**PET ADOPTION**

This is a problem using class inheritance and polymorphism. The New Friend pet adoption agency handles only dogs and cats. This is how they keep track of their stock.

We will have the following 5 classes:

Base Class: **Pet**

Derived from Pet: **Cat**, **Dog**

Derived from Dog: **DogAdoptee**

Derived from Cat : **CatAdoptee**

Use these UML diagrams to code your classes; note: the # symbol means Protected access.

|  |
| --- |
| Pet |
| #size : string //possible values are small, medium, large  #age : string //possible values are young, adult, senior  #gender : char //possible values are M or F |
| +Pet()  +Pet(sz : string, a : string, gen: char)  <setters and getters can be omitted or coded; they are optional>  +printInfo() : void //print pet’s information including size, age and gender (Hint: you will need an if  statement to print “male” when gender == ‘M’) |

|  |
| --- |
| Dog |
| #breed : string //this could be any breed or mixed breed  #houseTrained : bool //true or false |
| +Dog()  +Dog(sz : string, a : string, gen : char, brd : string, trained : bool)  +setBreed(brd : string) : void  +getBreed() : string  +setTrained(trained : bool) : void  +getTrained() : bool  +printInfo() : void //print Pet information and information about breed and whether house trained |

|  |
| --- |
| Cat |
| #hairLength : string //values can be long, medium or short  #otherCats : bool //true or false on whether it gets along with other cats in the house |
| +Cat()  +Cat(sz : string, a : string, gen : char, hair : string, other : bool)  +setHairLength(hair : string) : void  +getHairLength() : string  +setOtherCats(other : bool) : void  +getOtherCats() : bool  +printInfo() : void //print Pet information along with hair length and whether it gets along with  other cats |

|  |
| --- |
| DogAdoptee |
| -name : string  -activityLevel : string //values can be high, medium, low |
| DogAdoptee()  DogAdoptee(sz : string, a : string, gen : char, brd : string, trained : bool, nm : string, activity : string)  +setName(nm : string) : void  +getName() : string  +setActivity(activity : string) : void  +getActivity() : string  +printInfo() : void //print all Pet and Dog information along with name and activity level |

|  |
| --- |
| CatAdoptee |
| -name : string  -personality : string //values can be cuddly, playful or shy |
| +CatAdoptee()  +CatAdoptee(sz : string, a : string, gen : char, hair : string, other : bool, nm : string, pers : string)  +setName(nm : string) : void  +getName() : string  +setPersonality(pers : string) : void  +getPersonality() : string  +printInfo() : void //print all Pet and Cat information along with name and personality |

Code all five classes using header files and putting member function code into separate .cpp files. Be sure to indicate inheritance when you code the header files for Dog, Cat, DogAdoptee and CatAdoptee. Be sure to include the proper header files where needed.

NOTES:

* The constructors of your derived classes should include the parents’ attributes and the current class attributes. Use the parent’s constructor for the parent’s attributes as shown in the Exercise.
* Your printInfo() methods should call the parent’s printInfo() and add any unique information. No redundant print statements allowed! Use inheritance.
* When printing information on booleans, use an if statement:

if (houseTrained)

cout << “This dog is housetrained” << endl;

else

cout << “This dog is not housetrained” << endl;

In the main function:

Create a DogAdoptee named “Pugster” that is a medium activity pug who is housetrained. He is considered a small, young male. Print all Pugster’s information with one printInfo() call using the DogAdoptee object.

Create a CatAdoptee named “Holly” that is a cuddly long-haired cat that gets along with other cats well. Holly is a small adult female cat. Print all Holly’s information with one printInfo() call.

Here is some example output; your should be similar but need not be identical:

