

Inheritance

Inheritance in C#

- *Inheritance* is the ability to derive new classes from existing ones. A derived class ("subclass") inherits the instance variables and methods of the base class ("parent class"), and may add new instance variables and methods.
- Inheritance defines a hierarchical relationship among classes wherein one class shares the attributes and methods defined in one or more classes.
- Inheritance is a relationship among classes in which one class shares the structure and behavior of another. A subclass inherits from a base class.

Its a ability of the derived class that's a new class to extent to information of the base class, Base class is something like a parent class

For Example :- a phone has came into market it is fabulous but its too costly so i want to take the look and feel

so i can borrow the look and feel of some other phone with same color same big screen size

Whenever i want to duplicate my phone i can copy

now what you want to borrow that is something which you want to question

now when i want to buy new phone so i only want to borrow the look and feel of i phone 13 its only the Abstraction which i want to borrow, if so then what you can do is We just create a instance of it and then we borrow it

to get same look and feel i'll just inherit it

when i inherit what happens all the function parent has, we'll have it in child but internally i can put all the chinese chip to make it cheaper so internally it will be different

But if i'm not bothered of look and feel but i want to copy inly the internal details so then i'll create the instance of these and i can access that information

So it all depends on the design that what you want to borrow

With inheritance you can borrow all the functionality, all the features of the class

Now we'll see same program in 6 different ways

1. Take a Example we have a particular client who want to create a PetStore

there where lot of dogs in the pet store so i want to maintain the dog information

PetStore.cs* X ConsoleApp*

ConsoleApp

ConsoleApp.Dog

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;

7 namespace ConsoleApp
8 {
9     0 references
10    class Dog
11    {
12    }
13
14    0 references
15    class PetStore
16    {
17    }
18}
```

So I'll just say dog,
I now what are the behaviours of the dog

walk
bark
Eat

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'ConsoleApp' (2 of 2 pr)

ConsoleApp

- Properties
- References
- CollectionsDemo
- AccessDemo.cs
- App.config
- ArrayDemo1.cs
- ArrayDemo2.cs
- BreakContinueDemo.cs
- Calc.cs
- ClassDemo.cs
- CommandLineDemo.cs
- ConvertType.cs
- EnumDemo.cs
- GoToDemo.cs
- IfDemo1.cs
- IfDemo2.cs
- IfDemo3.cs
- IncDemo.cs
- InOutDemo.cs
- InOutDemo1.cs

Diagnostic Tools Properties Git Changes Notifications

PetStore.cs* X ConsoleApp*

ConsoleApp

ConsoleApp.Dog

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      class Dog
10     {
11         void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15
16         void Sound()
17         {
18             Console.WriteLine("Bow Bow...");
19         }
20     }
}
```

```
15
16     0 references
17     void Sound()
18     {
19         Console.WriteLine("Bow Bow...");
20     }
21
22     0 references
23     void Eat()
24     {
25         Console.WriteLine("Veg and Non-veg");I
26     }
27
28     0 references
29     class PetStore
30     {
31 }
```

```
    void Eat()
    {
        Console.WriteLine("Veg and Non-veg");
    }
}

0 references
class PetStore
{
    0 references
    static void Main()
    {
        Dog dog = new Dog();
    }
}
```

So what are the things we can access by creating instance of dog

```
        Console.WriteLine("Veg and Non-veg");
    }

0 references
class PetShop
{
    static void Main()
    {
        Dog dog = new Dog();
        dog.~
```

You can see i cannot
access any of these

Why?

Because we haven't use any
access modifire while creating the method
so by default they are private so
we have to make them public

```
7  namespace ConsoleApp
8  {
9      class Dog
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15
16         public void Sound()
17         {
18             Console.WriteLine("Bow Bow...");
19         }
20
21         public void Eat()
22         {
23             Console.WriteLine("Veg and Non-veg");
24         }
25     }
26 }
```

now you can see when i say dog. i'll get all the methods listed

0 references

```
class PetStore
```

```
{
```

0 references

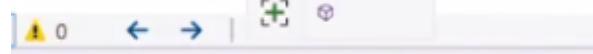
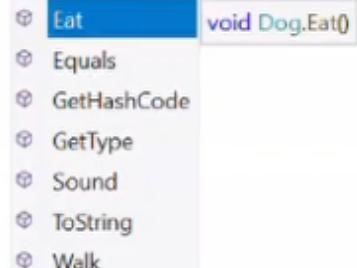
```
    static void Main()
```

```
{
```

```
        Dog dog = new Dog();
```

```
        dog..
```

```
}
```



The screenshot shows the Visual Studio IDE with the code editor open to `PetStore.cs`. The code defines a `Dog` class with `Eat()`, `Sound()`, and `Walk()` methods, and a `PetStore` class with a `Main()` method that creates a `Dog` instance and calls its methods. A terminal window at the bottom shows the program's output: "Veg and Non-veg", "Bow Bow...", "Use 4 legs to walk", and "Press any key to continue . . .".

```
20
21     1 reference
22     public void Eat()
23     {
24         Console.WriteLine("Veg and Non-veg");
25     }
26
27     0 references
28     class PetStore
29     {
30         0 references
31         static void Main()
32         {
33             Dog dog = new Dog();
34             dog.Eat();    I
35             dog.Sound();
36             dog.Walk();
37         }
38     }
```

so in the 1st program what we have done ?

We have seen a crowd for dog so we are just creating a class called dog and dog will have certain features like walk sound and eat, so when i went to doctor and showed them, So they are quite happy about this but they say shashi you know what with the Dog even get the cat also, so cat also does the same

So now we are in the second sprint and we want to enhance now so we are going with the class called cat

So when i want to doctor and ask like what are the feature my cat will have
SO doctor said the same thing it can walk sound eat

The screenshot shows a code editor window with the following code:

```
PetStore.cs  X  ConsoleApp
ConsoleApp
ConsoleApp.Cat

class Dog...
class Cat
{
    public void Walk()
    {
        Console.WriteLine("Use 4 legs");
    }

    public void Sound()
    {
        Console.WriteLine("Meo..Meo...");
    }

    public void Eat()
    {
        Console.WriteLine("Nog-veg");
    }
}
class PetStore
```

The code defines a `Cat` class with three methods: `Walk()`, `Sound()`, and `Eat()`. The `Eat()` method is currently selected and being edited, as indicated by the cursor and the green vertical bar on the left.

so we need implementation of cat so we create instance of it

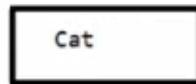
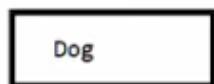
```
PetStore.cs  X  ConsoleApp
ConsoleApp  ConsoleApp.PetStore

9   class Dog...
26
27   class Cat...
44 class PetStore
45 {
46     static void Main()
47     {
48       Dog dog = new Dog();
49       Cat cat = new Cat();
50       Console.WriteLine("Dog info");
51       dog.Eat();
52       dog.Sound();
53       dog.Walk();
54
55       Console.WriteLine("Cat info");
56       cat.Eat();
57       cat.Sound();
58       cat.Walk();
59     }
60   }
61 }

121 %  No issues found  |  ⌂  ▾
Output  Package Manager Console  Error List ...  Immediate Window
```

```
C:\WINDOWS\system32\cmd.exe
Dog info
Veg and Non-veg
Bow Bow...
Use 4 legs to walk
Cat info
Non-veg
Meo..Meo...
Use 4 legs
Press any key to continue . . .
```

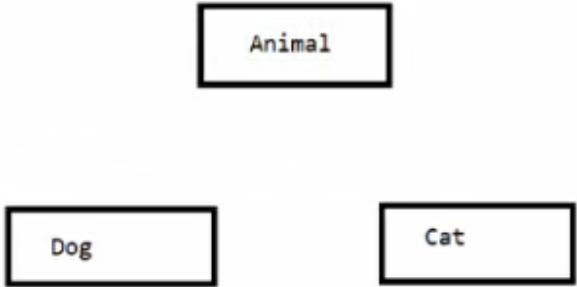
so we have 2 different classes cat and dog



now cat has its own implementation do has its own implementation

so when i look into this particular program so we will have something which is called retrospective done

In Retrospective what we do we just check what is the code we have written this is something which should be carry forward or not when i open this particular project and see, walk are present in both cat and dog and both are using 4 legs to walk but the implementation of sound and eat is different in dog and cat , So i'll discuss with the project lead and we came out with the new class called Animal



Why are we bringing a new class animal here ?

because i have some features which i can put it in my base class which i can always inherit from here

So from dog and cat i remove walk method which is repeated and i'll put it in the class called animal

PetStore.cs* X ConsoleApp

ConsoleApp

ConsoleApp.Animal

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      class Animal
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15     }
16     class Dog
17     {
18         public void Sound()
19         {
20             Console.WriteLine("Bow Bow...");
21         }
22     }
23 }
```

121 % 1 0 ↶ ↷ ⌂

```
PetStore.cs*  ✘  ConsoleApp
ConsoleApp  ✘  ConsoleApp.PetStore
40      }
41  class PetStore
42  {
43      static void Main()
44      {
45          Dog dog = new Dog();
46          Cat cat = new Cat();
47          Console.WriteLine("Dog info");
48          dog.Eat();
49          dog.Sound();
50          dog.Walk(); // Red squiggle underline
51
52          Console.WriteLine("Cat info");
53          cat.Eat();
54          cat.Sound();
55          Icat.Walk(); // Blue outline suggestion
56      }
57  }
58
59
```

Why we create Animal class?

Because we saw some common implementation so we keep that in Animal Class

The screenshot shows a code editor in Visual Studio with the file 'PetStore.cs' open. The code defines a class hierarchy:

```
class Animal
{
    public void Walk()
    {
        Console.WriteLine("Use 4 legs to walk");
    }
}

class Dog : Animal
{
    public void Sound()
    {
        Console.WriteLine("Bow Bow...");
    }

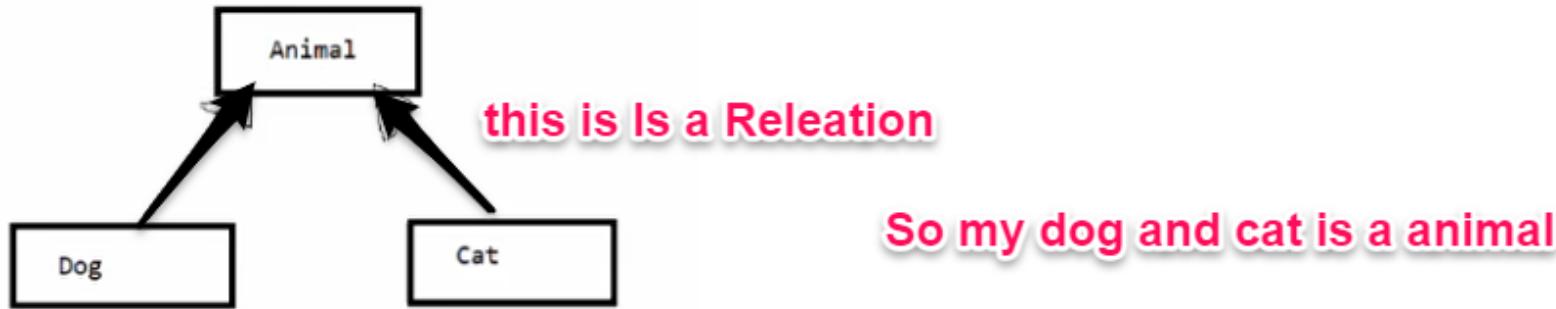
    public void Eat()
    {
        Console.WriteLine("Veg and Non-veg");
    }
}
```

A red box highlights the line 'class Dog : Animal'. A red arrow points from this line to a pink annotation.

by doing this we are inheriting we are saying that dog belongs to the animal Class

In java instead of : colon
we use extends keyword

this is a **IS A** Relation, so by doing this we are saying dog is a animal cat is a animal and it should be represented as arrow headed



So i don't have the implementation of walk in the dog or cat

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.Animal

```
37 }  
38     }  
39 }  
40 }  
41 class PetStore  
42 {  
43     static void Main()  
44     {  
45         Dog dog = new Dog();  
46         Cat cat = new Cat();  
47         Console.WriteLine("Dog info");  
48         dog.Eat();  
49         dog.Sound();  
50         dog.Walk(); ←  
51  
52         Console.WriteLine("Cat info");  
53         cat.Eat();  
54         cat.Sound();  
55         cat.Walk(); ←  
56     }  
57 }  
58 }
```

121 % No issues found

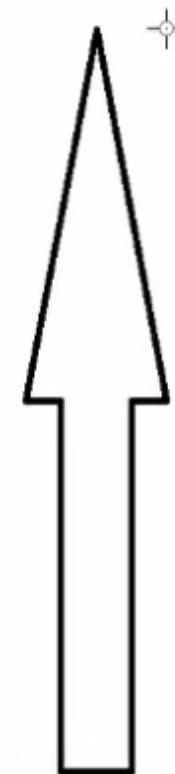
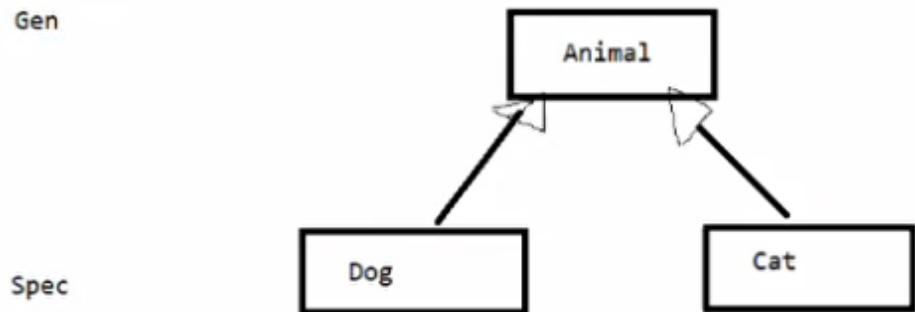
Output Package Manager Console Error List Immediate Window

But still we are able to access
that methods



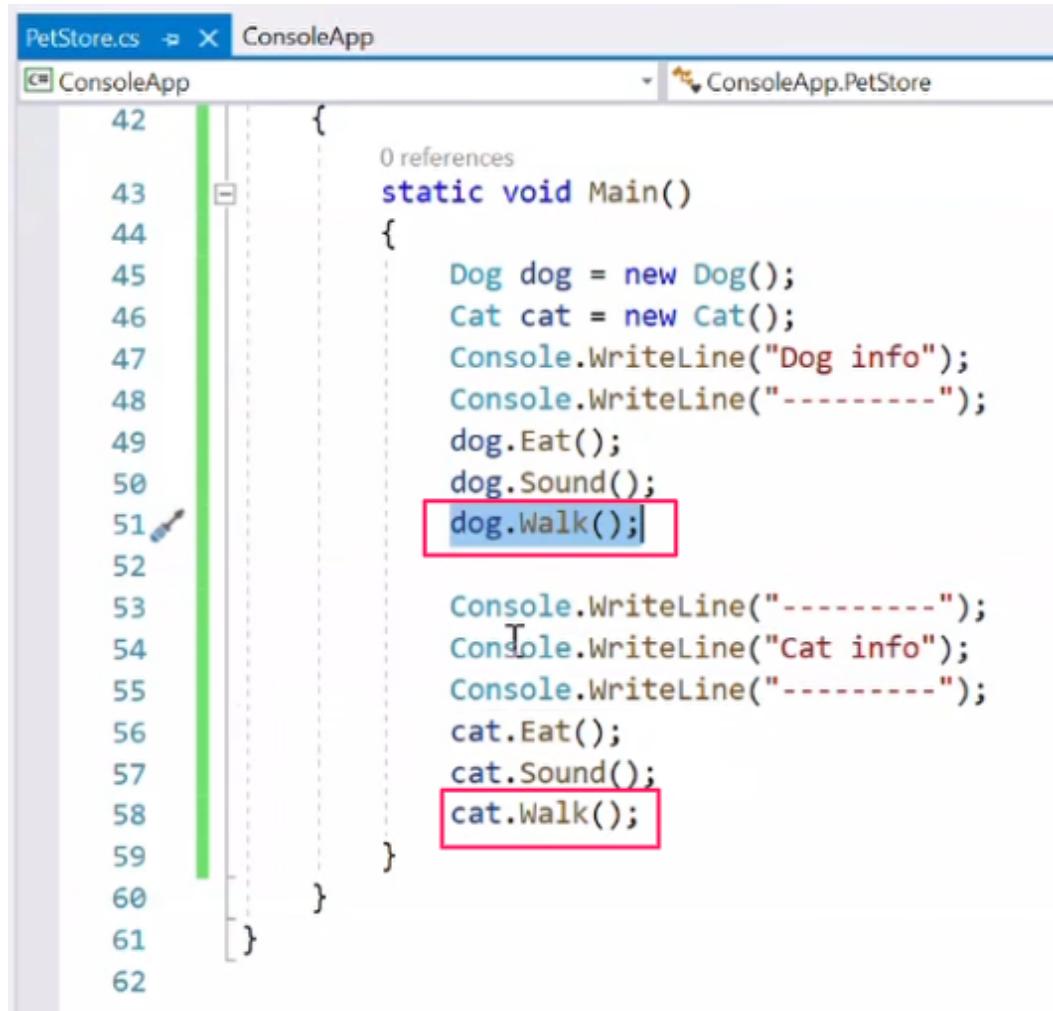
**So this is a type of Inheritance
Specialization to Generalization**

**So this will be between the classes
and up the ladder it is always generalization**



So this is Fabulous but still i'm thinking of Optimization

^{3.} program no 3



```
PetStore.cs  X  ConsoleApp
ConsoleApp
ConsoleApp.PetStore
42  {
43      static void Main()
44      {
45          Dog dog = new Dog();
46          Cat cat = new Cat();
47          Console.WriteLine("Dog info");
48          Console.WriteLine("-----");
49          dog.Eat();
50          dog.Sound();
51          dog.Walk(); // Red box highlights this line
52
53          Console.WriteLine("-----");
54          Console.WriteLine("Cat info");
55          Console.WriteLine("-----");
56          cat.Eat();
57          cat.Sound();
58          cat.Walk(); // Red box highlights this line
59      }
60  }
61 }
62 }
```

can we generalize as we have walk walk repeated, so that we have something that repeated so instead of doing that we have

```
PetStore.cs*  X  ConsoleApp
ConsoleApp  ConsoleApp.PetStore

40      }
41  class PetStore
42  {
43      void Display(Animal animal)
44      {
45          animal.Walk();
46      }
47
48  static void Main()
49  {
50      Dog dog = new Dog();
51      Cat cat = new Cat();
52      Console.WriteLine("Dog info");
53      Console.WriteLine("-----");
54      dog.Eat();
55      dog.Sound();
56      dog.Walk();

58      Console.WriteLine("-----");
59      Console.WriteLine("Cat info");
60      Console.WriteLine("-----");

121 %  No issues found
```

So can we access
this display method
here in main

No, because
we have to create
instance of this
petstore in main to access
the Display()

Or we can change this
to Static

PetStore.cs* ✘ X ConsoleApp

ConsoleApp

ConsoleApp.PetStore

```
40 }
41 class PetStore
42 {
43     static void Display(Animal animal)
44     {
45         animal.Walk();
46     }
47
48 static void Main()
49 {
50     Dog dog = new Dog();
51     Cat cat = new Cat();
52     Console.WriteLine("Dog info");
53     Console.WriteLine("-----");
54     dog.Eat();
55     dog.Sound();
56     dog.Walk();
57
58     Console.WriteLine("-----");
59     Console.WriteLine("Cat info");
60     Console.WriteLine("-----");
```

121 %

No issues found

Output Package Manager Console Error List ... Immediate Window

this is just a reference

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.PetStore

```
44 } animal.Walk();  
45 }  
46  
47 0 references  
48 static void Main()  
49 {  
50     Dog dog = new Dog();  
51     Cat cat = new Cat();  
52     Console.WriteLine("Dog info");  
53     Console.WriteLine("-----");  
54     dog.Eat();  
55     dog.Sound();  
56     //dog.Walk();  
57     Display(dog);  
58  
59     Console.WriteLine("-----");  
60     Console.WriteLine("Cat info");  
61     Console.WriteLine("-----");  
62     cat.Eat();  
63     cat.Sound();  
64     //cat.Walk();  
65     Display(cat);  
66 }
```

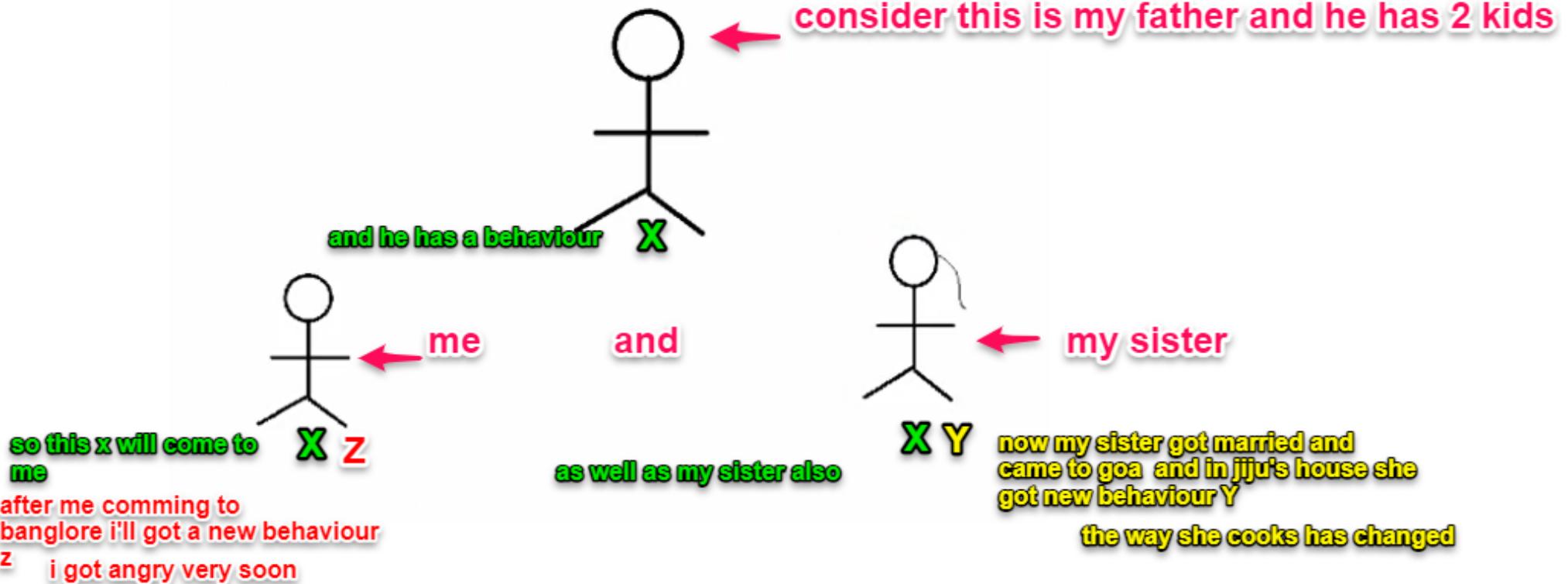
121 %

✖ 1 ⚠ 0 ← → ⌂ ▾

The screenshot shows the Microsoft Visual Studio IDE with the code editor open to a file named `PetStore.cs`. The code defines a `PetStore` class with a `Display` method that calls `Walk` and `Eat` methods on an `Animal` object. A red annotation with an arrow points to the `Eat()` call in the `Display` method, asking why it can't be accessed. Another annotation explains that `Eat` is from the child class or parent class, specifically the child class, and that it is a behaviour.

```
40 }
41 }
42 class PetStore
43 {
44     static void Display(Animal animal)
45     {
46         animal.Walk();
47         animal.Eat(); ← Why can't we access eat here
48     }
49     static void Main()
50     {
51         Dog dog = new Dog();
52         Cat cat = new Cat();
53         Console.WriteLine("Dog info");
54         Console.WriteLine("-----");
55         dog.Eat();
56         dog.Sound();
57         //dog.Walk();
58         Display(dog);
59
60         Console.WriteLine("-----");
61     }
62 }
```

Why can't we access eat here
Eat is from the child class or parent class?
--> Child Class
it is a behaviour



Now when i say **Son son = new Son();**

so with son i can access X because son already has a property x so he can access

From son i can access z **s.Z;**

so when i say **Father f = new Son();**

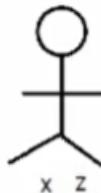
What i'm doing here?
i'm taking the reference of father

so can i say **f.X;**
obviously i can because father has a X property

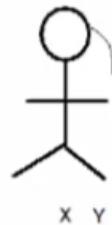
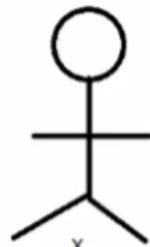
can a father access Z property
f.Z;

no because i cannot say dad is also become angry like me

my dad can never have my property



I



So to achieve this i'll Down Caste it

Down Casting

So to Down Caste it i don't know weather my animal is now handelling with the dog instance or with the cat instance

PetStore.cs* ✘ ConsoleApp

ConsoleApp

```
40 }  
41 class PetStore  
42 {  
43     static void Display(Animal animal)  
44     {  
45         animal.Walk();  
46         if(animal is Dog)  
47         {  
48             }  
49         animal.Eat();  
50     }  
51 }  
52  
53 static void Main()  
54 {  
55     Dog dog = new Dog();  
56     Cat cat = new Cat();  
57     Console.WriteLine("Dog info");  
58     Console.WriteLine("-----");  
59     dog.Eat();  
60     dog.Sound();
```

Output Package Manager Console Error List ... Immediate Window

that i have to check here

is animal a dog

PetStore.cs* X ConsoleApp

ConsoleApp

ConsoleApp.PetStore

```
40    }
41    class PetStore
42    {
43        static void Display(Animal animal)
44        {
45            animal.Walk();
46            if(animal is Dog)
47            {
48                Dog d = (Dog)animal;           if it is holding a reference of dog
49            }
50            animal.Eat();                 then i'll say
51        }
52
53        static void Main()
54        {
55            Dog dog = new Dog();
56            Cat cat = new Cat();
57            Console.WriteLine("Dog info");
58            Console.WriteLine("-----");
59            dog.Eat();
60            dog.Sound();                  i'm not creating a new object here
61        }
62
63    }
64
65    0 references
66    static void Main()
67    {
68        Dog dog = new Dog();
69        Cat cat = new Cat();
70        Console.WriteLine("Dog info");
71        Console.WriteLine("-----");
72        dog.Eat();
73        dog.Sound();
74    }
75
76    0 references
77    static void Main()
78    {
79        Dog dog = new Dog();
80        Cat cat = new Cat();
81        Console.WriteLine("Cat info");
82        Console.WriteLine("-----");
83        cat.Eat();
84        cat.Sound();
85    }
86
87    0 references
88    static void Main()
89    {
90        Dog dog = new Dog();
91        Cat cat = new Cat();
92        Animal animal = new Animal();
93        Console.WriteLine("Animal info");
94        Console.WriteLine("-----");
95        animal.Eat();
96        animal.Sound();
97    }
98
99    0 references
100   static void Main()
101   {
102       Dog dog = new Dog();
103       Cat cat = new Cat();
104       Animal animal = new Animal();
105       Console.WriteLine("Animal info");
106       Console.WriteLine("-----");
107       animal.Eat();
108       animal.Sound();
109   }
110
111 % 1 0 ← → ⌂ ⌃ ⌄
112 Output Package Manager Console Error List ... Immediate Window
113 Ready
```

if it is holding a reference of dog

then i'll say

i'm not creating a new object here

i'm down casting my animal here

A screenshot of the Microsoft Visual Studio IDE showing the code editor for a C# file named `PetStore.cs`. The code defines a `PetStore` class with a `Display` method. Inside the `Display` method, there is a conditional block that checks if the `animal` is a `Dog`. If true, it creates a `Dog` object named `d` and calls its `Eat` method. A code completion dropdown is open at the `d.` position, listing several methods: `Equals`, `GetHashCode`, `GetType`, `Sound`, `ToString`, and `Walk`. The `Eat` method is highlighted in blue, indicating it is the next available completion. The status bar at the bottom shows the zoom level as 121% and three errors (indicated by red X icons).

```
40 }  
41 }  
42 class PetStore  
43 {  
44     static void Display(Animal animal)  
45     {  
46         animal.Walk();  
47         if(animal is Dog)  
48         {  
49             Dog d = (Dog)animal;  
50             d.  
51         }  
52     }  
53     static void  
54     {  
55         Dog do  
56         Cat ca  
57         Console.WriteLine("Dog info");  
58         Console.WriteLine("-----");  
59         dog.Eat();  
60     }  
61 }
```

Now i'll say d. look at this i'll get eat method

The screenshot shows the Visual Studio IDE with the file 'PetStore.cs' open. The code defines a class 'PetStore' with a static method 'Display'. Inside 'Display', there is a conditional block that checks if the animal is a 'Dog' or a 'Cat'. If it's a dog, it calls 'Walk()', 'Eat()', and 'Sound()'. If it's a cat, it calls 'Walk()', 'Eat()', and 'Sound()'. The entire conditional block is highlighted with a red rectangle.

```
40     }
41     class PetStore
42     {
43         static void Display(Animal animal)
44         {
45             animal.Walk();
46             if(animal is Dog)
47             {
48                 Dog d = (Dog)animal;
49                 d.Eat();
50                 d.Sound();
51             }
52             else if(animal is Cat)
53             {
54                 Cat c = (Cat)animal;
55                 c.Eat();
56                 c.Sound();
57             }
58         }
59     }
60     static void Main()
61 }
```

From the parent if you want to access the child information you should always Down Caste it

The screenshot shows the Visual Studio IDE with the file 'PetStore.cs' open. The code defines a class 'PetStore' with a static method 'Display'. The 'Display' method takes an 'Animal' parameter and performs different actions based on whether it's a 'Dog' or a 'Cat'. A red box highlights the conditional blocks for 'Dog' and 'Cat'. The status bar at the bottom indicates 'No issues found'.

```
40 }  
41 class PetStore  
42 {  
43     static void Display(Animal animal)  
44     {  
45         animal.Walk();  
46         if(animal is Dog)  
47         {  
48             Dog d = (Dog)animal;  
49             d.Eat();  
50             d.Sound();  
51         }  
52         else if(animal is Cat)  
53         {  
54             Cat c = (Cat)animal;  
55             c.Eat();  
56             c.Sound();  
57         }  
58     }  
59     static void Main()  
60 }
```

Now as i'm writing
this particular piece of code

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.PetStore

```
53
54     Cat c = (Cat)animal;
55     c.Eat();
56     c.Sound();
57 }
58
59 0 references
60 static void Main()
61 {
62     Dog dog = new Dog();
63     Cat cat = new Cat();
64     Console.WriteLine("Dog info");
65     Console.WriteLine("-----");
66     dog.Eat(); // dog.Walk();
67     dog.Sound();
68 //dog.Walk();
69     Display(dog);
70
71     Console.WriteLine("-----");
72     Console.WriteLine("Cat info");
73     Console.WriteLine("-----");
74     cat.Eat();
75     cat.Sound();
```

121 % No issues found

i can remove this 2 lines of code
these 2 lines of code is not required

The screenshot shows the Microsoft Visual Studio IDE with the 'PetStore.cs' file open. The code is as follows:

```
59
60     static void Main()
61     {
62         Dog dog = new Dog();
63         Cat cat = new Cat();
64         Console.WriteLine("Dog info");
65         Console.WriteLine("-----");
66         //dog.Eat();
67         //dog.Sound();
68         //dog.Walk();
69         Display(dog);
70
71         Console.WriteLine("-----");
72         Console.WriteLine("Cat info");
73         Console.WriteLine("-----");
74         //cat.Eat();
75         //cat.Sound();
76         //cat.Walk();
77         Display(cat);
78     }
79
80 }
81
```

Red boxes highlight the commented-out code blocks for both the dog and cat classes. A green arrow points from the text "Look at this" to the first red box. Another green arrow points from the text "How beautifully we have eliminated these lines" to the second red box. The text "i can directly call display Dog and display cat" is displayed in large green letters at the bottom right.

Look at this
How beautifully we have
eliminated these lines

i can directly call display Dog and display cat

```
C:\WINDOWS\system32\cmd.exe
ls
Dog info
-----
Use 4 legs to walk
Veg and Non-veg
Bow Bow...
-----
Cat info
-----
Use 4 legs to walk
Nog-veg
Meo..Meo...
Press any key to continue . . .
```

The screenshot shows a C# code editor with the file 'PetStore.cs' open. The code defines a class 'PetStore' with a static method 'Display'. Inside 'Display', there is a switch statement that falls through to multiple cases. The first case checks if 'animal' is a 'Dog', and the second case checks if it is a 'Cat'. Both cases fall through to a common block of code that calls 'c.Eat()' and 'c.Sound()'. A red box highlights the entire switch block, and a blue box highlights the common code block. The status bar at the bottom indicates 'No issues found'.

```
40     }
41 }
42 class PetStore
43 {
44     static void Display(Animal animal)
45     {
46         animal.Walk();
47         if(animal is Dog)
48         {
49             Dog d = (Dog)animal;
50             d.Eat();
51             d.Sound();
52         }
53         else if(animal is Cat)
54         {
55             Cat c = (Cat)animal;
56             c.Eat();
57             c.Sound();
58         }
59     }
60 }
```

Now
Me being developer
i was thinking aree yaar
this is bakwas code

I cannot keep doing if and else
because tomorrow if i'll going to add new animal
so i cannot keep doing this if else if else

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.Animal

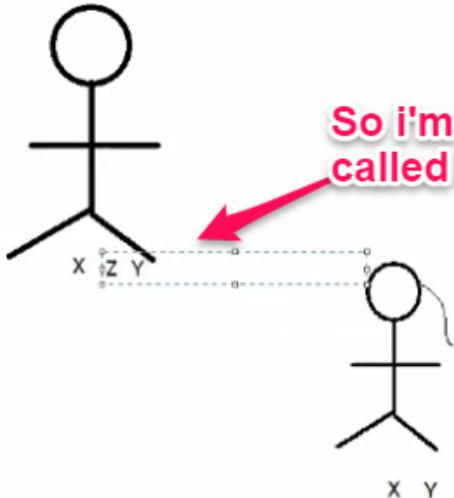
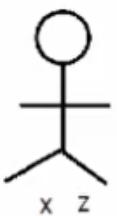
```
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9     class Animal
10    {
11        public void Walk()
12        {
13            Console.WriteLine("Use 4 legs to walk");
14        }
15    }
16    class Dog : Animal
17    {
18        public void Sound()
19        {
20            Console.WriteLine("Bow Bow...");
21        }
22    }
23 }
```

121 % No issues found

Output Package Manager Console Error List Immediate Window

So what i do is i'll write all the
Generalize-methods here
inside animal class

Build succeeded



So i'm introducing new behaviour
called Z and Y here which my dad doesn't have

so i'll say that these are dummy methods
they don't have any implementation

So by doing this we are saying father has property Z and Y and son and daughter will also have Z and Y so no need of me doing Down Casting it

i can directly say

f.X;

f.Y;

f.Z;

The screenshot shows a code editor in Visual Studio with the file 'PetStore.cs' open. The code defines a class 'Animal' with methods 'Walk()', 'Sound()', and 'Eat()'. It also defines a class 'Dog' that inherits from 'Animal' and overrides the 'Sound()' method. A green vertical bar on the left indicates code coverage. Red arrows point to the 'Sound()' methods in 'Animal' and 'Dog', while a green arrow points to the 'Sound()' method in 'Dog'. The status bar at the bottom shows '121 %' and '0'.

```
9     class Animal
10    {
11        1 reference
12        public void Walk()
13        {
14            Console.WriteLine("Use 4 legs to walk");
15        }
16        0 references
17        public void Sound()
18        {
19            Console.WriteLine("Used to talk");
20        }
21        0 references
22        public void Eat()
23        {
24            Console.WriteLine("Gives energy");
25        }
26        5 references
27        class Dog : Animal
28        {
29            1 reference
30            public void Sound()
```

i'll gives the general information
dosen't have any meaning

so as soon as i do this
ill get the green line here
i don't know about that
i'm not bothered also

Now you can see Animal has Eat walk all the methods

PetStore.cs* X ConsoleApp

ConsoleApp

ConsoleApp.PetStore

```
50 } 0 references
51 class PetStore
52 {
53     2 references
54         static void Display(Animal animal)
55         {
56             animal.Walk();
57             animal.
58             //if(an Eat
59             //{
60                 Equals
61                 // Dog GetHashCode
62                 // d.E GetType
63                 // d.S Sound
64                 //{
65                     // Cat , , animal;
66                     // c.Eat();
67                     // c.Sound();
68                 //}
69             }
70             0 references
71         static void Main()
```

121 % 0 4 ← → ⌂

Output Package Manager Console Error List ... Immediate Window

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.Animal

```
47
48     }
49 }
50 }
51 class PetStore
52 {
53     static void Display(Animal animal)
54     {
55         animal.Walk();
56         animal.Eat();
57         animal.Sound();

58         //if(animal is Dog)
59         //{
60             // Dog d = (Dog)animal;
61             // d.Eat();
62             // d.Sound();
63         //}
64         //else if(animal is Cat)
65         //{
66             // Cat c = (Cat)animal;
67             // c.Eat();
68             // c.Sound()\n
```

121 % 0 4 ← → ⌂ ⌄

Output Package Manager Console Error List Immediate Window

Item(s) Saved

```
C:\WINDOWS\system32\cmd.exe
Dog info
-----
Use 4 legs to walk
Gives energy
Used to talk
-----
Cat info
-----
Use 4 legs to walk
Gives energy
Used to talk
Press any key to continue . . .
```

Look at this it is not printing me the child details rather its printing me the parent details

SO this is wrong this is not what i want

but in Java if i write this particular code it will print me the information of child Because all methods in java are virtual by default and that's the biggest blunder of java that's why java is so slow

So this is wrong output

So what want is if i'm passing Dog here so i should get dog information, if i pass cat i should get cat information

meaning based on the reference what we are passing here in display, the respective information of that class gets printed

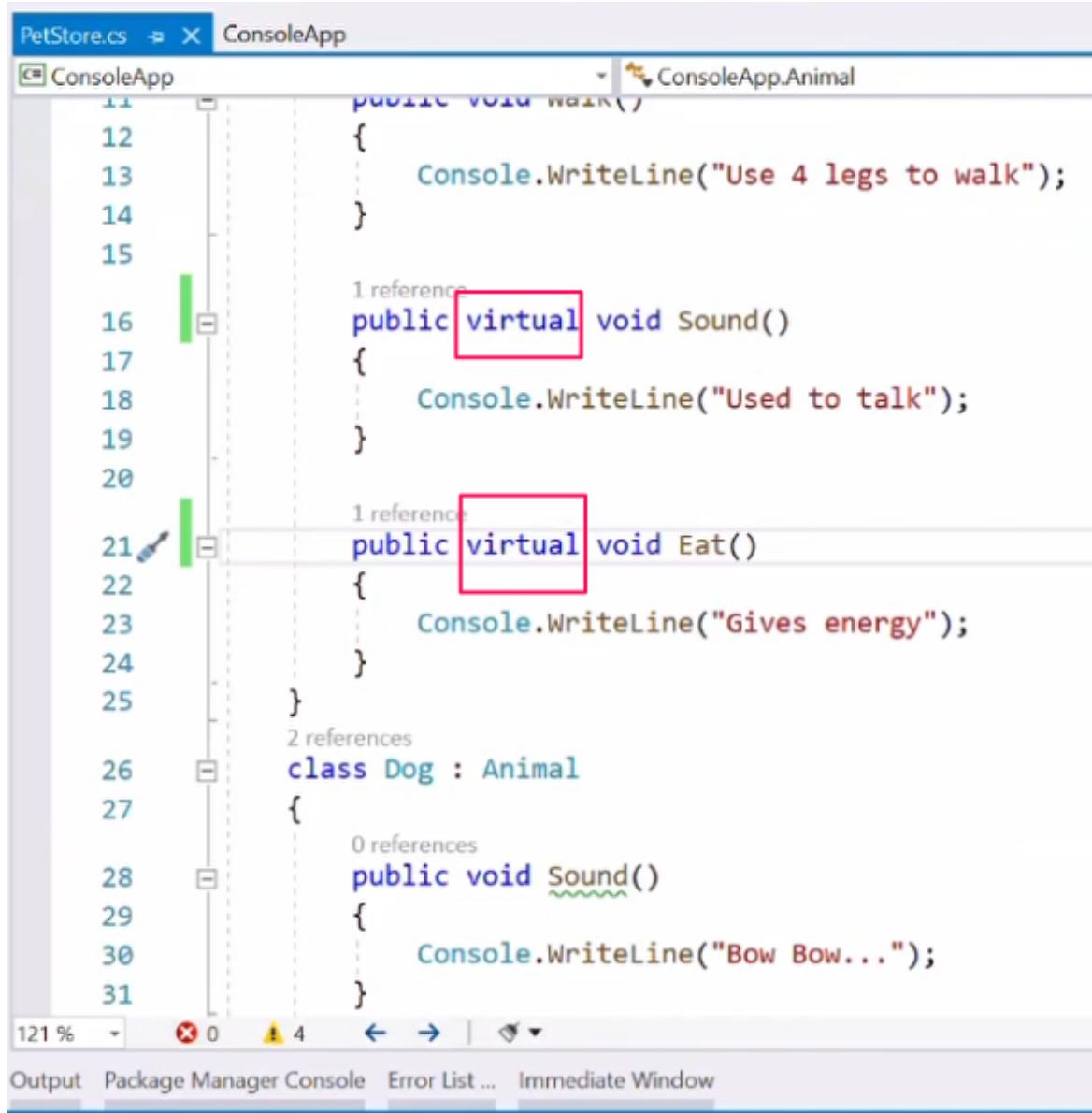
The screenshot shows a Microsoft Visual Studio interface. The title bar displays "PetStore.cs" and "ConsoleApp". The main window contains the following C# code:

```
70 } //}
71 }
72
73 static void Main()
74 {
75     Dog dog = new Dog();
76     Cat cat = new Cat();
77     Console.WriteLine("Dog info");
78     Console.WriteLine("-----");
79     //dog.Eat();
80     //dog.Sound();
81     //dog.Walk();
82     Display(dog); I
83
84     Console.WriteLine("-----");
85     Console.WriteLine("Cat info");
86     Console.WriteLine("-----");
87     //cat.Eat();
88     //cat.Sound();
89     //cat.Walk();
90     Display(cat);
91 }
92 }
```

The code defines a `Main` method that creates instances of `Dog` and `Cat`, prints their information, and then calls the `Display` method on each. The `Display` method is partially implemented, indicated by an insertion point (I) after the parameter name. The code uses standard C# syntax with `Dog` and `Cat` classes defined elsewhere.

So How will i achieve that

So i'll just add **Virtual** KeyWord here



```
PetStore.cs  X  ConsoleApp
ConsoleApp
public void Walk()
{
    Console.WriteLine("Use 4 legs to walk");
}

1 reference
public virtual void Sound()
{
    Console.WriteLine("Used to talk");
}

1 reference
public virtual void Eat()
{
    Console.WriteLine("Gives energy");
}

2 references
class Dog : Animal
{
    0 references
    public void Sound()
    {
        Console.WriteLine("Bow Bow...");
    }
}
```

The screenshot shows a code editor in Visual Studio with the file 'PetStore.cs' open. The code defines a class 'Animal' with two methods: 'Walk()' and 'Sound()'. Below it, a class 'Dog' inherits from 'Animal' and overrides the 'Sound()' method. The 'Sound()' and 'Eat()' methods in the 'Animal' class are highlighted with red boxes. The status bar at the bottom shows '121 %' zoom, 0 errors, 4 warnings, and navigation icons.

So as soon as i add virtual here it will say boss is this a overriding method

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. Below the menu is a toolbar with various icons. The main workspace displays the code for `PetStore.cs` under the `ConsoleApp` project. The code defines a base class `Animal` with an `Eat()` method and a derived class `Dog` that overrides the `Sound()` and `Eat()` methods.

```
17     {
18         Console.WriteLine("Used to talk");
19     }
20
21     public virtual void Eat()
22     {
23         Console.WriteLine("Gives energy");
24     }
25 }
26 class Dog : Animal
27 {
28     public void Sound()
29     {
30         Console.Wr
31     }
32 }
33 public void Eat()
34 {
35     Console.WriteLine("Veg and Non-veg");
36 }
```

A red box highlights a tooltip for the `Sound()` method in the `Dog` class. The tooltip contains the following information:

- `CS0114: 'Dog.Sound()' hides inherited member 'Animal.Sound()'.`
- To make the current member override that implementation, add the `override` keyword.
- Otherwise add the `new` keyword.
- [Show potential fixes \(Alt+Enter or Ctrl+.\)](#)

The status bar at the bottom shows the current line (Ln: 21), column (Ch: 18), and character count (Col: 24). The bottom navigation bar includes tabs for Output, Package Manager Console, Error List, and Immediate Window. A message bar at the bottom indicates "Item(s) Saved".

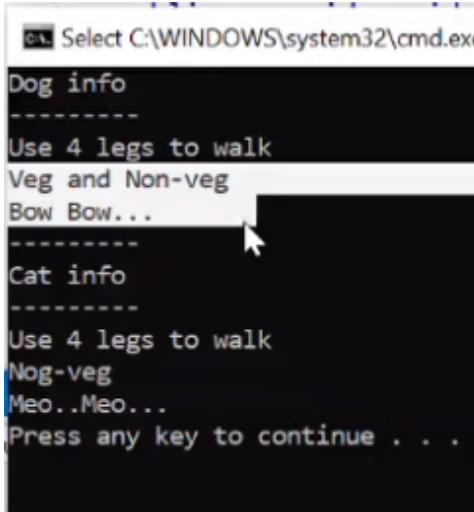
So i'll say yes this is a overriding method

The screenshot shows a code editor in Visual Studio with the file 'ConsoleApp.cs' open. The code defines a class 'Dog' that inherits from 'Animal'. It overrides two methods: 'Sound()' and 'Eat()'. The 'override' keyword is used in both cases. Red arrows point from the text 'I have to explicitly mention this in .Net but whereas java does this automatically' to the 'override' keywords in the 'Sound()' and 'Eat()' definitions.

```
17     {
18         Console.WriteLine("Used to talk");
19     }
20
21     public virtual void Eat()
22     {
23         Console.WriteLine("Gives energy");
24     }
25 }
26 class Dog : Animal
27 {
28     public override void Sound()
29     {
30         Console.WriteLine("Bow Bow...");
31     }
32
33     public override void Eat()
34     {
35         Console.WriteLine("Veg and Non-veg");
36     }
}
```

I have to explicitly mention this in .Net
but whereas java does this automatically

so i'm just saying that it's a overriding method in my dog class as well as in the cat class



```
PS C:\WINDOWS\system32\cmd.exe
Dog info
-----
Use 4 legs to walk
Veg and Non-veg
Bow Bow...
-----
Cat info
-----
Use 4 legs to walk
Non-veg
Meo..Meo...
Press any key to continue . . .
```

So even though you pass the reference of the child it automatically picks the information of the child not the parent because we are using virtual here, so internally a virtual table is created where these particular methods are registered and whenever you say override it will go to replace dynamically

this is what we called as **Dynamic Binding**

this is a concept of **Over Riding**

we are overriding the method on the fly

Now based on the object what we are passing whether it is a dog or a cat class whatever you are passing based on that the respective information gets printed

So whenever you see such code you should relate like this look like animal class this looks like cat class this looks like dog class

Now there is a possibility that we can create object of Animal class

The screenshot shows the Visual Studio IDE with the file 'PetStore.cs' open. The code defines a class 'PetStore' with a static method 'Display'. Inside 'Display', there is a line of code: 'Animal animal1 = new Animal();'. This line is highlighted with a red box and has a yellow warning lightbulb icon to its left. The status bar at the bottom shows '121 %' and 'No issues found'.

```
47
48     }
49 }
50 }
51 class PetStore
52 {
53     static void Display(Animal animal)
54     {
55         Animal animal1 = new Animal(); ← Any way we can create object
56         animal.Walk();
57         animal.Eat();
58         animal.Sound();
59
60         //if(animal is Dog)
61         //{
62             // Dog d = (Dog)animal;
63             // d.Eat();
64             // d.Sound();
65         //}
66         //else if(animal is Cat)
67         //{
68             // Cat c = (Cat)animal;
69             // c.Eat();
70         //}
71     }
72 }
```

Any way we can create object
of animal class

But Do Animal Exist in Real World

-----No

So i want that nobody will creates object of animal class
i don't want anybody to say

Animal animal1 = new Animal();

because this is bad, animal doesn't exist in real world So, How will i restrict them ?

By Making the animal class as Abstract we are ensuring that no one can create the object of that animal class

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.Animal

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      [abstract] class Animal
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15
16         public virtual void Sound()
17         {
18             Console.WriteLine("Used to talk");
19         }
20
21         public virtual void Eat()
22     }
23 }
```

Output Package Manager Console Error List ... Immediate Window

Item(s) Saved

Type here to search



As soon as i do this you can see i'll get Error here

PetStore.cs X ConsoleApp

ConsoleApp

ConsoleApp.Animal

```
64 } //}
65 //else if(animal is Cat)
66 //{
67 //  Cat c = (Cat)animal;
68 //  c.Eat();
69 //  c.Sound();
70 //}
71 }
72
73 static void Main()
74 {
75     Animal animal1 = new Animal(); 
76     Dog dog = new Dog();
77     Cat cat = new Cat();
78     Console.WriteLine("Dog info");
79     Console.WriteLine("-----");
80     //dog.Eat();
81     //dog.Sound();
82     //dog.Walk();
83     Display(dog);
84
85     Console.WriteLine("-----");
86     Console.WriteLine("Cat info");
```

121 %

Output Package Manager Console Error List Immediate Window

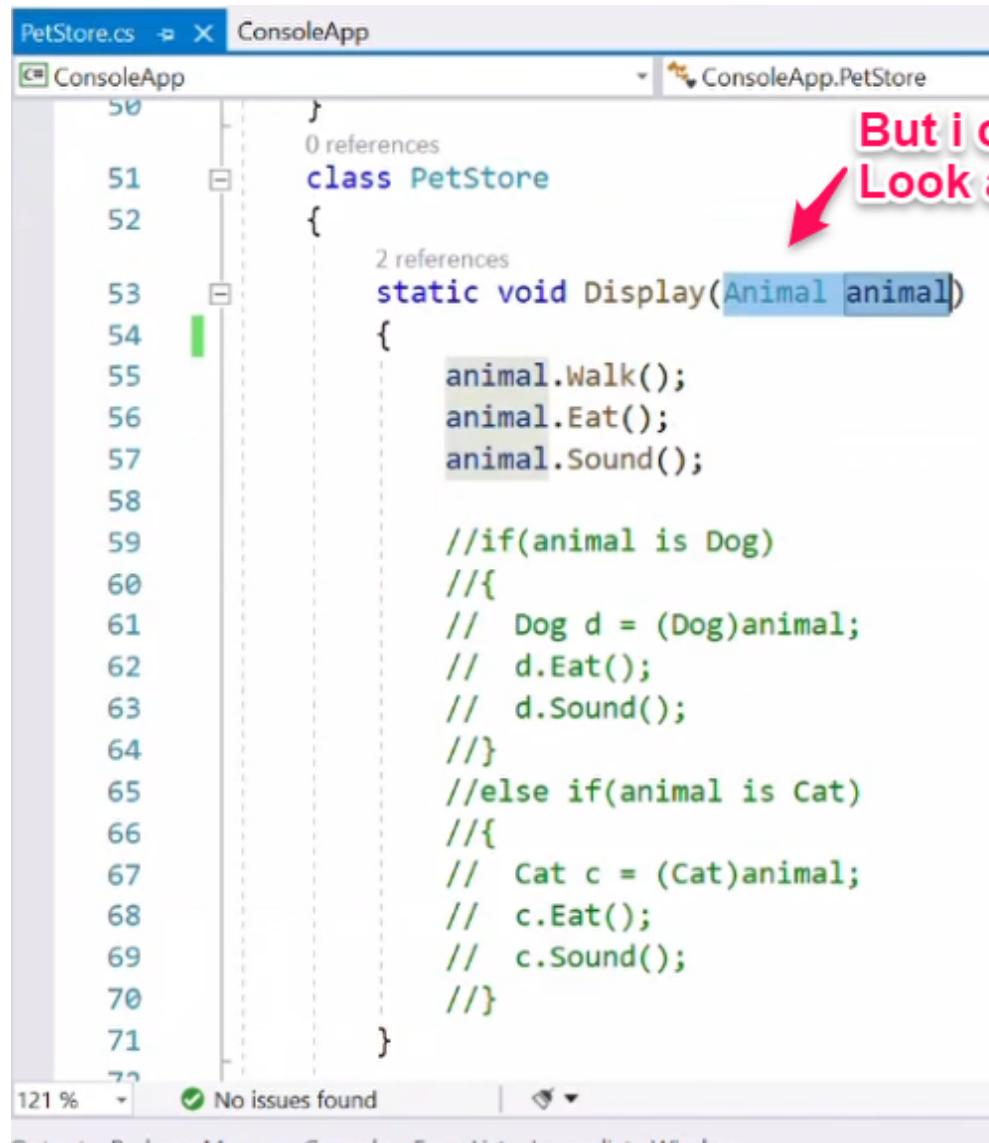
The screenshot shows a Microsoft Visual Studio interface with the following details:

- Title Bar:** Shows "PetStore.cs" and "ConsoleApp".
- Project Explorer:** Shows "ConsoleApp" and "ConsoleApp.PetStore".
- Code Editor:** Displays the following C# code:

```
64 } //}
65 //else if(animal is Cat)
66 //{
67 //  Cat c = (Cat)animal;
68 //  c.Eat();
69 //  c.Sound();
70 //}
71 }
72
0 references
73 static void Main()
74 {
75     ~Dog dog = new Dog();
76     Cat cat = new Cat();
77     Console.WriteLine("Dog info");
78     Console.WriteLine("-----");
79     //dog.Eat();
80     //dog.Sound();
81     //dog.Walk();
82     Display(dog);
83
84     Console.WriteLine("-----");
85     Console.WriteLine("Cat info");
86     Console.WriteLine("-----");
```
- Status Bar:** Shows "121 %", "x 1", "0", and navigation icons.
- Bottom Navigation:** Shows tabs for "Output", "Package Manager Console", "Error List ...", and "Immediate Window".

What are Abstract Classes ?

Abstract Classes are those classes of which we cannot create a Object or Instance



But i can create that as a Reference
Look at this

```
PetStore.cs  X  ConsoleApp
ConsoleApp      ConsoleApp.PetStore
50 }
51 class PetStore
52 {
53     static void Display(Animal animal)
54     {
55         animal.Walk();
56         animal.Eat();
57         animal.Sound();
58
59         //if(animal is Dog)
60         //{
61             // Dog d = (Dog)animal;
62             // d.Eat();
63             // d.Sound();
64         //}
65         //else if(animal is Cat)
66         //{
67             // Cat c = (Cat)animal;
68             // c.Eat();
69             // c.Sound();
70         //}
71 }
```

121 % No issues found

```
PetStore.cs*  X  ConsoleApp
ConsoleApp  ConsoleApp.PetStore

61      // Dog d = (Dog)animal;
62      // d.Eat();
63      // d.Sound();
64      //}
65      //else if(animal is Cat)
66      //{
67      // Cat c = (Cat)animal;
68      // c.Eat();
69      // c.Sound();
70      //}
71  }
72
73  0 references
74  static void Main()
75  {
76      Animal animal = new Dog();    I
77      animal = new Cat();          ← I can say this
78      animal = new Animal();       ← or this
79      animal = new Animal();       ← But i cannot say this
80
81      Dog dog = new Dog();
82      Cat cat = new Cat();
83      Console.WriteLine("Dog info");
84      Console.WriteLine("-----");
85      //dog.Eat();

121 %  ✘ 1  0  ← →  ⌂
```

So whenever you have abstract classes you can create a reference of it, you cannot create a object

Behaviours of Abstract Class

1. **Abstract Classes May or May not have Abstract methods**

The screenshot shows a code editor window for a C# file named PetStore.cs. The code defines an abstract class Animal with three methods: Walk(), Sound(), and Eat(). A concrete class Dog is derived from Animal. The code is annotated with callouts and boxes:

- A green vertical bar highlights the namespace declaration.
- A blue box highlights the entire body of the Animal class.
- Callouts indicate:
 - "3 references" for the Animal class definition.
 - "1 reference" for the Walk() method.
 - "3 references" for the Sound() method.
 - "3 references" for the Eat() method.
- A blue box highlights the Dog class definition.
- A callout indicates "2 references" for the Dog class definition.
- A status bar at the bottom shows "No issues found".

```
7  namespace ConsoleApp
8  {
9      3 references
10     abstract class Animal
11     {
12         1 reference
13         public void Walk()
14         {
15             Console.WriteLine("Use 4 legs to walk");
16         }
17         3 references
18         public virtual void Sound()
19         {
20             Console.WriteLine("Used to talk");
21         }
22         3 references
23         public virtual void Eat()
24         {
25             Console.WriteLine("Gives energy");
26         }
27     }
28     2 references
29     class Dog : Animal
30 }
```

Summary: Why I use Animal as a reference If I want to create object of dog we can create reference of dog, If I want to create object of cat we can create reference of cat, Why i use reference of animal ?

So

0 references
static void Main()

```
{  
    Dog dog = new Dog();  
    Cat cat = new Cat();  
    Console.WriteLine("Dog info");  
    Console.WriteLine("-----");
```

```
//dog.Eat();  
//dog.Sound();  
//dog.Walk();  
Display(dog);
```

```
Console.WriteLine("-----");  
Console.WriteLine("Cat info");  
Console.WriteLine("-----");
```

```
//cat.Eat();  
//cat.Sound();  
//cat.Walk();  
Display(cat);
```

```
}
```

is found | ⌂ ▾

Console Error List Immediate Window

So instead of me saying dog eat(),
dog sound(), dog walk(), cat eat(), cat walk()
here dog and cat has same implementation
so we are trying to generalize it here

Now take a example that you are creating a huge and bulky application with different type of databases
you have a piece of code and this piece of common code irrespective of database

Clipboard

Image

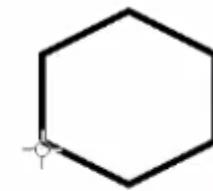
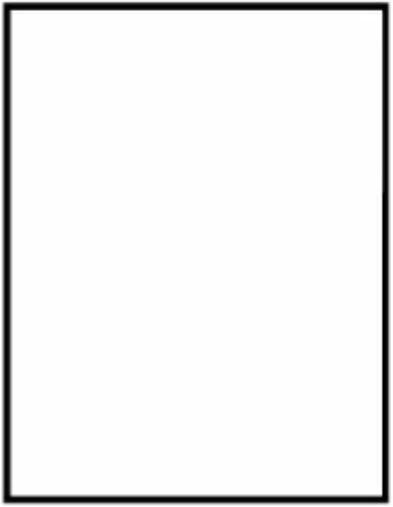
Tools

Shapes

Colors



and 1 u have a access database, and it was a pretty old database,



now we are trying to migrate the data from access database to new database i.e my sql database

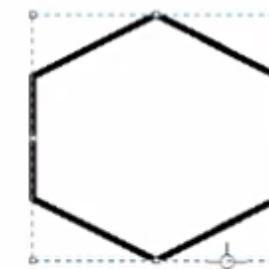
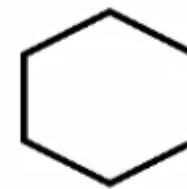
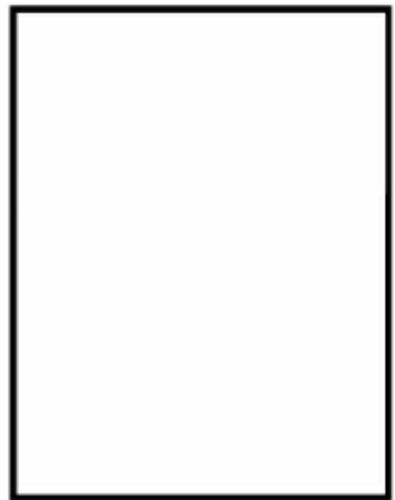
Clipboard

Image

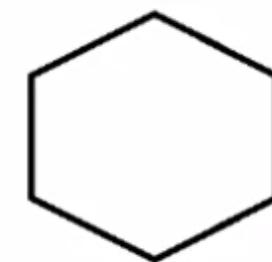
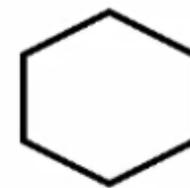
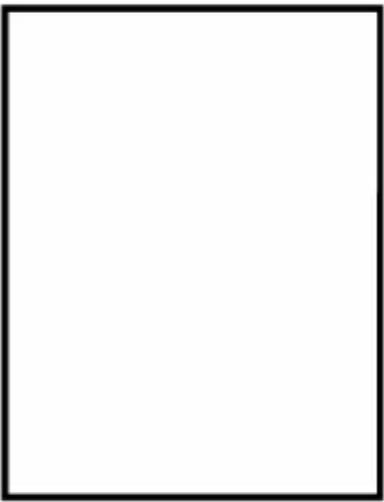
Tools

Shapes

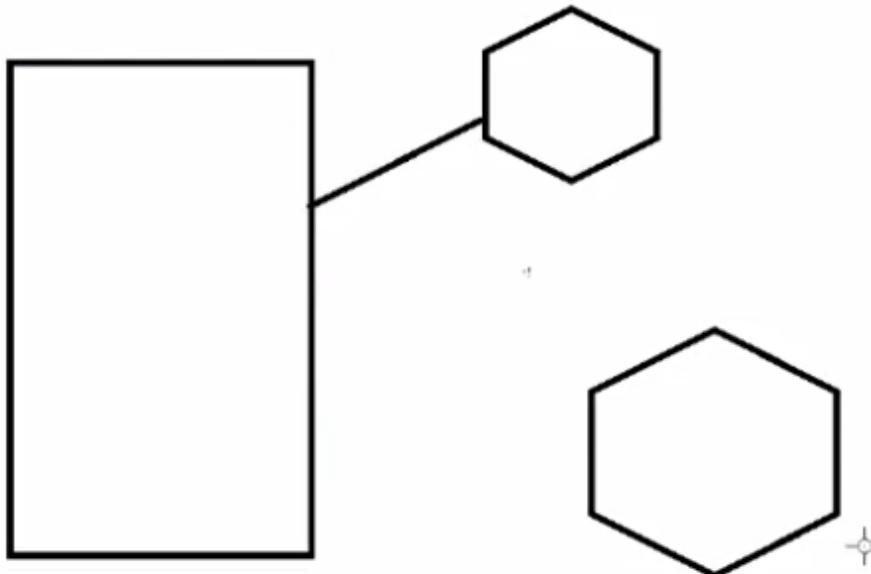
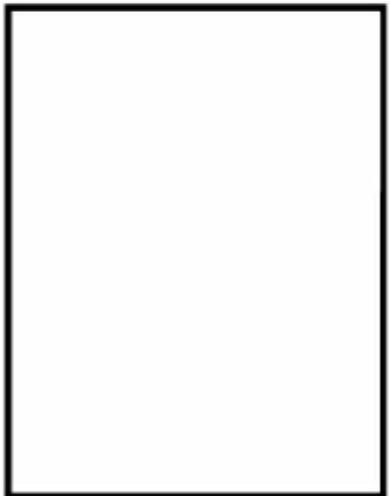
Colors



now while doing so you cannot directly do this as it has huge amount of data, we cannot directly dump all this data at day 1, so what we are doing is, we are writing a particular program



ad this particular database fetching the data from this database as of today

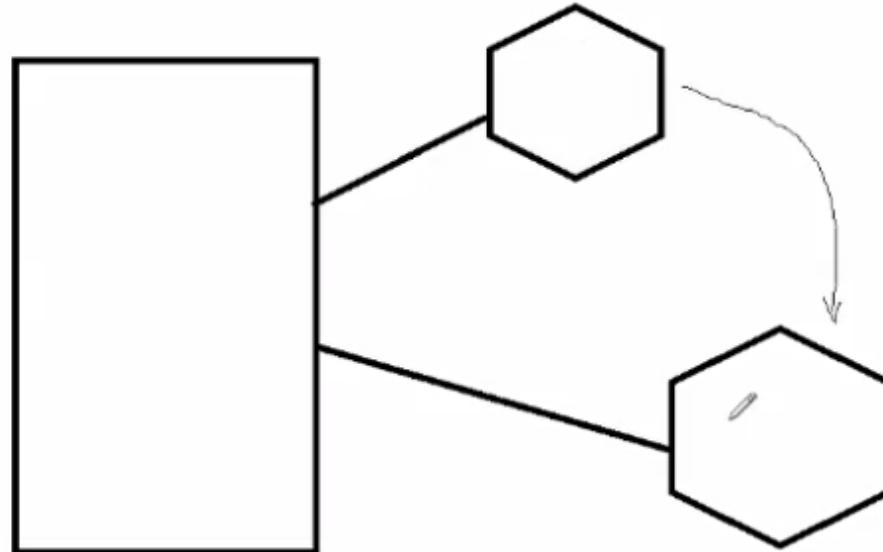
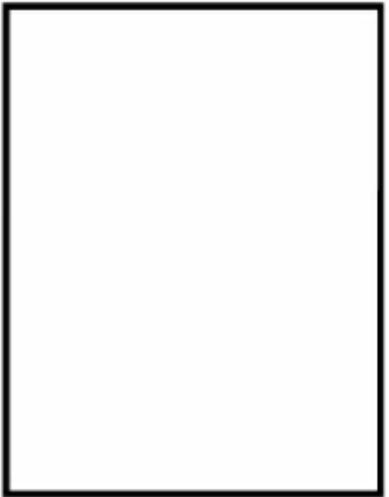


but client will say you know what deepak it is taking lot of time to fetch it is taking 2 min 6 min because access database will not support so many data

so looking that we thought that we'll migrate that to new database

so i want to make sure that what ever the screen we are getting here either from the access database or from the new database

When can it come to the new database? only for those data which is already been transferred here



only for those data it should get the data from new database not from the old one, and it is happening at the fly , meaning my application is running and I'm doing all this things

Dynamically I want to take a call weather weather I should go to this one or to this one

How do you do this If I'm creating Instance of both separately So How do you switch, Dynamically you cannot do this right

So what we are doing is based on what we are passing, dog or cat, The Implementation gets changed So dynamically based on some condition I can get the data from access database or from mysql database

This kind of configuration based working, I'll write configuration for both , I will not touch the code, inside the code I will not say go connect to this database or to this database

How to switch based on config

Like suppose I have a ADO Implementation and tomorrow I want to go with EF Implementation, So Implementations can be different but the methods what we are using will be common, So I will write a particular configuration to switch between both the implementation, that's where

your dynamic Overriding will come into pitcher, Dynamic Polymorphism will come into pitcher, Late Binding will come into pitcher

Now you can see that we are creating virtual methods here and now one are using it

```
ConsoleApp.PetStore
```

```
namespace ConsoleApp
{
    3 references
    abstract class Animal
    {
        1 reference
        public void Walk()
        {
            Console.WriteLine("Use 4 legs to walk");
        }

        3 references
        public virtual void Sound()
        {
            Console.WriteLine("Used to talk");
        }

        3 references
        public virtual void Eat()
        {
            Console.WriteLine("Gives energy");
        }
    }
    2 references
    class Dog : Animal
}
```

No issues found

Manager Console Error List Immediate Window

A tooltip for the `Console.WriteLine` call in the `Eat()` method of the `Dog` class is displayed, showing the `System.Console` class documentation:

`System.Console`
Represents the standard input, output, and error streams for console applications. T

So i'll Eliminate this methods and I'll make this methods as Abstract

```
namespace ConsoleApp
{
    abstract class Animal
    {
        public void Walk()
        {
            Console.WriteLine("Use 4 legs to walk");
        }

        public void Sound();
        //public virtual void Sound()
        //{
        //    Console.WriteLine("Used to talk");
        //}

        //public virtual void Eat()
        //{
        //    Console.WriteLine("Gives energy");
        //}
    }
}
```

6 0 ← → ⌂

Output Window Error List Immediate Window

AS soon as I put semicolon it means i don't want to write any sorts of dummy code here because it will also take some memory, Implementation someone will do

```
ConsoleApp
ConsoleApp.Animal
Walk()

7  namespace ConsoleApp
8  {
9      abstract class Animal
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15
16         public void Sound();
17         //public virtual
18         //{
19         //    Console.WriteLine("Used to bark");
20         //}
21
22         //public virtual void Eat()
23         //{
24         //    Console.WriteLine("Gives energy");
25         //}
26
27     }
}
```

The screenshot shows the code editor with the following details:

- Project: ConsoleApp
- File: ConsoleApp.Animal
- Method: Walk()
- Method: Sound();

The Sound() method is highlighted with a red squiggle under the semicolon. A tooltip appears with the message: "CS0501: 'Animal.Sound()' must declare a body because it is not marked abstract, extern, or partial".

But you get error saying that it always expects that to marked as Abstract , External or partial
because in .net every method should have body and if it is not having the body then
you should always mark method as Abstract , External or partial
You cannot just put the semicolon at the end, it is not possible
So I'll make it as abstract

PetStore.cs* ✘ X ConsoleApp

ConsoleApp

ConsoleApp.Animal

```
7  namespace ConsoleApp
8  {
9      abstract class Animal
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15
16         public abstract void Sound();
17
18         //public virtual void Sound()
19         //{
20             // Console.WriteLine("Used to talk");
21         //}
22
23         //public virtual void Eat()
24         //{
25             // Console.WriteLine("Gives energy");
26         //}
27     }
}
```

121 % ✘ 3 ⚠ 0 ← → 🔍 ▾

```
ip ConsoleApp1
```

```
namespace ConsoleApp
{
    3 references
    abstract class Animal
    {
        1 reference
        public void Walk()
        {
            Console.WriteLine("Use 4 legs to walk");
        }

        3 references
        public abstract void Sound();

        public abstract void Eat();|
```

```
//public virtual void Sound()
//{
//    Console.WriteLine("Used to talk");
//}
```

meaning i don't have Impl for this, Impl was done by this peoples below the classes

```
55
56
57
58
59
59 3 references
60     public override void Eat()
61     {
62         Console.WriteLine("Veg and Non-veg");
63     }
64
65
66
66 2 references
67     class Cat : Animal
68     {
69         3 references
70             public override void Sound()
71             {
72                 Console.WriteLine("Meo..Meo...");
73             }
74
75
76         3 references
77             public override void Eat()
78             {
79                 Console.WriteLine("Nog-veg");
80             }
81
82     }
83
84 0 references
```

I have a parent, So parent is having some kind of abstract behaviour

for ex parent is taking some kind of loan and now he is not paying it, the loan what ever he has taken

3 references
abstract class Animal

```
{  
    1 reference  
    public void Walk()  
    {  
        Console.WriteLine("Use 4 legs to walk");  
    }  
}
```

3 references
public abstract void Sound();



3 references
public abstract void Eat();

So the loan will be paid by his son or daughter , His son or daughter will override that particular method

```
//public virtual void Eat()
//{
//    Console.WriteLine("Gives energy");
//}
}
2 references
class Dog : Animal
{
    0 references
    public override void Sound()
    {
        Console.WriteLine("Bow Bow...");
    }

    3 references
    public override void Eat()
    {
        Console.WriteLine("Veg and Non-veg");
    }
}
```

2. `ConsoleApp`

```
7  namespace ConsoleApp
8  {
9      [yellow lightbulb icon] abstract class Animal
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15     }
16     public abstract void Sound();
17
18     public abstract void Eat();
19
20     //public virtual void Sound()
21     //{
22     //    Console.WriteLine("Used to talk");
23     //}
24
25     //public virtual void Eat()
26     //{
27     //    Console.WriteLine("Gives energy");
```

I'll remove this abstract

```
7  namespace ConsoleApp
8  {
9      class Animal
10     {
11         public void Walk()
12         {
13             Console.WriteLine("Use 4 legs to walk");
14         }
15
16         public abstract void Sound();
17
18         public abstract void Eat();
19
20         //public virtual void Sound()
21         //{
22         //    Console.WriteLine("Used to talk");
23         //}
24
25         //public virtual void Eat()
26         //{
27         //    Console.WriteLine("Gives energy");
28     }
29 }
```

you can see we got err, it is saying that it is abstract but it is present in non abstract class called animal

2. **Even if you have 1 method as Abstract the class should and must be abstract**

When ever you have abstract members, Class should and must be defined as abstract

What about reverse, If the class is abstract can it have a abstract method or it can have even non abstract method as well

==> It may or may not have

3. I have a abstract method here

The screenshot shows a code editor window with the file 'PetStore.cs' open. The code defines an abstract class 'Animal' with several methods. The 'Sound()' method is highlighted with a blue selection bar, indicating it is currently selected or being edited. The code is as follows:

```
abstract class Animal
{
    public void Walk()
    {
        Console.WriteLine("Use 4 legs to walk");
    }

    public abstract void Sound(); // This line is selected

    public abstract void Eat();

    //public virtual void Sound()
    //{
    //    Console.WriteLine("Used to talk");
    //}

    //public virtual void Eat()
    //{
    //    Console.WriteLine("Gives energy");
    //}
}
```

The code editor interface includes a status bar at the bottom showing '121 %' zoom, a red error icon with '1', a yellow warning icon with '0', navigation arrows, and other standard IDE controls.

and i'll not implement it here

The screenshot shows a Visual Studio code editor window with the following code:

```
soleApp                                ConsoleApp.Dog
22                                     // Console.WriteLine("Used to talk");
23                                     //}
24
25                                     //public virtual void Eat()
26                                     //{
27                                     //    Console.WriteLine("Gives energy");
28                                     //}
29
30 class Dog : Animal
31 {
32     // ...
33
34     public override void Eat()
35     {
36         Console.WriteLine("Veg and Non-veg");
37     }
38
39
40 class Cat : Animal
41 {
42     // ...
43     public override void Sound()
```

The code defines three classes: Animal, Dog, and Cat. The Animal class has a virtual Eat() method. The Dog class overrides Eat() and prints "Veg and Non-veg". The Cat class overrides Eat() and is partially visible. The code editor shows syntax highlighting and some annotations: a yellow lightbulb icon on line 32, a red error icon on line 42, and a status bar at the bottom showing 1 error and 0 warnings.

It simply means your father has taken a loan and neither he is paying the loan nor you
So bank wale err marenge 😊

So this will give me err saying

A screenshot of a code editor in Visual Studio. A tooltip is displayed over the word 'Dog' in the line 'class Dog : Animal'. The tooltip contains:

- 2 references
- class ConsoleApp.Dog
- CS0534: 'Dog' does not implement inherited abstract member 'Animal.Sound()'
- Show potential fixes (Alt+Enter or Ctrl+.)

The code in the editor is:

```
public class Dog : Animal
{
}
```

So I'll say boss i'll not implement it, so remove this you have to make this class also as abstract

A screenshot of a code editor in Visual Studio. The code is:

```
//public virtual void Eat()
//{
//    Console.WriteLine("Gives energy");
//}
2 references
abstract class Dog : Animal
{
}

3 references
public override void Eat()
{
    Console.WriteLine("Veg and Non-veg");
}
2 references
```

meaning I'm saying that dog class is a incomplete class
now meaning animal and dog both are incomplete class

that means now I cannot create instance of dog class

```
66      //{
67      //  Cat c = (Cat)animal;
68      //  c.Eat();
69      //  c.Sound();
70      //}
71  }
72
73  0 references
74  static void Main()
75  {
76      Dog dog = new Dog();
77      Cat cat = new Cat();
78      Console.WriteLine("Dog info");
79      Console.WriteLine("-----");
80      //dog.Eat();
81      //dog.Sound();
82      //dog.Walk();
83      Display(dog);
84
85      Console.WriteLine("-----");
86      Console.WriteLine("Cat info");
87      Console.WriteLine("-----");
88      //cat.Eat();
89  }
```

Do we have a dog in the real world

So when i say dog so which pitcher is coming in your mind ?

=> There are so many dogs are there right, street dog, bull dog, labra

So dog is not there in the world, When you say dog so dog is generalized

So I'll say Golden Retriever (GR) is a dog

```
25     //public virtual void Eat()
26     //{
27     //    Console.WriteLine("Gives energy");
28     //}
29 }
30 abstract class Dog : Animal
31 {
32
33     public override void Eat()
34     {
35         Console.WriteLine("Veg and Non-veg");
36     }
37 }
38
39 class GR : Dog
40 {
41     class ConsoleApp.GR
42     {
43     }
44 }
```

CS0534: 'GR' does not implement inherited abstract member 'Animal.Sound()'

Show potential fixes (Alt+Enter or Ctrl+.)

Now look at this I got a err msg here, saying boss you should inherit

As soon as a new baby takes a birth, so new baby get a msg on his mobile that you have to pay this particular amount

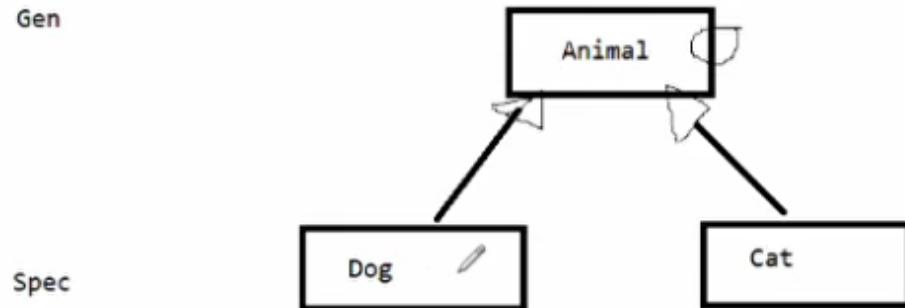
So what i do id I press Ctrl + . enter

So that particular method is present here, You just write a impl here

```
27     // Console.WriteLine("Gives energy");
28 }
29 }
30 abstract class Dog : Animal
31 {
32
33     public override void Eat()
34     {
35         Console.WriteLine("Veg and Non-veg");
36     }
37 }
38
39 class GR : Dog
40 {
41
42     public override void Sound()
43     {
44         throw new NotImplementedException(); I
```

So somewhere down the line you should handle this abstraction otherwise all the intermediate methods you have to keep as abstract

Summary of Inheritance: the commonly written code can always be pushed to a common class and that can be inherited
why we are doing this ---- Code Reuse ability
So this is Specialization to Generalization



While doing this if we inherit animal to dog and cat, so a copy of animal from the memory (it will take the complete animal class) and it is pasted here in cat and dog class

that's why when we say dog. so it will access the whole animal class info as well

But what if I wrote my own information(method) in Dog which is similar to parent i.e Animal, Then we saw the information of parent was getting printed

So how did we overcome this, I wan't my child information to get printed not the parent

So to overcome this we have to use keywords **Virtual** and **Overriding**

We mention Parent to **Virtual**

and Child to **Overriding**

as soon as we did this, the child based on the reference, the respective method gets called

Doing this we saw that we want to access the child info from the animal reference

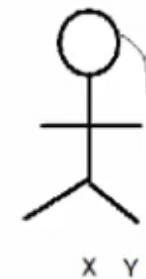
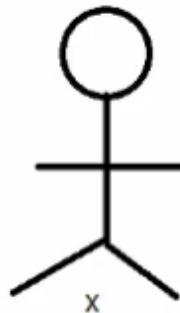
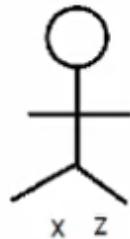
CAN I access child data from animal

No, because copy of child is not present in the parent

So How can i achieve this, With the F I want to access the Z

```
Son s = new Son();
s.x;
s.z;
```

```
Father f = new Son();
f.x;
f.z;
```



I.e means I want to do **Down casting**

The screenshot shows a Microsoft Visual Studio interface. The top bar displays the project name "ConsoleApp" and the file "PetStore.cs". The code editor window contains the following C# code:

```
63     animal.Walk();
64     animal.Eat();
65     animal.Sound();
66
67     //if(animal is Dog)
68     //{
69     //    Dog d = (Dog)animal;
70     //    d.Eat();
71     //    d.Sound();
72     //}
73     //else if(animal is Cat)
74     //{
75     //    Cat c = (Cat)animal;
76     //    c.Eat();
77     //    c.Sound();
78     //}
79 }
80
81 static void Main()
82 {
83     Dog dog = new GR();
84     Cat cat = new Cat();
85     Console.WriteLine("Dog info");
```

The code uses multiple levels of indentation and includes several comments. The class names "Dog" and "Cat" are highlighted in blue, while method names like "Walk()", "Eat()", and "Sound()" are in black. The code editor has a light gray background with vertical and horizontal dashed lines for readability. A status bar at the bottom indicates "121 %", "No issues found", and shows icons for Output, Package Manager Console, Error List, and Immediate Window.

so we are down casting it and after down casting, we are trying to access it .

So we understood that it is a costly process, to access it you have to write again and again if else if else soo many times with 2 class its ok but what if i keep adding multiple animal class, so this becomes very huge

So to overcome this we introduce 2 dummy method in the parent

The screenshot shows the code editor for a C# file named PetStore.cs. The code defines an abstract class Animal with two abstract methods: Walk() and Sound(). It also contains two dummy methods: Sound() and Eat(), which are declared as public virtual void but contain only double slashes (//). An abstract class Dog is then derived from Animal. The code editor highlights the Sound() and Eat() methods in blue, indicating they are being edited.

```
PetStore.cs  ConsoleApp
ConsoleApp
public void Walk()
{
    Console.WriteLine("Use 4 legs to walk");
}

3 references
public abstract void Sound();

3 references
public abstract void Eat();

//public virtual void Sound()
//{
//    Console.WriteLine("Used to talk");
//}

//public virtual void Eat()
//{
//    Console.WriteLine("Gives energy");
//}

abstract class Dog : Animal
{}
```

then we could directly access it

to access that we say virtual and override

but we understood that Implementation of this will also be costly, it will also take memory,
So that's a reason we created it as a ABSTRACT method

The screenshot shows a code editor window with the title "ConsoleApp.Animal". The code is as follows:

```
namespace ConsoleApp
{
    abstract class Animal
    {
        public void Walk()
        {
            Console.WriteLine("Use 4 legs to walk");
        }

        public abstract void Sound();
        public abstract void Eat();

        //public virtual void Sound()
        //{
        //    Console.WriteLine("Used to talk");
        //}

        //public virtual void Eat()
        //{
        //    Console.WriteLine("Gives energy");
        //}
    }
}
```

A mouse cursor is hovering over the `Sound()` method. The status bar at the bottom left says "No issues found".

so when ever you create a abstract method then you have to create class also as abstract

when ever you do this, you cannot create a object/Instance of abstract class

you cannot say

```
animal a = new animal
```

that is not possible

and abstract method you have should and must be implemented in its child class

```
PetStore.cs  X  ConsoleApp
ConsoleApp  ConsoleApp.Dog

25     //public virtual void Eat()
26     //{
27     //    Console.WriteLine("Gives energy");
28     //}
29 }
30 2 references
31 abstract class Dog : Animal
32 {
33     3 references
34     public override void Eat()
35     {
36         Console.WriteLine("Veg and Non-veg");
37     }
38
39 0 references
40 class GR : Dog
41 {
42     3 references
43     public override void Sound()
44     {
45
46
47     }
}

121 %  ✓ No issues found
```

otherwise you should make even the child class as abstract

In a project Naming is very important

student if you look into, we have already created a constructor in this

The screenshot shows a Microsoft Visual Studio code editor window for a file named `InhDemo1.cs`. The code is as follows:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      class InhDemo1
10     {
11         static void Main()
12         {
13             Student
14         }
15     }
16 }
17
```

A yellow vertical bar highlights the word "Student" at the end of line 13. A tooltip box appears over the cursor, containing the following text:

CS0119: 'Student' is a type, which is not valid in the given context
Show potential fixes (Alt+Enter or Ctrl+.)

look at this we already have a constructor

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays `ConsoleApp.cs` with the following content:

```
public class Student
{
    public int id { get; set; }

    public string name { get; set; }

    public Student(int id)
    {
        if(id > 5)
        {
            id = id + 5;
        }
        this.id = id;
        Console.WriteLine("0 para const");
    }

    public Student(int id, string name):this(id)
    {
        this.name = name;
        Console.WriteLine("1 para const");
    }
}

class PassbyValueDemo2
```

The Solution Explorer on the right shows the project structure for 'ConsoleApp' (2 of 2 items):

- AccessTest
- ConsoleApp
 - Properties
 - References
 - CollectionsDemo
 - AccessDemo.cs
 - App.config
 - ArrayDemo1.cs
 - ArrayDemo2.cs
 - BreakContinueDemo
 - Calc.cs
 - ClassDemo.cs
 - CommandLineDemo
 - ConvertType.cs
 - EnumDemo.cs
 - GoToDemo.cs
 - IfDemo1.cs
 - IfDemo2.cs
 - IfDemo3.cs
 - IncDemo.cs
 - InhDemo1.cs
 - InOutDemo.cs
 - InOutDemo1.cs

Is it ok if i inherit that class here

InhDemo1.cs* ✘ ConsoleApp*

ConsoleApp

ConsoleApp.DegreeStudent

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9
10    class DegreeStudent : Student
11    {
12    }
13  }
14
15  class InhDemo1
16  {
17    static void Main()
18    {
19      Student
20    }
21 }
```

121 % 3 0 ← → ⌂

Output Package Manager Console Error List Immediate Window

Ready

Type here to search

As soon as i say degree student is a Student here , I'm getting here, Why??

```
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9
10    class DegreeStudent : Student
11    {
12      class ConsoleApp.DegreeStudent
13      {
14        CS1729: 'Student' does not contain a constructor that takes 0 arguments
15        Show potential fixes (Alt+Enter or Ctrl+.)
16
17    class InhDemo1
18    {
19      static void Main()
20      {
21      }
22    }
23 }
```

Lets get back to this later

Suppose I have a class called Base

my Base class is having some kind of property

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The toolbar below has icons for file operations like Open, Save, and Print, along with buttons for Debug, Start, and other tools.

The main window displays a C# code editor for a file named InhDemo1.cs. The code is part of a project named ConsoleApp. The code shown is:

```
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp
{
    class Base
    {
    }

    //class DegreeStudent : Student
    //{
    //}

    class InhDemo1
    {
        static void Main()
    }
}
```

The code editor features color-coded syntax highlighting: blue for keywords, green for comments, and red for errors. A yellow vertical bar highlights the opening brace of the 'Base' class. The status bar at the bottom indicates "121 %", "No issues found", and shows tabs for Output, Package Manager Console, Error List ..., Immediate Window. The bottom left corner shows a "Ready" status.

InhDemo1.cs* X ConsoleApp*

ConsoleApp

ConsoleApp.InhDemo1

```
25
26
27     //class DegreeStudent : Student
28     //{
29
30     //}
31
32     0 references
33     class InhDemo1
34     {
35         0 references
36         static void Main()
37         {
38             Derive d = new Derive();
39             Base b = new Base();
40             Base b1 = new Derive();
41         }
42     }
```

in Main() i'll say

```
InhDemo1.cs*  X ConsoleApp*  
ConsoleApp  
ConsoleApp.InhDemo1  
25  
26  
27     //class DegreeStudent : Student  
28     //{
/>29  
30     //}
/>31  
32     0 references  
class InhDemo1  
33     {  
34         0 references  
static void Main()  
35         {  
36             Derive d = new Derive();|  
37             Base b = new Base();  
38             Base b1 = new Derive();  
39         }  
40     }  
41 }  
42
```

I'm creating object of child

I'm creating object of parent

I'm creating a reference of the father but it is having object of the child

InhDemo1.cs* X ConsoleApp*

ConsoleApp

ConsoleApp.InhDemo1

```
25
26
27     //class DegreeStudent : Student
28     //{
29
30     //}
31
32     class InhDemo1
33     {
34         static void Main()
35         {
36             Derive d = new Derive();
37             Base b = new Base();
38             Base b1 = new Derive();
39             Derive d1 = new Base(); ← here I'm trying to say
40         }
41     }
42
43
```

Son = new Father
which is not possible

What will be the output of this program

InhDemo1.cs* ✘ X ConsoleApp*

ConsoleApp

ConsoleApp.InhDemo1

```
20      2 references
21      public Derive()
22      {
23          Console.WriteLine("Const of derive called");
24      }
25      // class DegreeStudent : Student ...
26
27
28      0 references
29      class InhDemo1
30      {
31          0 references
32          static void Main()
33          {
34              Derive d = new Derive();
35              Base b = new Base();
36              Base b1 = new Derive();
37              //Derive d1 = new Base();
38          }
39      }
40  }
41  }
42 }
```

121 % ✅ No issues found

C:\WINDOWS\system32\cmd.exe

```
Const of base called
Const of derive called
Const of base called
Const of base called
Const of derive called
Press any key to continue . . .
```

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays two files: `InhDemo1.cs` and `ConsoleApp`. The `InhDemo1.cs` file contains C# code demonstrating inheritance. The `ConsoleApp` file shows the output of the application's execution.

```
20     2 references
21         public Derive()
22             {
23                 Console.WriteLine("Const of derive called");
24             }
25
26 // class DegreeStudent : Student ...
27
28     0 references
29         class InhDemo1
30         {
31             0 references
32                 static void Main()
33                 {
34                     Derive d = new Derive();
35                     Console.WriteLine("-----");
36                     Base b = new Base();
37                     Console.WriteLine("-----");
38                     Base b1 = new Derive();
39                     //Derive d1 = new Base();
40
41                 }
42             }
43 }
```

The output window on the right shows the console log from `C:\WINDOWS\system32\cmd.exe`:

```
Const of base called
Const of derive called
-----
Const of base called
-----
Const of base called
Const of derive called
Press any key to continue . . .
```

my derive class is inheriting the base class that's how first it has to create the object of base class
like son cannot be existing without the father,
so first father should be there
that's why first it create object of base then derived

Now move to 2nd Can a father be there without child ----- YES

now when i say Base b1 = new Derive();

So obviously it will go to the base create the object of base class then it create the object of derived class

So whenever you create a object of derive it will always make sure 1st it calls the base class constructor and then it creates the object of base class and the it creates the object of self

when ever you create a object of the derive it will always check weather the base class already has the information or it has a constructor or not

```
InhDemo1.cs*  X ConsoleApp
ConsoleApp
ConsoleApp.Base
10 4 references
11 class Base
12 {
13 }
14
15 4 references
16 class Derive : Base
17 {
18     2 references
19     public Derive()
20     {
21         Console.WriteLine("Const of derive called");
22     }
23 }
```

I don't have anything any constructor

```
inhDemo1.cs*  ✘ X  ConsoleApp
ConsoleApp  ConsoleApp.Base

10  5 references
11  class Base
12  {
13  1 reference
14  12: public Base(int x)
15  {
16  Console.WriteLine("Const of base called");
17  }
18  4 references
19  class Derive : Base
20  {
21  2 references
22  20: public Derive()
23  {
24  Console.WriteLine("Const of derive called");
25  }
26  }
```

Now I'll pass int X here, see I got Error

now this derive is trying to create a object of base, but as my base doesn't have a default constructor or 0 parameterize constructor in this case

So it will say Boss you know what I cannot do this because you don't have a constructor which will not give me any parameter

So I need to support this constructor

I need to create a instance of this particular constructor

```
1 reference
public Base(int x)
{
    Console.WriteLine("Const of base called");
}
```

this base is expecting me to pass x

So i need to ask user to pass x here or in this particular constructor I have to call base passing some value (dummy value)