UNDERSTANDING INHERITANCE AND DIFFERENT TYPES OF INHERITANCE

Inheritance is a mechanism of acquiring the features and behaviors of a class by another class. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class. Inheritance implements the IS-A relationship.

For example, mammal IS-A animal, dog IS-A mammal; Hence dog IS-A animal as well.

Advantages

1. Reduce code redundancy.
2. Provides code reusability.
3. Reduces source code size and improves code readability.
4. Code is easy to manage and divided into parent and child classes.
5. Supports code extensibility by overriding the base class functionality within child classes.

Disadvantages

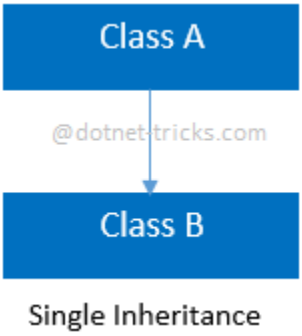
1. In Inheritance base class and child classes are tightly coupled. Hence If you change the code of parent class, it will get affects to the all the child classes.
2. In class hierarchy many data members remain unused and the memory allocated to them is not utilized. Hence affect performance of your program if you have not implemented inheritance correctly.

Different Types of Inheritance

OOPs supports the six types of inheritance as given below-

1. Single inheritance

In this inheritance, a derived class is created from a single base class.

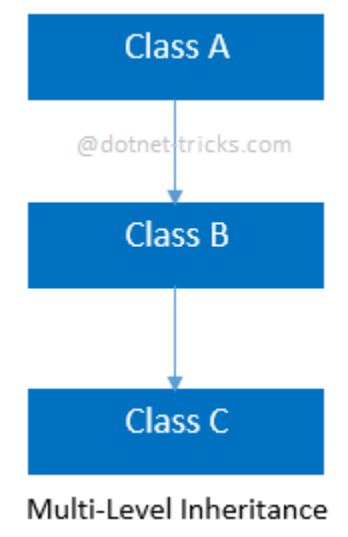


Single Inheritance

```
1. //Base Class
2. class A
3. {
4.  public void fooA()
5.  {
6.  //TO DO:
7.  }
8. }
9.
10. //Derived Class
11. class B : A
12. {
13.  public void fooB()
14.  {
15.  //TO DO:
16.  }
17. }
```

2. Multi-level inheritance

In this inheritance, a derived class is created from another derived class.



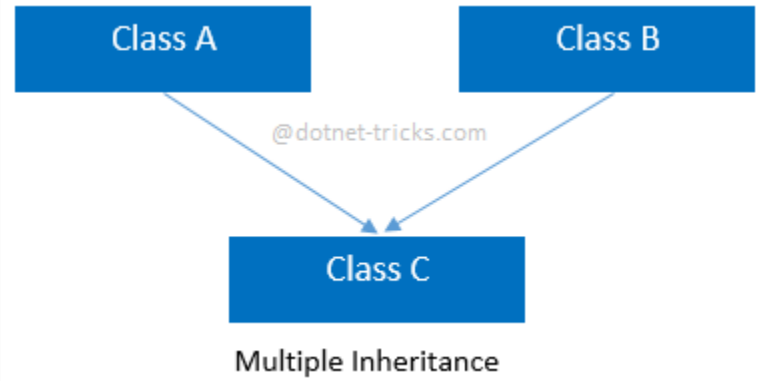
Multi-Level Inheritance

```
1. //Base Class
2. class A
3. {
```

```
4.  public void fooA()
5.  {
6.  //TO DO:
7.  }
8. }
9.
10. //Derived Class
11. class B : A
12. {
13.  public void fooB()
14.  {
15.  //TO DO:
16.  }
17. }
18.
19. //Derived Class
20. class C : B
21. {
22.  public void fooC()
23.  {
24.  //TO DO:
25.  }
26. }
```

3. Multiple inheritance

In this inheritance, a derived class is created from more than one base class. This inheritance is not supported by .NET Languages like C#, F# etc.

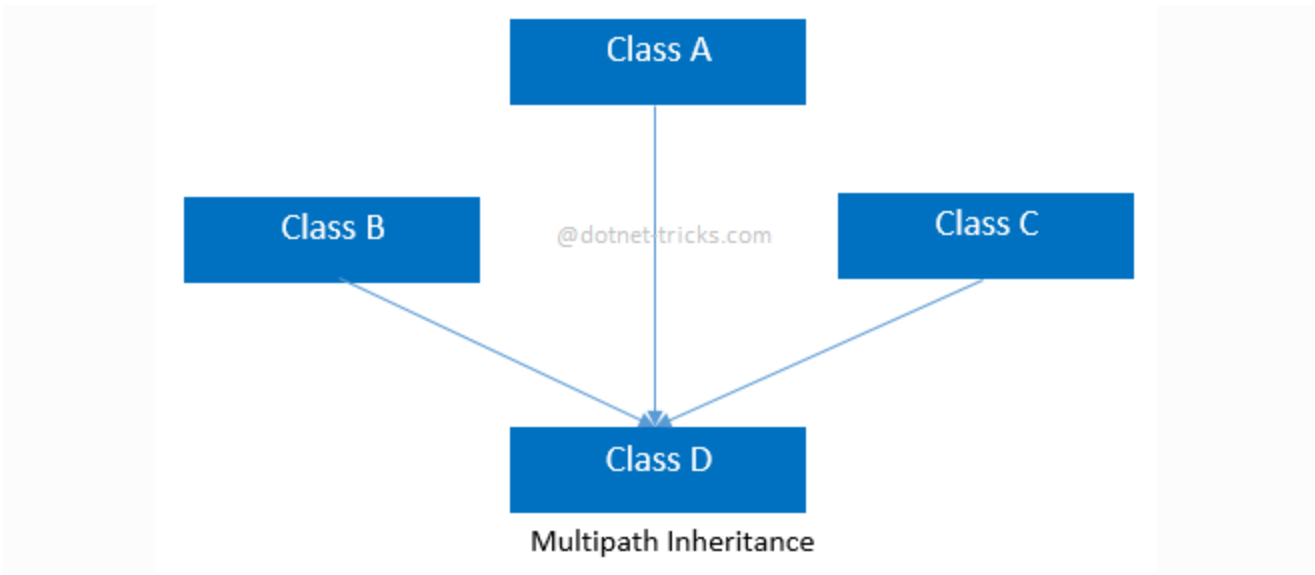


Multiple Inheritance

```
1. //Base Class
2. class A
3. {
4.  public void fooA()
5.  {
6.  //TO DO:
7.  }
8. }
9.
10. //Base Class
11. class B
12. {
13.  public void fooB()
14.  {
15.  //TO DO:
16.  }
17. }
18.
19. //Derived Class
20. class C : A, B
21. {
22.  public void fooC()
23.  {
24.  //TO DO:
25.  }
26. }
```

4. Multipath inheritance

In this inheritance, a derived class is created from another derived classes and the same base class of another derived classes. This inheritance is not supported by .NET Languages like C#, F# etc.



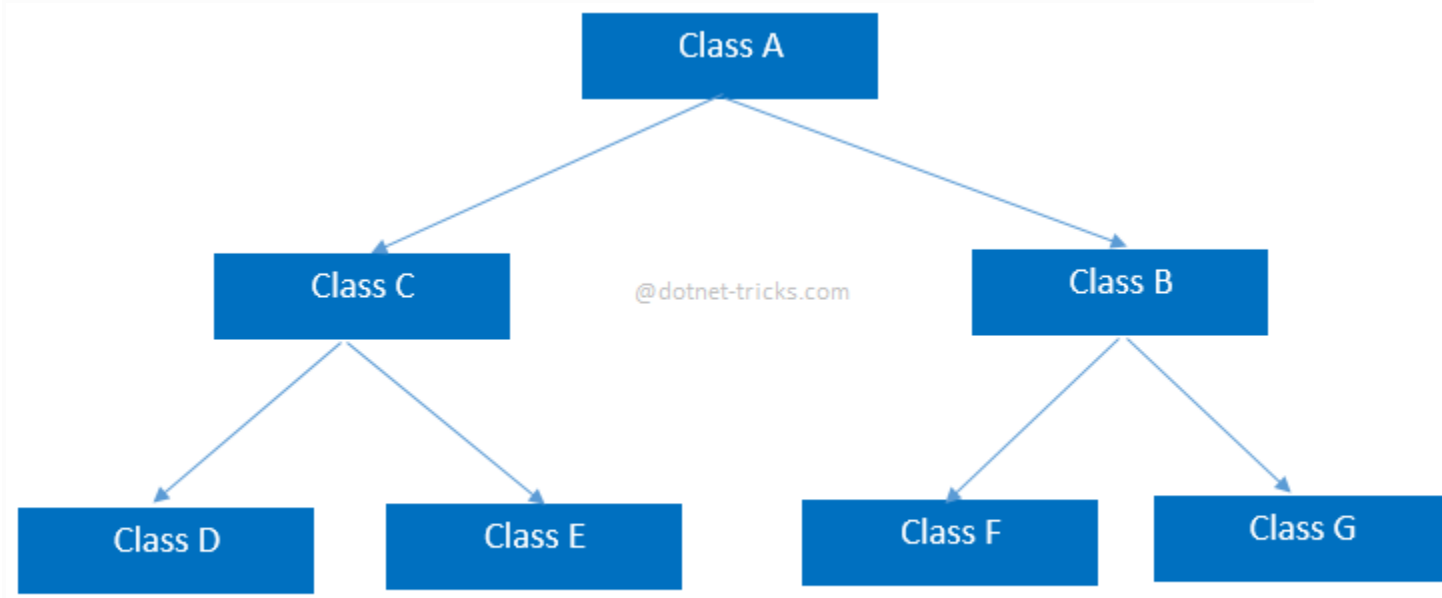
Multipath Inheritance

```
1. //Base Class
2. class A
3. {
4.  public void fooA()
5.  {
6.  //TO DO:
7.  }
8. }
9.
10. //Derived Class
11. class B : A
12. {
13.  public void fooB()
14.  {
15.  //TO DO:
16.  }
17. }
18.
19. //Derived Class
20. class C : A
21. {
22.  public void fooC()
23.  {
```

```
24.   //TO DO:
25.   }
26. }
27.
28. //Derived Class
29. class D : B, A, C
30. {
31.  public void fooD()
32.   {
33.   //TO DO:
34.   }
35. }
```

5. Hierarchical inheritance

In this inheritance, more than one derived classes are created from a single base.



Hierarchical Inheritance

```
1. //Base Class
2. class A
3. {
4.  public void fooA()
5.
6.  {
```

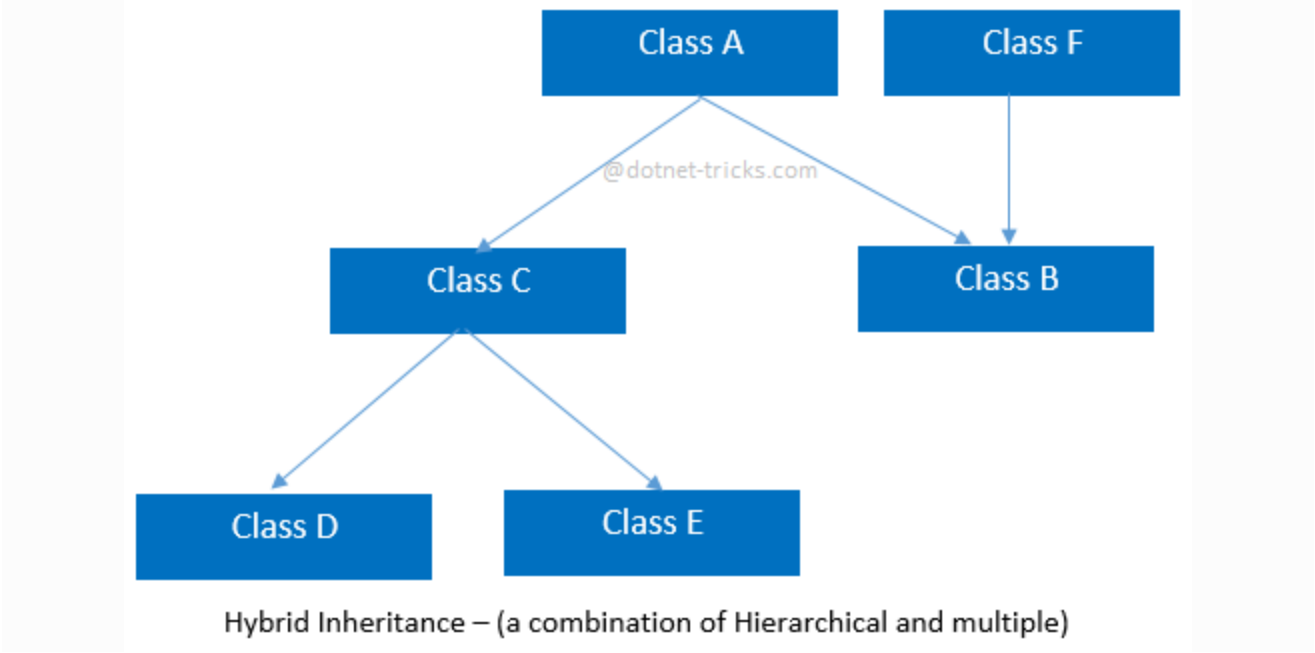
```
7.  //TO DO:
8.  }
9. }
10.
11. //Derived Class
12. class B : A
13. {
14.  public void fooB()
15.  {
16.  //TO DO:
17.  }
18. }
19.
20. //Derived Class
21. class C : A
22. {
23.  public void fooC()
24.  {
25.  //TO DO:
26.  }
27. }
28.
29. //Derived Class
30. class D : C
31. {
32.  public void fooD()
33.  {
34.  //TO DO:
35.  }
36. }
37.
38. //Derived Class
39. class E : C
40. {
41.  public void fooE()
42.  {
```

```
43.  //TO DO:
44.  }
45. }
46.
47. //Derived Class
48. class F : B
49. {
50.  public void fooF()
51.  {
52.  //TO DO:
53.  }
54. }
55.
56. //Derived Class
57. class G :B
58. {
59.  public void fooG()
60.  {
61.  //TO DO:
62.  }
63. }
```

6. Hybrid inheritance

This is combination of more than one inheritance. Hence, it may be a combination of Multilevel and Multiple inheritance or Hierarchical and Multilevel inheritance or Hierarchical and Multipath inheritance or Hierarchical, Multilevel and Multiple inheritance.

Since .NET Languages like C#, F# etc. does not support multiple and multipath inheritance. Hence hybrid inheritance with a combination of multiple or multipath inheritance is not supported by .NET Languages.



Hybrid Inheritance – (a combination of Hierarchical and multiple)

```
1. //Base Class
2. class A
3. {
4.  public void fooA()
5.  {
6.  //TO DO:
7.  }
8. }
9.
10. //Base Class
11. class F
12. {
13.  public void fooF()
14.  {
15.  //TO DO:
16.  }
17. }
18.
19. //Derived Class
20. class B : A, F
21. {
```

```
22.  public void fooB()
23.  {
24.  //TO DO:
25.  }
26. }
27.
28. //Derived Class
29. class C : A
30. {
31.  public void fooC()
32.  {
33.  //TO DO:
34.  }
35. }
36.
37. //Derived Class
38. class D : C
39. {
40.  public void fooD()
41.  {
42.  //TO DO:
43.  }
44. }
45.
46. //Derived Class
47. class E : C
48. {
49.  public void fooE()
50.  {
51.  //TO DO:
52.  }
53. }
```

Source : http://www.dotnet-tricks.com/Tutorial/oops/JaIO211013-Understanding-Inheritance-and-Different-Types-of-Inheritance.html