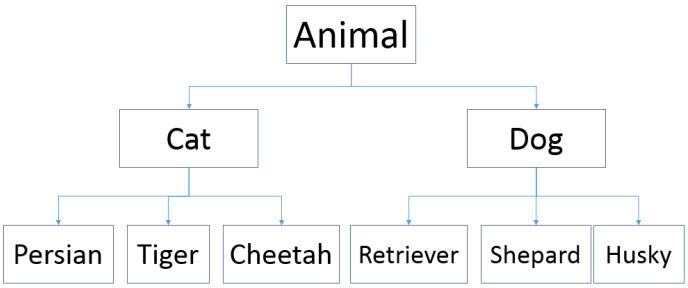

Figure 12.1: Class inheritance diagram of what you will implement

```cpp
#include <iostream>
#include <vector>
#include <string>

// ================================================
// ==Base Class from which all animals inherit from==
// ================================================
class Animal{
public:
    virtual ~Animal() { }

    void getName(){
        std::cout << "Hello, my name is: " << name << "\n";
    }

    void setName(std::string _name) {
        name = _name;
    }

private:
    std::string name;
};

// ================================================
// ==    All Cat's use Animal as a Base Class      ==
// ================================================
class Cat : public Animal{
public:
    virtual ~Cat(){};

    virtual void makeNoise(){};

    // If I do not redine this later, then
    // this member function is used by default
    virtual void getTopSpeed(){
        std::cout << "My top speed is: " << topSpeed << "\n";
    }
protected:
    int topSpeed; // In mph

private:
};
```

```
43
44 // ================================================
45 // == The Following 3 Classes all inherit from Cat ==
46 // ================================================
47 class Persian : public Cat{
48 public:
49
50     Persian(){
51         setName("Fluffy");
52         topSpeed = 30;
53     }
54
55     void makeNoise(){
56         std::cout << "meow!\n";
57     }
58
59 private:
60
61 };
62
63 class Tiger : public Cat{
64 public:
65
66     Tiger(){
67         setName("Shere Khan");
68         topSpeed = 45;
69     }
70
71     void makeNoise(){
72         std::cout << "Roar!!\n";
73     }
74
75 private:
76
77 };
78
79 class Cheetah : public Cat{
80 public:
81
82     Cheetah(){
83         setName("Duma");
84         topSpeed = 70;
85     }
86
87     void makeNoise(){
88         std::cout << "Vroom Vroom\n";
89     }
90
91 private:
92
93 };
94
95
96
97 int main(){
98
99     // Create three different cats
```

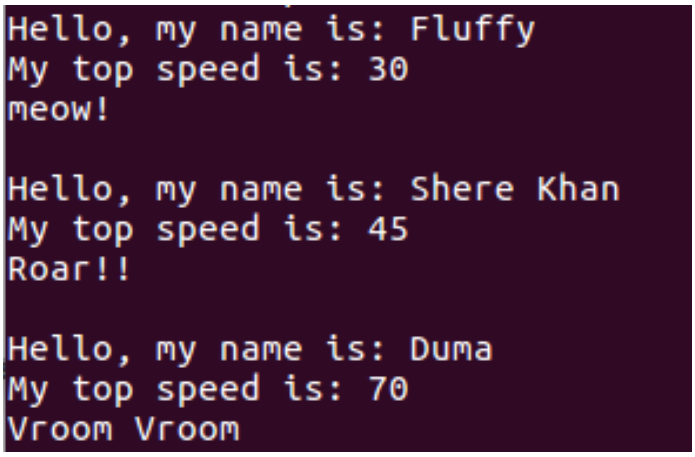

Figure 12.2: Sample output with the Cats, now you must repeat the same with dogs!

```
100     // They are each instantiated with different
101     // typse, but they all inherit from the same base class
102     Persian meow1;
103     Tiger   meow2;
104     Cheetah meow3;
105
106     meow1.getName();
107     meow1.getTopSpeed();
108     meow1.makeNoise();
109     std::cout << '\n';
110
111     meow2.getName();
112     meow2.getTopSpeed();
113     meow2.makeNoise();
114     std::cout << '\n';
115
116     meow3.getName();
117     meow3.getTopSpeed();
118     meow3.makeNoise();
119     std::cout << '\n';
120
121     return 0;
122 }
```

Listing 12.1: main.cpp

12.3 Output

12.4 Refresher

Revisit the lecture slides for some examples of concepts demonstrated.

12.5 Submission

```
provide comp11 lab12 all_of_your_cpp_files README
```

Listing 12.2: Submit Assignment

12.6 Going Further

Did you enjoy this lab? Want to try out some additional commands to go further?

- Separate the interface from the implementation where it makes sense.
- How would you handle errors in this lab?