Exercise 2: Displaying Message Using Virtual Functions

This exercise requires you to implement the concept of virtual functions to display information about two base classes.

WE'LL COVER THE FOLLOWING

Problem Statement

You will first build **three** *classes*:

- `Mammal` (*parent class*)
- `Dog` (*derived class*)
- `Cat` (*derived class*)

`Dog` and `Cat` class will inherit from `Mammal`.

In the exercise you have to implement the following:

- `Mammal` class:
  - Has one `protected` **property**, `age`, of the mammal.
  - A **default** *constructor*
  - A *constructor* that takes the `age` of mammal as input and sets it.
  - The method `Eat()` that displays “Mammal eats food”.
  - Method `Speak()` that displays “Mammal speaks mammalian!!”.
  - Method `get_Age()` which returns the `age` of the mammal.
- `Dog` class:
  - Inherits all the *members* from `Mammal` class.
  - It’s constructor calls on the Mammal class constructor with parameters.

- Implement all `member` functions of `Mammal` class for `Dog` class.
- `Eat()` should display **“Dog eats meat”**.
- `Speak()` should display **“Dog barks: ruff! ruff!”**.
- `get_Age()` should *return* Dog’s **age**.

- `Cat` class:
  - Inherits all the *members* from `Mammal` class.
  - It’s constructor calls on the Mammal class constructor with parameters.
  - Implement all methods of `Mammal` class for `Cat` class.
  - `Eat()` should display **“Cat eats meat”**.
  - `Speak()` should display **“Cat meows: Meow! Meow!”**.
  - `get_Age()` should *return* Cat’s **age**.

> **Hint**: Think along the direction of `virtual` methods and their use to implement the **same** *method* for **different** *classes* separately.

Here’s a sample result which you should get.

Input:

```
Dog(5);
Cat(4);
```

Expected Output:

```
Dog eats meat
Dog barks: ruff! ruff!
Dog's age is: 5
Cat eats meat
Cat meows: Meow! Meow!
Cat's age is: 4
```

Write your code below. It is recommended that you try solving the exercise yourself before viewing the solution.

Good Luck!

Exercise | Solution

```
using System;

class Mammal
{
    //define protected and public members here
}

//define the base class named "Dog" here


//define the base class "Cat" here

class VirtualExercise {
  static void Main() {

   //make object of Mammal class
  //making object of child class Dog
  //making object of child class Cat

  Console.WriteLine("Calling Dog class functions");
  //call Eat and Speak methods here for the Dog object

  Console.WriteLine("Dog's age is: "); //displaying the age by calling the get_Age() method h

  Console.WriteLine("Calling Cat class functions");
  //call Eat and Speak methods here for the Cat object

  Console.WriteLine("Cat's age is: "); //displaying the age by calling the get_Age() method h

  }
}
```