tasarlanmakta ve o nesnelerinde özellikleri ve olayları kısaca tanımlayıcıları ortaya konmaktadır. Sonrasında ise oluşturulan bu nesneler birleştirilerek uygulamaları meydana getirmektedirler.

Nesnelerden meydana gelen uygulamalar ise nesnelerin özelliklerinden dolayı daha *müdahale edilebilir* olmaktadır.

…1960'lı yılların sonuna doğru ortaya çıkan bu yaklaşım, o dönemin yazılım dünyasında beliren bir bunalımın sonucudur. Yazılımların karmaşıklığı ve boyutları sürekli artıyor, ancak belli bir nitelik düzeyi korumak için gereken bakımın maliyeti zaman ve çaba olarak daha da hızlı artıyordu. NYP'yi bu soruna karşı bir çözüm haline getiren başlıca özelliği, yazılımda birimselliği (modularity) benimsemesidir. NYP ayrıca, bilgi gizleme (information hiding), veri soyutlama (data abstraction), çok biçimlilik (polymorphism) ve kalıtım (inheritance) gibi yazılımın bakımını ve aynı yazılım üzerinde birden fazla kişinin çalışmasını kolaylaştıran kavramları da yazılım literatürüne kazandırmıştır. Sağladığı bu avantajlardan dolayı, NYP günümüzde geniş çaplı yazılım projelerinde yaygın olarak kullanılmaktadır.

NYP'nın altında yatan birimselliğin ana fikri, her bilgisayar programının (izlence), etkileşim içerisinde olan birimler veya nesneler kümesinden oluştuğu varsayımıdır. Bu nesnelerin her biri, kendi içerisinde veri işleyebilir ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilir. Hâlbuki NYP'den önce var olan tek yaklaşımda (Yordamsal programlama), programlar sadece bir komut dizisi veya birer işlev (fonksiyon) kümesi olarak görülmekteydiler. (Nesne Yönelimli Programlama - Vikipedi)

4.2. Nesne Yönelimli Programlama Tekniğinin Avantajları

!*Nesne Yönelimli Programlama Tekniği* ili *Görsel Programlama* kavramlarını karıştırmamak gerekmektedir. Şöyle söyleyebiliriz: Nesne Yönelimli Programlama tekniği sağladığı avantajlardan dolayı görsel programlama sürecinde yararlanacağımız başta gelen tekniklerden olacaktır.

NYP'nin sağladığı avantajlara da kısaca değinmek gerekirse eğer; yukarıdaki Wikipedi'adan yaptığımız alıntıda da bahsedilen özelliklerine göz atmak gerekmektedir. Zira avantajları bu özellikleri içerisinde saklı durumdadır.

NYP tekniği ile oluşturulan nesneler temelde üç özelliğe sahiptirler:

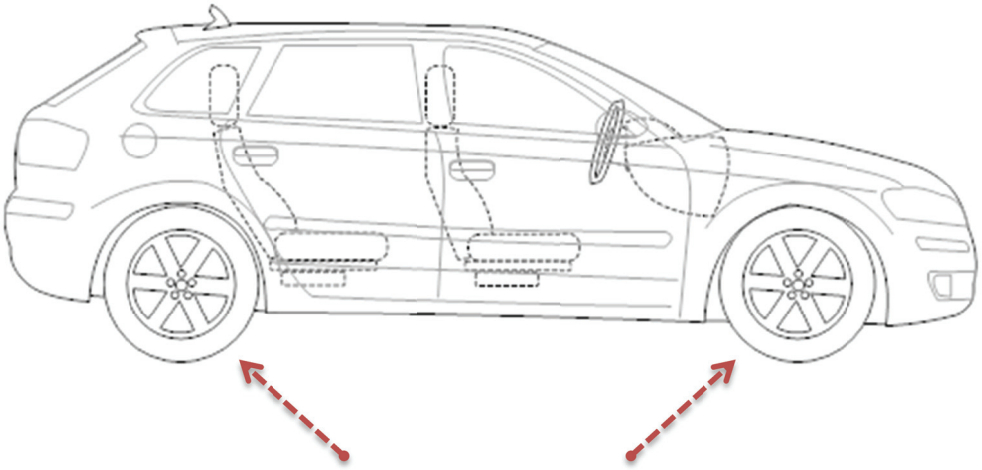
- Bilgi gizleme (information hiding), bazen Veri Soyutlama (data abstraction) da denilebilmektedir,
- Kalıtım (inheritance) ve
- Çok biçimlilik (polymorphism).

İşte tüm bu özelliklerinden dolayı da nesnelerden oluşturulmuş uygulamalar yukarıda da belirttiğimiz gibi daha müdahale edilebilir yapılar olarak karşımıza çıkmaktadırlar.

4.2.1. Nesnelerin Bilgi Gizleme Özelliği

Örneğin aşağıdaki gibi irili ufaklı ve belki de binlerce nesneden oluşan bir otomobil sistemini düşündüğümüzde tüm bu nesnelerin bir şekilde belli bir amaç için bir araya

getirildikleri ve aralarında da bir uyumun –ahenkli bir çalışmanın- olduğu aşikârdır. Zira siz direksiyonu *sola* çevirdiğinizde tekerlekler *sağa* gitmek isteseydi otomobil sistemini oluşturan nesneler arasındaki bir uyumdan söz edilemezdi. Oysaki bildiğimiz gibi günümüz teknolojisiyle donatılmış araçlar bırakın direksiyonu sola çevirdiğinizde tekerleklerin sola yönelmesini, o esnada aracın değişen yeni ağırlık merkezini de hesaba katarak ilgili tekerleklerin fren sistemini kontrol edebilen yapılarla donatılmış durumdadırlar. İşte tüm bunlar otomobili oluşturan nesnelerin aralarındaki uyumlu çalışmadan kaynaklanmaktadır.



Bir otomobil sistemini oluşturan tüm nesnelerin görevleri ve birbirleriyle olan ilişkiler önceden belirlenmiş durumdadır.

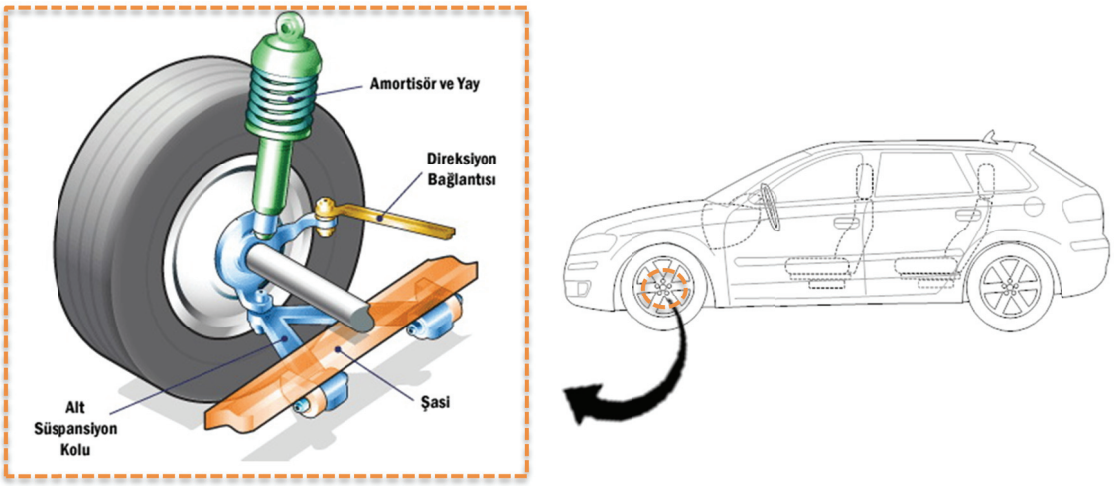
Birbirleriyle bu kadar uyumlu çalışan nesneler ise tek tek düşünüldüğünde kendi içlerinde ayrı bir dünyadır: Örneğin otomobilin lastikleri.

Otomobilin lastikleri de kendi içerisinde belki de yüzlerce nesneden oluşmaktadır ve geliştirilmeye de son derece açıktırlar. Yol tutuşlarındaki artıştan tutun da ömürlerinin uzunluğuna varıncaya kadar otomobil lastiklerinde bir sürü iyileştirmeye gidilebilir.

? İşte burada ki en önemli sorulardan bir tanesi de bu örneğe göre, lastiklerde yapılan iyileştirmeler tüm sistemi yani otomobili nasıl etkiler?

Şüphesiz olumlu yönde etkiler, zira kim istemez ki yola daha iyi tutunan uzun ömürlü lastikleri demi☺ Peki, her lastikleri değiştirdiğimizde aracımızı da değiştiriyor muyuz? Değiştirmiyoruz tabii!

İşte Nesne Yönelimli Programlama tekniğinin arkasında yatan mantıklardan biriside budur. Uygulamalarımızı meydana getiren nesneler belli bir amaç için bir arada çalışabilirlerken kendi içlerinde de *adeta ayrı bir dünya gibidirler*. İşin iyi tarafı ise tüm sistemin, ilgili nesnenin içeriğiyle ilgilenmesine gerek yoktur. Yeter ki aralarındaki bağlantı noktaları iyi belirlenmiş ve ilgili nesnenin de tüm sistem içerisinde ki kendisine düşen görevi layıkıyla yerine getiriyor olsun.



Yine otomobil lastiklerinden yola çıkacak olursak yukarıdaki şekilde de gösterildiği gibi lastiklerin otomobilin şasisine, amortisörlere ve direksiyona bağlantısı önceden ayarlanmıştır. Hatta öyle ki biz son kullanıcılar olarak, bu bağlantıları bilmek zorunda bile değilizdir. Bizi ilgilendiren kısım daha çok, lastiği araca tutturacak olan somunların sıkılmasıdır. Aslına bakarsanız bağlantılar tamam olduktan sonrada, tüm sistemin ilgili nesnenin içeriğiyle ilgili bir fikri yoktur.

Buradan şu noktaya gelmek istiyorum: Aynen yukarıda verdiğimiz örnekteki gibi NYP tekniğinde de uygularımız birden fazla nesneden oluşacak ve bu nesneler kendi aralarında bir şekilde haberleşebiliyor olacaktır. Yine de tüm sistemin ilgili nesnenin içeriğiyle ilgili *doğrudan bir müdahalesi ise söz konusu değildir*. Çünkü dediğimiz gibi "*her bir nesne kendi içinde adeta ayrı bir dünya gibi çalışmaktadır*" ve dolayısıyla nesnenin kendi içindeki bu çalışmadan tüm sistem doğrudan haberdar değildir. İşte bu da nesnelerin *Bilgi Gizleme* veya *Veri Soyutlama* özelliklerine karşılık gelmektedir.

Daha da basit düşünecek olursak, yine nesnelerin bilgi gizleme özellikleriyle ilgili olarak çay-kahve otomatları örnek olarak gösterilebilir. Bu tür cihazlarda biz son kullanıcılar olarak makinenin nasıl çalıştığıyla ilgili olarak çok da fikir sahibi değilizdir. *Nasıl çalıştığıyla ilgili olarak* derken bize hitap eden *arayüzünü* değil de *makinenin iç çalışmasını* kastediyorum. Bizi ilgilendiren kısım ise elimizdeki bozuk paralarla çayımızı nasıl alabileceğimizdir. Bunun içinde genellikle elimizdeki bozuk paralardan yeterli bir kısmını bize gösterilen bir hazneden makinenin içine atarız. İstediğimiz içeriği de belirledikten sonra belki şeker miktarını falan girerek makineden istediğimizi alabiliriz. Tabii bu arada şansımız varsa para üstünü de alabiliriz belki☺ Fakat bu süreçte makinenin içinde olan biten konusunda ise bihaberizdir (habersizizdir).

İşte bu örneğimizde de varmak istediğimiz nokta, daha doğrusu nesnelerle ilgili olarak altını çizmek istediğimiz özellik nesnelerin bilgi gizleyebilme yetenekleridir.

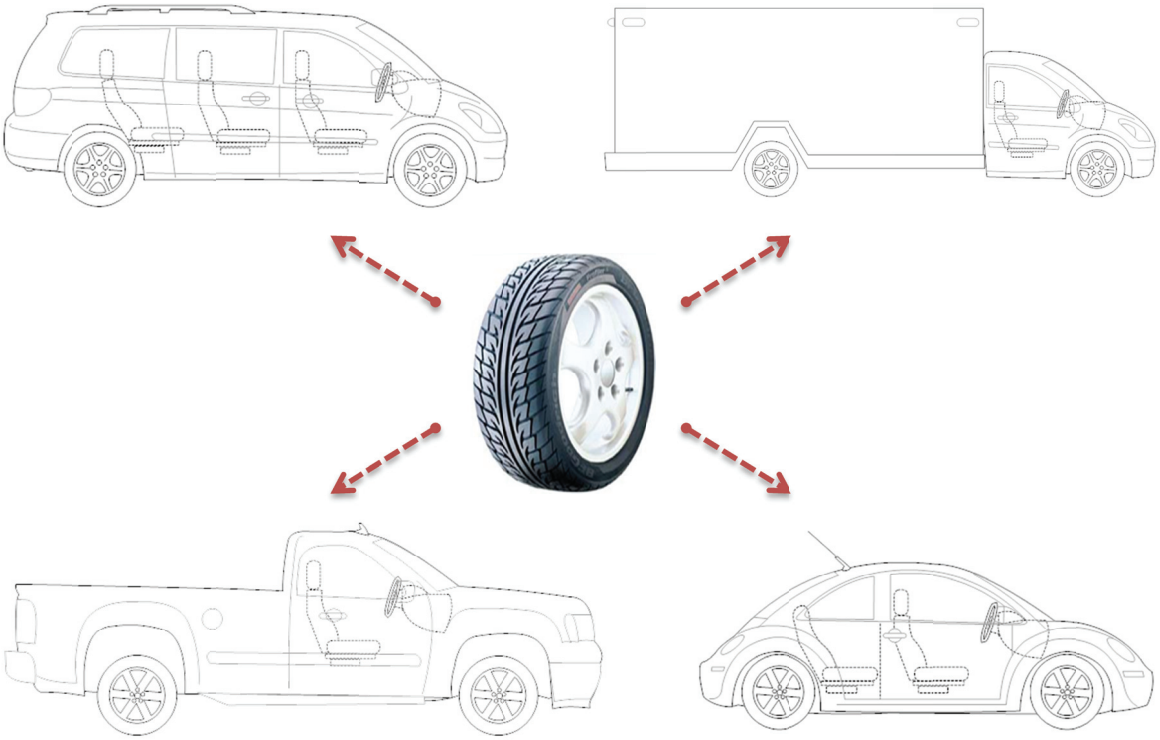
!Nesneler kendi içlerindeki çalışmayla ilgili olan verileri genellikle dış dünya ile paylaşmadan kendi içlerinde işlerler. Dış dünyadan kasıt ise bazen son kullanıcı olabilirken bazen de bağlı bulunduğu sistemin ta kendisidir. Ancak yukarıda ki çay-kahve otomatları örneğinde olduğu gibi dış dünyaya açılan da bir arayüzleri vardır. Otomobil lastiklerinin de bir şekilde otomobile bağlı olmaları gibi.

Bir yandan da nesnelerin bu özellikleri tüm sistemi müdahale edilebilir kılmaktadır. Yine yukarıdaki örneklerimizde olduğu gibi otomobilleri geliştirmek için otomobilleri oluşturan nesnelerde iyileştirmelere gidilebilmektedir. Düşünün, otomobilinizin lastiklerini daha iyi yol tutabilir hale getirdiğinizde tüm sistem bundan nasıl etkilenir veya çay-kahve makinesinin -tamam kendi içinde nasıl çalıştığını bilmiyoruz fakat bu aleti yapan kişiler- aleti arayüzünü değiştirmeseler bile bizim isteklerimize daha hızlı tepki verebilen bir hale getirdiklerinde buna kim itiraz eder ki demi☺

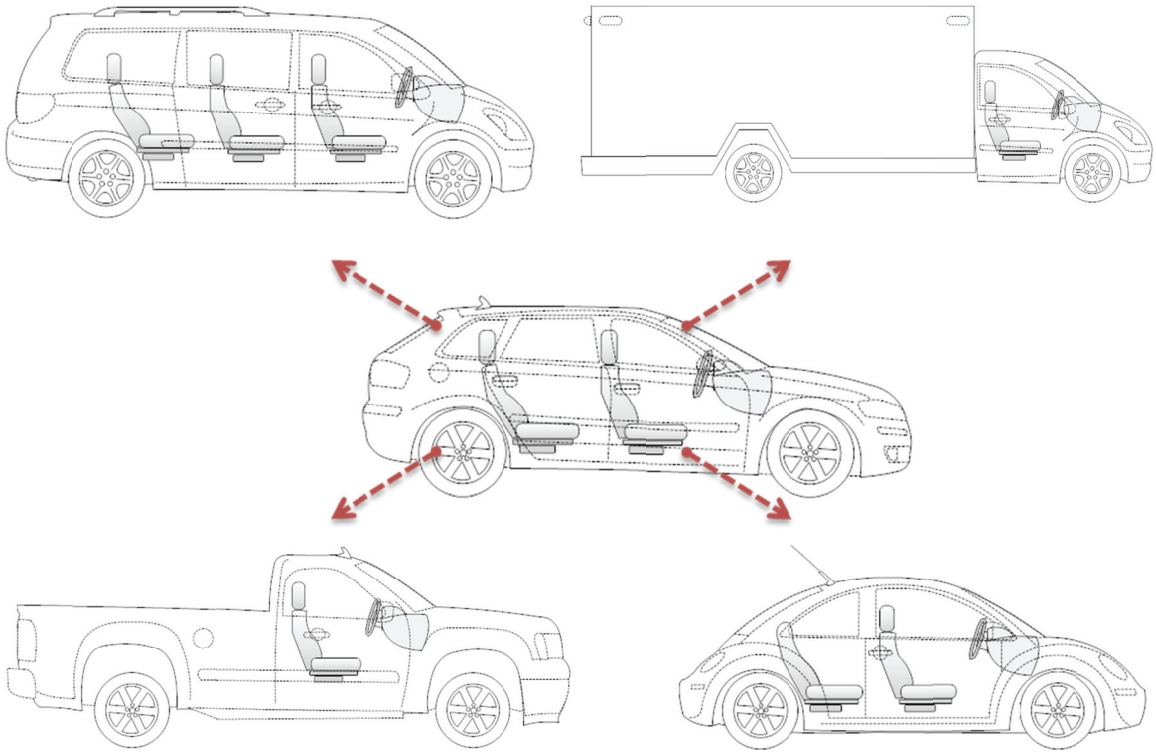
İşte Delphi ile NYP tekniğini kullanarak geliştirdiğimiz uygulamalarda da durum böyledir. Tüm uygulamayı meydana getiren nesneler de zamanla kendi içlerinde geliştirilme yoluna gidilecektir. Hatta nesnelerin yazarları (programcıları) farklı olsalar bile. Kimse

birbirinin nesnesinin çalışması hakkında hiçbir fikir sahibi olmasa bile herkes kendi geliştirdiği nesneyi güncellediğinde tüm uygulama bundan olumlu etkilenecektir ve bu da daha müdahale edilebilir bir yapı demektir.

Kaldı ki oluşturduğumuz ve kendi içlerinde ayrı bir dünya olan bu nesneleri sadece tek bir uygulamada da kullanmak zorunda değiliz.



Yukarıdaki şekilde de anlatılamaya çalışıldığı gibi oluşturduğumuz bir lastik nesnesini küçük rötuşlarla kolaylıkla diğer uygulamalara da taşıyabiliriz.



Keza koltuklar, direksiyon sistemi ve hava yastıkları gibi pek çok sistem küçük rötuşlarla diğer uygulamalarda da kullanılabilir olmaktadırlar. Tüm sistemi oluşturan programcının artık koltuğun kumaşının terletip terletmeyeceğine kafa yorması gerekmez. O koltuğu programlayan programcının düşünmesi gereken bir konudur bu. Tüm sistemi programlayan programcı daha çok koltuğun bağlantı noktaları ve sisteme entegrasyonu üzerinde kafa yormaktadır.

4.2.2. Nesnelerin Kalıtım Özelliği

NYP tekniğinin diğer önemli bir avantajı da nesnelerin sahip oldukları kalıtım özelliğinden ileri gelmektedir.

Nesneler, sahip oldukları bu kalıtım özellikleri sayesinde temel bazı özelliklerini ebeveynlerinden kalıtım yoluyla devralabilmektedirler.

Bu sayede de uygulamalar daha kolay oluşturulabilmekte, düzeltilebilmekte ve daha kolay da genişletilebilmektedir. Tüm bunlar da zaten yine sistemlerin müdahale edilebilir olmalarına hizmet eden özelliklerdir.

Örneğin bir *süper market* için geliştireceğiz *stok takip uygulamasında* NYP tekniğini tercih etmeniz durumunda süper marketteki *her bir ürünü bir nesne olarak oluşturmak* iyi bir çözüm olacaktır.