(*inheritance*). Bu yöntemde yeni oluşturacağımız sınıfı, daha evvelden yazılmış ve doğruluğu kanıtlanmış olan sınıftan türetilir; böylece yeni oluşan sınıf, türetildiği sınıfın özelliklerine sahip olur; Ayrıca oluşan bu yeni sınıfın kendisine ait yeni özellikleri de olabilir.

Motor.java

```
 1  public class Motor {
 2      private static int motor_gucu = 3600;
 3
 4      public void calis() {
 5          System.out.println("Motor Calisiyor") ;
 6      }
 7
 8      public void dur() {
 9          System.out.println("Motor Durdu") ;
10      }
11  }
```

Şimdi bu *Motor* sınıfını, arabamızın içerisine yerleştirelim;

AileArabasi.java

```
1  public class AileArabasi {
2      private Motor m = new Motor();
3      public void hareketEt() {
4          m.calis();
5          System.out.println("Aile Arabasi Calisti");
6      }
7      public void dur() {
8          m.dur();
9          System.out.println("Aile Arabasi Durdu");
10     }
11     public static void main(String args[]) {
12         AileArabasi aa = new AileArabasi() ;
13         aa.hareketEt();
14         aa.dur();
15     }
16 }
```

AileArabası sınıfının içerisine, *Motor* tipinde global bir alan yerleştirilerek, bu iki sınıf birbirine bağlanmış oldu. *AileArabası* sınıfının hareketEt() ve dur() metotlarında, önce *Motor* sınıfına ait yordamlar (methods) direk olarak çağrıldı.

Motor sınıfının private erişim belirleyicisine sahip olan motor_gucu alanına, *AileArabasi* sınıfının içerisinde ulaşamayız. Bunun nedenlerini bir önceki bölümlerde incelemiştik. *AileArabasi* sınıfı *Motor* sınıfının sadece iki adet public yordamına (*method*) erişebilir: calis() ve dur().

5.2. **Kalıtım**

Kalıtım konusu nesneye yönelik programlamanın (*object oriented programming*) en önemli kavramlarından bir tanesidir. Kalıtım kavramı, kısaca bir sınıftan diğer bir sınıfın türemesidir. Yeni türeyen sınıf, türetilen sınıfın global alanlarına ve yordamlarına (statik veya değil) otomatik olarak sahip olur (*private* olanlar hariç).

Unutulmaması gereken unsur, yeni türeyen sınıf, türetilen sınıfın `private` global alanlarına ve yordamlarına (statik veya değil) otomatik olarak sahip olamaz.
Ayrıca yeni türeyen sınıf eğer türetilen sınıf ile ayrı paketlerde ise yeni türeyen sınıf, türetilen sınıfın sadece `public` ve `protected` erişim belirleyicisine sahip olan global alanlarına (statik veya değil) ve yordamlarına (statik veya değil) otomatik olarak sahip olur.

Kedi sınıfından türeyen *Kaplan* sınıfı… İki sınıf arasındaki ilişkiyi şöyle tarif edebiliriz, her *Kaplan* **bir** *Kedi*'dir. Yani her kaplan kedisel özellikler taşıyacaktır ama bu özelliklerin üzerine kendisine bir şeyler eklemiştir

Kedi.java *

```
1  class Kedi {
2
3          protected int ayakSayisi = 4 ;
4          public void yakalaAv() {
5                  System.out.println("Kedi sinifi Av yakaladi");
6          }
7
8          public static void main(String args[]) {
9                  Kedi kd= new Kedi() ;
10                 kd.yakalaAv() ;
11         }
12 }
13
14 class Kaplan extends Kedi {
15
16         public static void main(String args[] ) {
17                 Kaplan kp = new Kaplan();
18                 kp.yakalaAv();
19                 System.out.println("Ayak Sayisi = " + kp.ayakSayisi);
20         }
21 }
```

Kaplan sınıfı *Kedi* sınıfından türemiştir. Görüldüğü üzere *Kaplan* sınıfının içerisinde ne yakalaAv() yordamı ne de ayaksayisi alanı tanımlanmıştır. *Kaplan* sınıfı bu özelliklerini kendisinin ana sınıfı olan *Kedi* sınıfından miras almıştır.

Kedi sınıfının içerisinde tanımlanmış ayaksayisi alanı, protected erişim belirleyicisine sahiptir. Bunun anlamı, bu alana aynı paket içerisinde olan sınıflar ve ayrı paket içerisinde olup bu sınıftan türetilmiş olan sınıfların erişebileceğidir. Böylece *Kaplan* sınıfı ister *Kedi* sınıfı ile aynı pakette olsun veya olmasın, *Kedi* sınıfına ait global int ilkel (*primitive*) tipindeki alanına (ayaksayisi) erişebilir.

Her sınıfın içerisine main yordamı yazarak onları tek başlarına çalışabilir bir hale sokabiliriz (*standalone application*); bu yöntem sınıfları test etmek açısından iyidir. Örneğin *Kedi* sınıfını çalıştırmak için komut satırından java Kedi veya *Kaplan* sınıfını çalıştırmak için java Kaplan yazılması yeterli olacaktır.