Week 2-04 Tutorial

This tutorial should be completed before the next lecture.

Task 1: Draw UML class diagrams (Beginner)

Using Microsoft Visio, draw UML class diagrams for classes that will satisfy the problem specification given below. Show the inheritance relationships between the classes.

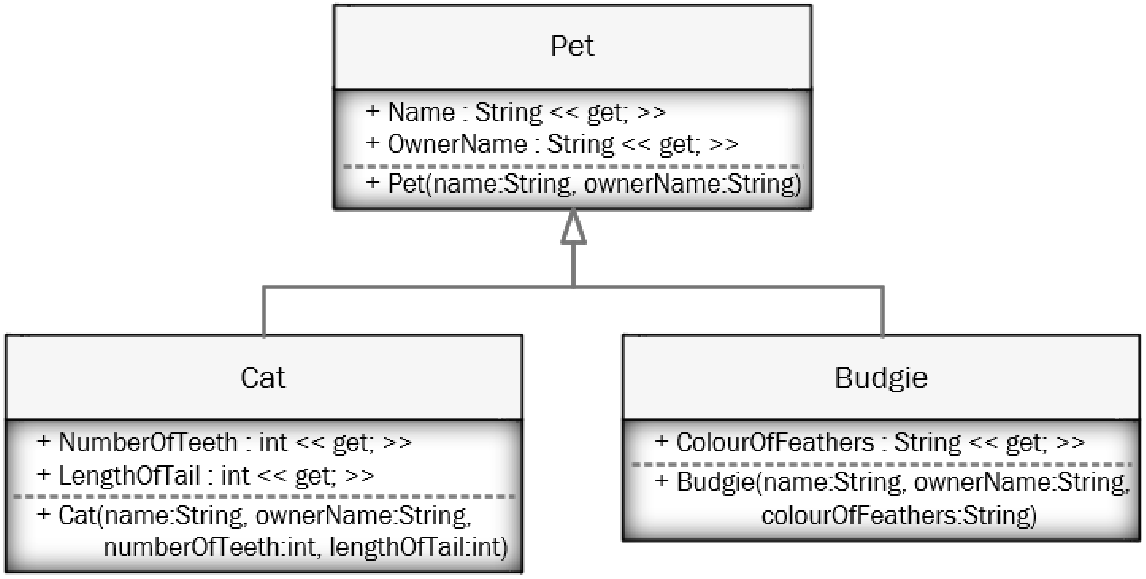
Problem specification

A publishing house prints and distributes two types of publication: books and journals. Each book has a title, an author, an edition number, and an ISBN. Each journal has a title, a volume number, an issue number, and an ISBN.

A programmer is to be allocated the task of writing an Object-Oriented application. Your task is to design for the programmer the classes needed for this problem specification.

Task 2: Convert UML class diagrams to C# code (Beginner)

Convert the UML class diagrams shown below into C# code. Be precise.

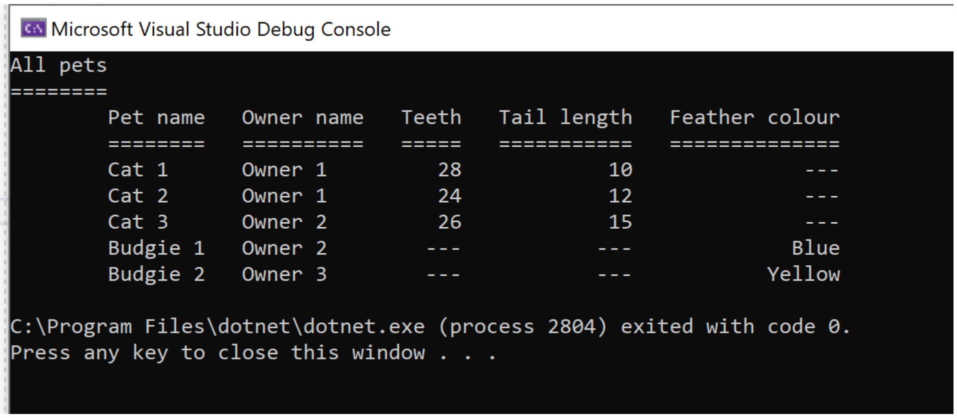

Pet
+ Name : String << get; >>
+ OwnerName : String << get; >>
+ Pet(name:String, ownerName:String)
Cat
+ NumberOfTeeth : int << get; >>
+ LengthOfTail : int << get; >>
+ Cat(name:String, ownerName:String, numberOfTeeth:int, lengthOfTail:int)
Budgie
+ ColourOfFeathers : String << get; >>
+ Budgie(name:String, ownerName:String, colourOfFeathers:String)

Hint: Remember to ensure that the constructor methods in the subclasses call the base-class's constructor method.

In the class called `Program`, the `Main()` method should:

1. create three `Cat` objects and two `Budgie` objects;

2. output the details of all pets to the console window in a tabulated format, similar to the illustration below.



```
All pets
========
        Pet name   Owner name   Teeth   Tail length   Feather colour
        ========   ==========   =====   ===========   ==============
        Cat 1      Owner 1         28            10              ---
        Cat 2      Owner 1         24            12              ---
        Cat 3      Owner 2         26            15              ---
        Budgie 1   Owner 2        ---           ---             Blue
        Budgie 2   Owner 3        ---           ---           Yellow

C:\Program Files\dotnet\dotnet.exe (process 2804) exited with code 0.
Press any key to close this window . . .
```

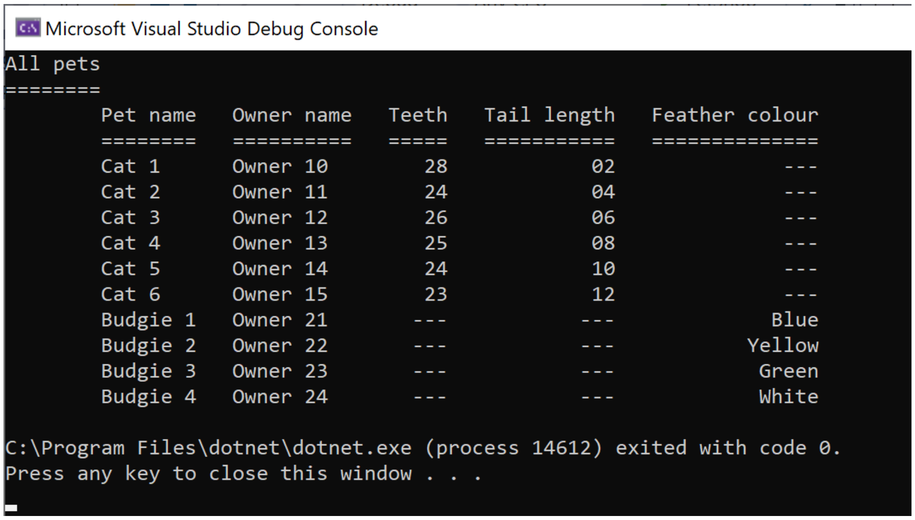
Task 3: Polymorphic pet shop (Intermediate)

Make a copy of your C# project from Task 2 and modify it so that polymorphism is used.

Instead of storing the pets in separate variables, use an array.

In the `Program` class, add a method called `printPetDetails()`, which outputs to the console the contents of the `pets` array in a tabulated format, similar to the illustration below. The `pets` array must be passed to the method as a parameter. Use the `is` operator as shown in the `Main()` method, `Program` class, `PersonInheritance3` project from the lecture.

The `Main()` method should create six `Cat` objects and four `Budgie` objects, storing them all in the `pets` array, and then call the `printPetDetails()` method, passing the `pets` array as the parameter.



```
All pets
========
        Pet name   Owner name   Teeth   Tail length   Feather colour
        ========   ==========   =====   ===========   ==============
        Cat 1      Owner 10        28            02              ---
        Cat 2      Owner 11        24            04              ---
        Cat 3      Owner 12        26            06              ---
        Cat 4      Owner 13        25            08              ---
        Cat 5      Owner 14        24            10              ---
        Cat 6      Owner 15        23            12              ---
        Budgie 1   Owner 21       ---           ---             Blue
        Budgie 2   Owner 22       ---           ---           Yellow
        Budgie 3   Owner 23       ---           ---            Green
        Budgie 4   Owner 24       ---           ---            White

C:\Program Files\dotnet\dotnet.exe (process 14612) exited with code 0.
Press any key to close this window . . .
```