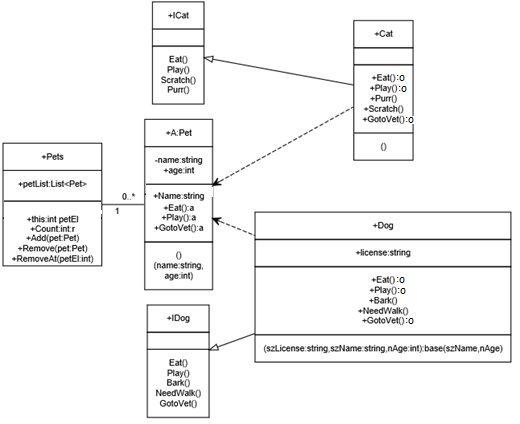
# PE13 – More Classes

Create a console application called PetApp with the following schUML and design:



A few notes:

1. In the Pets class, be sure to declare petList as: **List<Pet> petList = new List<Pet>();**
2. In the Pets class, "this:int petEl" is an indexer property based on the int "petEl". We only implemented a string-based indexer with a SortedList. Here's what it looks like for a List<>. Use this code:

public Pet this[int nPetEl]

{

get

{

Pet returnVal;

try

{

returnVal = (Pet)petList[nPetEl];

}

catch

{

returnVal = null;

}

return (returnVal);

}

set

{

// if the index is less than the number of list elements

if( nPetEl < petList.Count )

{

// update the existing value at that index

petList[nPetEl] = value;

}

else

{

// add the value to the list

petList.Add(value);

}

}

}

1. In Pets, "Count:int:r" is a read-only property that returns (**get'**s) petList.Count.
2. Pets.Add(pet:Pet) should call petList.Add(pet)
3. Pets.Remove(pet:Pet) should call petList.Remove(pet)
4. Pets.RemoveAt(petEl:int) should call petList.RemoveAt(petEl)
5. Pet.Name is the readwrite property for the private name field. You will need to use that in your output to indicate which pet is responding.
6. If there is a 4th box for a class, then that is the list of required constructors. For example, Cat should only have the default constructor. Dog has one constructor which accepts license, name and age, and calls the Pet base constructor with name and age. Pet should have 2 constructors:

public Pet()

{

}

public Pet(string name, int age )

{

this.name = name;

this.age = age;

}

1. The member methods in Dog and Cat should call Console.WriteLine() with the pet's name and a phrase that suggests the activity. (Be creative!).

Your Main() function should do the following:

1. Create reference variables for the pets and interfaces:

Pet thisPet = null;

Dog dog = null;

Cat cat = null;

IDog iDog = null;

ICat iCat = null;

1. Create the list of pets:

Pets pets = new Pets();

1. Create the random number generator:

Random rand = new Random();

1. Implement a for() loop that iterates 50 times, and within the for() loop:
   1. Generate a number between 1 and 10. If it generates a 1, you bought a new animal and generate another random number with 50% chance of it being a dog or cat. Prompt the user for the fields as shown below. Note that the Dog() object must be passed the 3 fields (license, name and age) in the constructor.

// 1 in 10 chance of adding an animal

if( rand.Next(1, 11) == 1 )

{

if( rand.Next(0, 2) == 0 )

{

// add a dog

}

else

{

// else add a cat

}

}

else

{

// choose a random pet from pets and choose a random activity for the pet to do  
}

* 1. Else if it did not generate a 1, then set thisPet to a random pet from the pets List<> based on a random number between 0 and pets.Count. Note that if there were no pets added yet, this will return null. If null was returned, then continue to the top of the for() loop.
  2. If a valid pet was returned, set your interface variable (iDog or iCat) to thisPet (based on thisPet.GetType()) and randomly call one of the member methods of the interface.

It should have output similar to the following:

You bought a dog!

*{prompt for the dog's name, age and license id and add it to the pets List}*

Dog's Name => Puddles

Age => 3

License => 345

*{it continues through the for() loop with a 10% chance of buying another pet, therefore 90% of the time it will choose an existing pet and have them do a random action}*

Puddles: Yummy, I will eat anything!

Puddles: Woof woof!

Puddles: Woof woof, I need to go out.

Puddles: Yummy, I will eat anything!

Puddles: Yummy, I will eat anything!

Puddles: Whimper, whimper, no vet!

Puddles: Woof woof!

Puddles: Throw the ball, throw the ball!

Puddles: Woof woof!

Puddles: Woof woof, I need to go out.

Puddles: Woof woof!

Puddles: Woof woof, I need to go out.

Puddles: Throw the ball, throw the ball!

Puddles: Woof woof!

Puddles: Whimper, whimper, no vet!

Puddles: Whimper, whimper, no vet!

Puddles: Whimper, whimper, no vet!

*{a cat was chosen this time, so prompt for the cat's name and age and add it to the pets List}*

You bought a cat!

Cat's Name => Cleo

Age => 2

Puddles: Woof woof, I need to go out.

Cleo: Hiss!

Cleo: Where's that mouse...

Cleo: Yuck, I don't like that!

Cleo: Hiss!

Puddles: Throw the ball, throw the ball!

Cleo: purrrrrrrrrrrrrrrrrrrr...

Puddles: Whimper, whimper, no vet!

Puddles: Yummy, I will eat anything!

Cleo: purrrrrrrrrrrrrrrrrrrr...

Cleo: purrrrrrrrrrrrrrrrrrrr...

Cleo: Where's that mouse...

Puddles: Woof woof!

Puddles: Yummy, I will eat anything!

Cleo: purrrrrrrrrrrrrrrrrrrr...

Cleo: Hiss!

Cleo: purrrrrrrrrrrrrrrrrrrr...

Cleo: Where's that mouse...

Cleo: Yuck, I don't like that!

Cleo: purrrrrrrrrrrrrrrrrrrr...

Puddles: Throw the ball, throw the ball!

Puddles: Yummy, I will eat anything!

Puddles: Yummy, I will eat anything!

Puddles: Throw the ball, throw the ball!

Puddles: Throw the ball, throw the ball!

Cleo: Yuck, I don't like that!

Puddles: Woof woof, I need to go out.

Cleo: Yuck, I don't like that!

Cleo: Yuck, I don't like that!

Puddles: Woof woof, I need to go out.

Puddles: Woof woof!

## Submission

Submit the GitHub URL of the VS .NET solution to the corresponding MyCourses dropbox.

<https://github.com/ndw1117/myIGME-201/tree/main/PetApp>