

# Nesne Tabanlı Programlama

## Bölüm 5

### Kalıtım (inheritance)

---

Dr. Öğr. Üyesi Murat TAŞYÜREK (kayubmprogramlama1@gmail.com)

30 Ekim 2023

Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

## Kalıtım ve Kompozisyon (Composition)

- Yazılım geliştirirken önceden yazılmış sınıflar **kalıtım (inheritance)** ve **kompozisyon (composition)** olarak üzere iki yöntem ile kullanılır.
- Önceden yazılmış sınıfların yeni yazılacak sınıflar içerisinde direk kullanılması **kompozisyon (composition)** olarak adlandırılır.
- Kompozisyon, bir sınıfın diğerini içermesine izin veren sınıflar arasındaki bir ilişki türüdür.
- Kompozisyona genellikle Has-A (B sınıfı A sınıfını içeririyor manasına gelen) ilişkisi denir. B sınıfın araba A sınıfında motor olduğunu düşünürsek, bir arabanın motoru vardır.
- Tıpkı kalıtım gibi, kompozisyon da kodun yeniden kullanılmasına izin verir.

## Kompozisyon (Composition)

- Sınıflar arasında bir ilişki oluşturmak (kompozisyon ilişkisi) için, kullanmak istediğimiz sınıfın içinde yeni bir sınıf nesnesi başlatırız.

2 references

```
class ClassA
```

```
{
```

```
    // class body
```

```
}
```

0 references

```
class ClassB
```

```
{
```

```
    ClassA nesne = new ClassA();
```

```
    //nesne.... ( ClassA sınıfdan metotların çağırılması);
```

```
}
```

## Örnek 1

- Bir arabamızın olduğunu düşünelim.
- Arabamızın motoru vardır ancak başka arabanın da aynı motoru olabilir.
- Nesne tabanlı programlama mantığında düşündüğümüzde az kod yazmak ve yazılan kodu tekrar kullanmak için Araba ve Motor için ayrı ayrı sınıf oluştururuz.
- Araba çalıştığına aslında önce motor çalışmış olur.
- Arabayı durdurduğumuzda önce motor durmuş olur.
- Araba ve motor arasındaki ilişkiyi içeren kodlamayı yapalım.

# Araba ve Motor Class

```
public class Motor
{
    1 reference
    public void MotoruCalistir()
    {
        Console.WriteLine("Motor çalıştı");
    }
    1 reference
    public void MotoruKapat()
    {
        Console.WriteLine("Motor kapatıldı");
    }
}
0 references
public class Araba
{
    Motor arabaninMotoru = new Motor();//Compoustion, Araba has Motor
    0 references
    public void ArabayiCalistir()
    {
        arabaninMotoru.MotoruCalistir();
        Console.WriteLine("Araba çalıştı");
    }
    0 references
    public void ArabayiIstopEt()
    {
        arabaninMotoru.MotoruKapat();
        Console.WriteLine("Araba istop edildi.");
    }
}
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MMDers50rnek1
8  {
9      internal class Program
10     {
11         static void Main(string[] args)
12         {
13             Araba arbm = new Araba();
14             arbm.ArabayiCalistir();
15             arbm.ArabayiIstopEt();
16         }
17     }
18     public class Motor...
19
20     public class Araba...
21 }
```

```
namespace MMDers50rnek1
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Araba arbm = new Araba();
            arbm.ArabayiCalistir();
            arbm.ArabayiIstopEt();
        }
    }
    2 references
    public class Araba
    {
        Motor motor;
        void Calistir()
        {
            Console.WriteLine("Motor çalıştı");
            motor.Calistir();
        }
        void IstopEt()
        {
            Console.WriteLine("Araba çalıştı");
            motor.IstopEt();
        }
        void Kapatildi()
        {
            Console.WriteLine("Motor kapatıldı");
        }
        void IstopEdildi()
        {
            Console.WriteLine("Araba istop edildi.");
        }
    }
}
```



C:\WINDOWS\system32\cmd.exe

```
Motor çalıştı
Araba çalıştı
Motor kapatıldı
Araba istop edildi.
Press any key to continue . . .
```

## Kalıtım (Inheritance)

- Nesne Yönelimli Programlama dillerinde **kalıtım (inheritance)**, bir sınıfta (class) tanımlanmış değişkenlerin ve/veya metotların (fonksiyon, procedure) yeniden tanımlanmasına gerek olmaksızın yeni bir sınıfa taşınabilmesidir. Bunun için yapılan iş, bir sınıftan bir alt-sınıf (subclass) türetmektir.
- Miras yoluyla (kalıtım), bir kez oluşturulmuş olan sınıfın tekrar tekrar kullanılması olanaklı olur. Böylece, programlar daha kısa olur, programın yazılma zamanı azalır ve gerektiğinde değiştirilmesi ve onarılması kolay olur.
- Kalıtım yoluyla yeni bir sınıf, önceden yazılmış başka bir sınıftan türetilir. Üretilen yeni sınıf, türetildiği sınıfın kalıtım yoluyla gelebilen özelliklerine sahip olur.
- Yeni üretilen bir sınıf türetildiği sınıfın özelliklerini değiştirerek kullanabilir. Yeni üretilen sınıfın kendisine ait yeni özellikleri de tanımlanabilir.

## Kalıtım (Inheritance)

- C#'ta kalıtım, mevcut bir sınıftan yeni bir sınıf oluşturmamıza olanak tanır.
- Kalıtım Nesneye Yönelik Programlamanın (OOP) en temel özelliklerinden birisidir.
- Yeni bir sınıfın oluşturulduğu sınıf, temel sınıf (parent or superclass) olarak bilinir. Yeni sınıfa türetilmiş sınıf (child or subclass) denir.
- Türetilmiş sınıf, temel sınıfın özelliklerini, öznitelikleri ve yöntemlerini devralır. Bu, C#'daki kodun yeniden kullanılabilirliğine yardımcı olur.

## Kalıtım (Inheritance)

- C#'ta kalıtımı gerçekleştirmek için : sembolünü kullanırız.

1 reference

```
class Animal
```

```
{
```

```
    // fields and methods
```

```
}
```

```
// Dog inherits from Animal
```

0 references

```
class Dog : Animal
```

```
{
```

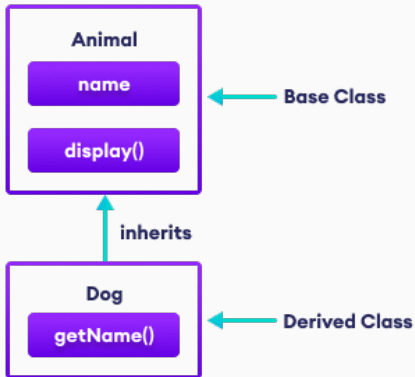
```
    // fields and methods of Animal
```

```
    // fields and methods of Dog
```

```
}
```

# Kalıtım (Inheritance)

- Burada, Animal temel sınıf. Dog sınıfı miras alan yani türetilmiş (derived class) sınıf. Dog sınıfı artık Animal sınıfının alanlarına ve yöntemlerine erişebilir.



# Animal ve Dog Class

// base class

1 reference

class Animal

{

public string name;

0 references

public void display()

{

Console.WriteLine("I am an animal");

}

}

// derived class of Animal

0 references

class Dog : Animal

→ kalıtım

{

0 references

public void getName()

{

Console.WriteLine("My name is " + name);

}

}

→ kalıtım yoluyla gelen özellik

0 references

```
internal class Program
```

```
{
```

0 references

```
static void Main(string[] args)
```

```
{
```

```
    // object of derived class
```

```
    Dog dg = new Dog();
```

```
    // access field and method of base class
```

```
    dg.name = "Kangal";
```

```
    dg.display();
```



Kalıtım yoluyla elde edilen özellik ve fonksionlar

```
    // access method from own class
```

```
    dg.getName();
```

```
}
```

```
}
```

```
// base class
```

1 reference

```
class Animal...
```

```
// derived class of Animal
```

2 references

```
class Dog ...
```

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        // object of derived class
        Dog dg = new Dog();

        // access field and method of base class
        dg.name = "Kangal";
        dg.display();

        // access method from own class
        dg.getName();
    }
}
```

C:\WINDOWS\system32\cmd.exe

```
I am an animal
My name is Kangal
Press any key to continue . . .
```

- Kalıtımın önemini daha iyi anlamak için bir örnek yapalım.
- Kare, diktörgen ve benzeri normal çok genler ile çalışıyoruz.
- Girdiye göre bu çok genlerin alanların ve çevrelerinin değerlerini bulmak istiyoruz.
- Alan hesaplama formülü tüm normal çokgenler için ortaktır. Bu nedenle Normal Çokgen sınıfı ve çevreyi hesaplamak için AlanHesapla() yöntemini oluşturabiliriz.

# Classes

```
public class CokGen
{
    2 references
    protected void AlanHesapla(int width, int lenght) → protected olduğundan sadece alt sınıflardan erişilebilir.
    {
        Console.WriteLine("Çok genin alanı:" + (width * lenght));
    }
}

2 references
public class DikTorgen : CokGen → kalıtım
{
    public int genislik, yukseklik;
    1 reference
    public void AlanCevreHesapla()
    {
        int cevre = 2 * (genislik + yukseklik);
        Console.WriteLine("Çok genin çevresi :" + cevre);
        AlanHesapla(genislik, yukseklik);
    }
}

2 references
public class Kare : CokGen → kalıtım
{
    public int kenar;
    1 reference
    public void AlanCevreHesapla()
    {
        int cevre = 4 * kenar;
        Console.WriteLine("Çok genin çevresi :" + cevre);
        AlanHesapla(kenar, kenar);
    }
}
```

```
namespace NNDers5Ornek4
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Kare kr = new Kare();
            kr.kenar = 5;
            kr.AlanCevreHesapla();

            DikTorgen dk = new DikTorgen();
            dk.yukseklik = 10;
            dk.genislik = 8;
            dk.AlanCevreHesapla();

            CokGen ck = new CokGen();
            ck.

            4 references
            public clas
            2 references
            public clas
            2 references
            public clas
        }
    }
}
```

Metodu protected olduğundan  
üretilen nesnede erişilemez sadece  
kalıtım yapıldığı yerde erişilebilir.

|               |                                                                         |
|---------------|-------------------------------------------------------------------------|
| ★ Equals      | bool object.Equals(object obj) (+ 1 overload)                           |
| ★ GetHashCode | Determines whether the specified object is equal to the current object. |
| ★ ToString    | ★ IntelliCode suggestion based on this context                          |
| ★ GetType     | Note: Tab twice to insert the 'Equals' snippet.                         |
| Equals        |                                                                         |
| GetHashCode   |                                                                         |
| GetType       |                                                                         |
| ToString      |                                                                         |

0 references

`internal class Program``{`

0 references

`static void Main(string[] args)``{``Kare kr = new Kare();``kr.kenar = 5;``kr.AlanCevreHesapla();``DikTorgen dk = new DikTorgen();``dk.yukseklik = 10;``dk.genislik = 8;``dk.AlanCevreHesapla();``CokGen ck = new CokGen();``}``}`

C:\WINDOWS\system32\cmd.exe

Çok genin çevresi :20

Çok genin alanı:25

Çok genin çevresi :36

Çok genin alanı:80

Press any key to continue . . .

- Alan hesabının yapıldığı yeri değiştirelim.
- Sonucun bütün sınıfları etkilediğini görelim.
- Program çalıştığından sonuçların değiştiğini göreceksiniz.
- Nesne yönelimli programlama bunun gibi durumlar için önemlidir. Sadece bir yerde değişiklik yaparsınız o sınıftan üretilen sınıflarda veya nesnelerin tamamında güncelleme yapmış olursunuz.

## Kalıtım ve ilk değer alma sırası

- Kalıtım, yeni oluşturulan sınıfın türetildiği sınıfa ait özellikleri alması ve ayrıca kendisine ait özellikleri tanımlayabilmesidir.
- Kalıtım yoluyla sınıflar oluşturulup daha sonra da nesne oluşturulurken öncelikle türetildiği sınıfın nesnesi oluşturulmaya çalışır.
- Bu işlem en son ilk kalıtım alınan sınıfın kurucu fonksiyonuna kadar gidilir. İlk önce o tetiklenir daha sonra ondan türeyen sınıfların metotları tetiklenir.

- Bengal kaplanı için bir sınıf olduğunu farz edelim. Bengal kaplanı aslında bir kaplandır ve kaplan da aslında bir hayvandır. Önce hayvan sınıfını tanımlayalım, daha sonra hayvan sınıfından kalıtım yoluyla kaplan sınıfını tanımlayalım ve daha sonra kaplan sınıfından kalıtım yoluyla türeyen bir bengal kaplanı sınıfını tanımlayalım.
- Her üç sınıfta constructor metodu olsun.

# Classes

```
public class Hayvan
{
    0 references
    public Hayvan()
    {
        Console.WriteLine("Hayvan sınıfı için tetiklenen yapılandırıcı.");
    }
}

2 references
public class Kaplan : Hayvan → kalıtım
{
    0 references
    public Kaplan()
    {
        Console.WriteLine("Kaplan sınıfı için tetiklenen yapılandırıcı.");
    }
}

3 references
public class BengalKaplani : Kaplan → Kalıtım
{
    1 reference
    public BengalKaplani()
    {
        Console.WriteLine("Bengal kaplanı sınıfı için tetiklenen yapılandırıcı.");
    }
}
```

```
public class Hayvan
{
    0 references
    public Hayvan()
    {
        Console.WriteLine("Hayvan sınıfı için tetiklenen yapılandırıcı.");
    }
}

2 references
public class Kaplan : Hayvan
{
    0 reference
    public Kaplan()
    {
        Console.WriteLine("Kaplan sınıfı için tetiklenen yapılandırıcı.");
    }
}

3 references
public class BengalKaplani : Kaplan
{
    1 reference
    public BengalKaplani()
    {
        Console.WriteLine("Bengal kaplanı sınıfı için tetiklenen yapılandırıcı.");
    }
}
```

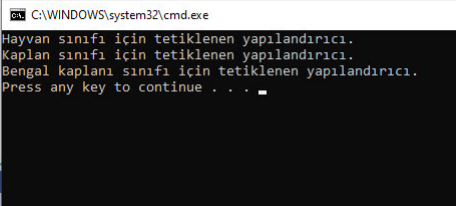


# Kurucu fonksiyonların çalışma sırası

```
Program.cs
NNDers5Ornek5
NNDers5Ornek5.Program
Main(string[])

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace NNDers5Ornek5
8 {
9     0 references
10     internal class Program
11     {
12         0 references
13         static void Main(string[] args)
14         {
15             BengalKaplani kaplan = new BengalKaplani();
16         }
17     }
18     2 references
19     public class Hayvan ...
20     2 references
21     public class Kaplan ...
22     3 references
23     public class BengalKaplani ...
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
namespace NNDers5Ornek5
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            BengalKaplani kaplan = new BengalKaplani();
        }
    }
    2 references
    public class Hayvan...
    2 references
    public class Kaplan...
    3 references
    public class BengalKaplani...
}
```



```
C:\WINDOWS\system32\cmd.exe
Hayvan sınıfı için tetiklenen yapılandırıcı.
Kaplan sınıfı için tetiklenen yapılandırıcı.
Bengal kaplanı sınıfı için tetiklenen yapılandırıcı.
Press any key to continue . . .
```

# Ödev - Notebook Sistemi

- 4 Farklı marka için bir laptop kütüphanesinin oluşturulması istenmektedir.
- Her markada İşlemci, RAM, Anakart, DISK ve Ekran bulunmaktadır.
- Ayrıca bu bilgisayarlara sonradan RAM takılabilmektedir.
- Lenova markaların ekran boyutu 17.3 olmaktadır. Sony bilgisayarların işlemcisi en az İ5 olmaktadır. Dell bilgisayarlarda disk SSD olmaktadır. Asus bilgisayarlar ise sadece i7 serisi üretmektedir.
- Bilgisayarın detay özelliklerine girildiğinde RAM'ın veri yolu hızı, İşlemcinin hızı, diskin okuma yazma hızı var ise Ekran kartı markası paylaşımlı olup olmadığı, monitörün boyutu vb. bütün detay özellikler olmalıdır.
- Fiyatlandırma ise laptopta bulunan özelliklere göre yapılmalıdır. Mesala İ3 5. Nesil 2.4 GHZ RAM'ın fiyatı 100\$'dır. Bu ram A marka notebook takarsa 110 (%10 kar ile), B markası takarsa 109, C Markası takarsa 113 ve D Marka notebook takarsa 115 olarak satmaktadır.
- Nesne yönelimli programlamada kalıtım mantığıyla ilgili projeyi kodlayınız.
- 6 Kasım 06.59'a kadar gönderilenler 8+2 puan, vize tarihine kadar gönderenler 8 puan (kayubmprogramlama1@gmail.com).