Katerina Betts and Donald Hutchinson

Professor Charles Zimmer

Data Structures

13 February 2018

Lab 4: Inheritance, Polymorphism, and Abstract Classes

In this lab, as listed in the title, the ideas of inheritance, polymorphism and abstract classes as well as sub-ideas such as virtual functions. Some older concepts were explored in the testing file, or Task 4, were switch statements, loops, and if/else statements. When the lab comes to inheritance, the Fish and Horse classes that are derived from the Animal class are prime examples. Those two classes take the public members of the Animal class to use in each of their respective classes due to the fact that they are derived from the Animal class. Polymorphism was utilized in this lab when the classes of Fish and Horse are passed into certain test cases in the last task that treats them like the Animal class. The Animal class itself is an abstract class because it contains virtual functions utilized by the derived classes Fish and Horse.

Each of these concepts is important to computer scientists and engineers in many ways. An abstract class can be used as a sort of base to create different classes with similar backgrounds that do similar things. This means in a career, an abstract class can be applied in many different ways, making it easier to write new classes without having to type as much code. Inheritance is a good skill in a career in a similar way. Because other classes can take member functions, or inherit, from a base class member, it utilizes time more efficiently and less code needs to be written. When it comes to polymorphism, the concept in this case utilizes inheritance and abstract classes. This is useful in a career where it would be easy to write a base class into a function that can also take in a derived class; thus, it would only require the creation of one such function in the code.

In Task 1, each member will be available in the derived classes because they are all declared as public. The animalMovement and animalEat functions are available in the derived classes if not overridden. In Task 2, the derived animalMovement function will be available in both instances of the derived class and instances of the derived class declared as the base class type because it is declared as a virtual function. The derived animalEat function is only available in instances of the derived class. In instances of the derived class declared as the base class, the base animalEat function is available because it is not a virtual function. In Task 3, it asks how the results compared to the expected results in Task 2. Due to the way the lab was completed, the expected results in Task 2 did not differ in any way from the actual results in Task 3.

Due to each task building off the other tasks, the classes and files were named after what they do. For example, the base class AnimalCross (files ANIMAL.h and ANIMAL.cpp) fits mostly into Task 1, and the derived classes FishInfo (files FISH.h and FISH.cpp) and HorseInfo (files HORSE.h and HORSE.cpp) fit mostly into Task 2. Task 3 is in fact named after the task because it is one file only used for testing the files from the other tasks.

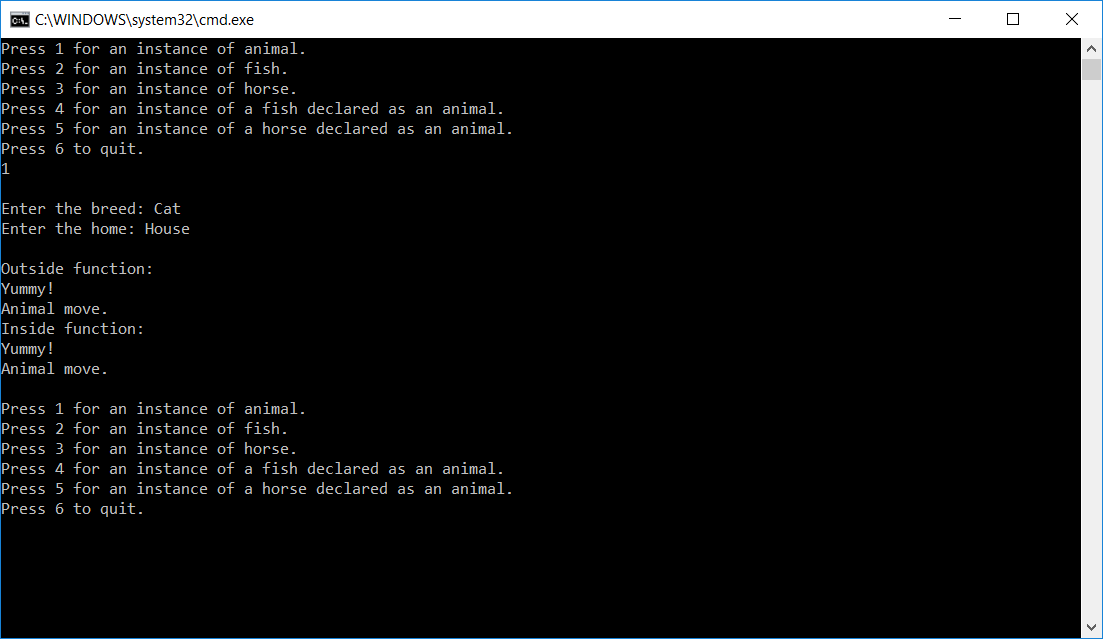


Figure 1: This figure shows the results when 1 is input into the test file. It will create an instance of an animal.

Figure 2: This figure shows the results when 2 is input into the test file. It will create an instance of a fish.

Figure 3: This figure shows the results when 3 is input into the test file. It will create an instance of a horse.

Figure 4: This figure shows the results when 4 is input into the test file. It will create an instance of a fish as an animal.

Figure 5: This figure shows the results when 5 is input into the test file. It will create an instance of a horse as an animal.

