Sınıf Diyagramları

Sınıf diyagramı statik bir diyagramdır. Bir uygulamanın statik görünümünü temsil eder. Sınıf diyagramı sadece bir sistemin farklı yönlerini görselleştirmek, açıklamak ve belgelemek için değil aynı zamanda yazılım uygulamasının yürütülebilir kodunu oluşturmak için kullanılmaktadır. Sınıf diyagramı bir sınıfın niteliklerini ve işlemlerini ve ayrıca sistem üzerinde uygulanan kısıtlamaları açıklar. Sınıf diyagramları nesne odaklı dillerin modellemesinde yaygın olarak kullanılmaktadır, çünkü bunlar doğrudan nesne yönelimli dillerle eşleştirilebilen tek UML diyagramlarıdır.

Sınıf diyagramı, sınıfların, arayüzlerin, ortaklıkların, işbirliklerinin ve kısıtlamaların bir koleksiyonunu gösterir. Yapısal diyagram olarak da bilinir.

Amaç:

Sınıf diyagramının amacı, bir uygulamanın statik görünümünü modellemektir. Sınıf diyagramları, nesne yönelimli dillerle doğrudan eşleştirilebilen ve inşaa anında yaygın olarak kullanılabilen tek diyagramlardır. Aktivite diyagramı, sıralama diyagramı gibi UML diyagramları sadece uygulamanın akış akışını verebilir ancak sınıf diyagramı biraz farklıdır. Bu kodlayıcı topluluğunda en popüler UML diyagramıdır. Sınıf diyagramının amacı şu şekilde özetlenebilir:

- Bir uygulamanın statik görünümünün analizi ve tasarımı.
- Bir sistemin sorumluluklarını açıklayın.
- Bileşen ve dağıtım diyagramları için temel.
- İleri ve geri mühendislik.

Sınıf Diyagramları Nasıl Çizilir?

Sınıf diyagramları, yazılım uygulamalarının oluşturulması için kullanılan en popüler UML diyagramlarıdır. Dolayısıyla sınıf diyagramının çizim prosedürünü öğrenmek çok önemlidir. Sınıf diyagramları, çizim sırasında göz önüne alınması gereken özelliklere sahiptir, ancak burada diyagram üst düzey bir görünümden ele alınacaktır. Sınıf diyagramı temel olarak sistemin statik görünümünü grafik olarak gösterir ve uygulamanın farklı yönlerini gösterir. Dolayısıyla sınıf diyagramlarından oluşan bir koleksiyon tüm sistemi temsil eder. Sınıf diyagramı çizilirken aşağıdaki noktalar hatırlanmalıdır:

- Sınıf diyagramının adı, sistemin yönünü tanımlamak için anlamlı olmalıdır.
- Her bir unsur ve bunların ilişkileri önceden belirlenmelidir.
- Her sınıfın sorumlulukları (nitelikleri ve yöntemleri) açıkça tanımlanmalıdır.
- Her sınıf için minimum özellik sayısı belirtilmelidir. Çünkü gereksiz özellikler diyagramı karmaşık hale getirecektir.
- Şemanın bazı yönlerini tanımlamak için gerekli olan notları kullanın. Çünkü çizimin sonunda geliştirici / kodlayıcı için anlaşılabilir olmalıdır.

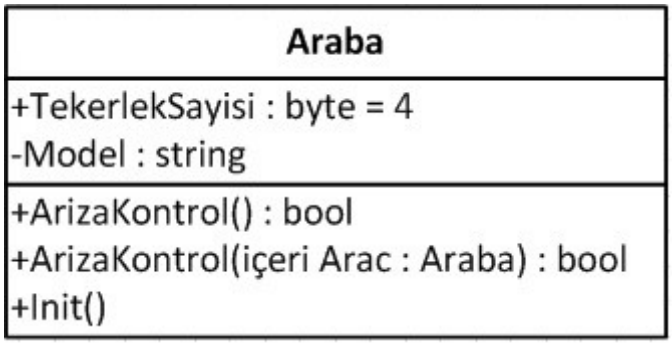
- Son olarak, son şekli vermeden önce, diyagram doğru yapmak için mümkün olduğunca çok kez düz kâğıda ve yeniden işleme tabi tutulmalıdır. Şimdi aşağıdaki diyagram bir uygulamanın Sipariş Sistemi örneğidir. Bu, uygulamanın belirli bir yönünü açıklar.

Sınıf diyagramı statik bir diyagramdır ve bir sistemin statik görünümünü modellemek için kullanılır. Statik görünüm, sistemin kelime dağarcığını tanımlar. Sınıf diyagramı ayrıca bileşen ve dağıtım diyagramlarının temeli olarak da düşünülür. Sınıf diyagramları yalnızca sistemin statik görünümünü görselleştirmek için değil, aynı zamanda herhangi bir sistemin ileri ve geri mühendisliği için yürütülebilir kodu oluşturmak için kullanılır. Genellikle UML diyagramları nesneye yönelik programlama dilleri ile doğrudan eşleşmez, ancak sınıf diyagramı bir istisna oluşturur. Sınıf diyagramı, Java, C ++ vb. Gibi nesne yönelimli dillerle eşlemeyi açıkça gösterir. Dolayısıyla, pratik deneyimlerden sınıf diyagramı genellikle inşaat amaçlı kullanılır. Kısacası, sınıf diyagramları aşağıdakiler için kullanılır:

- Sistemin statik görünümünü tanımlama.
- Statik görünüm unsurları arasındaki işbirliğini gösterme.
- Sistem tarafından gerçekleştirilen işlevleri tanımlamak.
- Nesneye yönelik diller kullanarak yazılım uygulamalarının oluşturulması.

1. Sınıf Tanımlaması

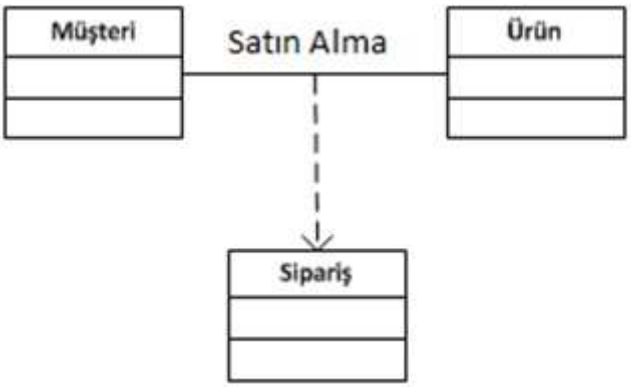
UML de sınıf tanımlama dikdörtgen şekli ile gösterilir. Dikdörtgenin en üstünde sınıfın adı, altında özellikleri ve onunda altında fonksiyonlar gösterilir.



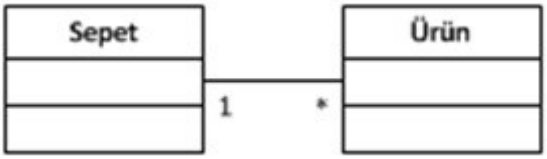
- Yukarıdaki sınıf gösteriminde Araba sınıfımızın adıdır. TekerlekSayisi ve Model sınıfın özellikleridir. Özelliğin tipi :byte ve :string şeklinde belirtilmiştir. TekerlekSayisi özelliğinin default değeri 4 olarak belirtilmiştir. Özellik ve fonksiyonların başındaki + ve – işaretleri de erişim seviyesini belirtir.
- + işareti public (diğer sınıflardan da erişilebilir) seviyesinde olduğunu gösterir.
- # işareti protected (sınıf içinden ve kendisinden türeyen sınıflardan erişilebilir) seviyesinde olduğunu gösterir.
- - işareti private (sadece sınıf içinden erişilebilir) seviyede olduğunu belirtir.
- Araba sınıfında 3 fonksiyon tanımlanmıştır. 1. ArizaKontrol fonksiyonu geriye bool değer döndürür ve parametre almaz. 2. ArizaKontrol fonksiyonu ise Araba sınıfı parametre alır ve geriye bool değer döndürür. Init fonksiyonu ise parametre almaz ve geriye değer döndürmez.

2. Sınıflar Arası İlişki (Assocation)

Sınıflar arasındaki ilişkiler çizgi ile gösterilir ve çizginin üstüne ilişki şekli yazılır. Sınıflar arası ilişkiler bire bir, bire çok, bire n gibi olabilir. Örneğin aşağıdaki gösterimde e-ticaret sisteminde ki Müşteri – Ürün arasında ki Satın Alma ilişkisi gösterilmiştir.



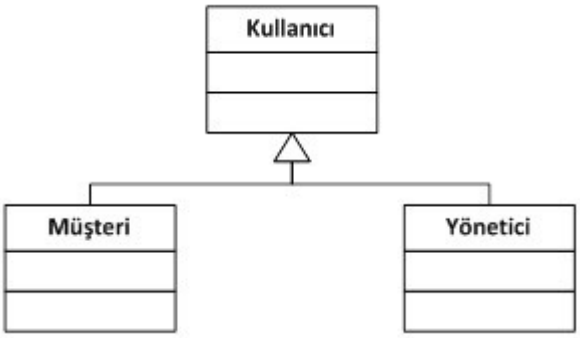
Yukarıda ki diyagrama göre müşteri ve ürün arasında Satın Alma ilişkisi vardır ve bunun için de Sipariş oluşturması gerekir.



Bu diyagramda ise bir sepette n tane ürün olabileceği gösterilmiştir. Bunun gibi bire 1, n e n gibi ilişkiler de belirtilebilir. Sınıfın kendi ile olan ilişkisi de olabilir. Örneğin Müşteri Temsilcisi ve Müşteri. Bunların ikisi de Kişi olmasına rağmen bir Müşteri Temsilcisinin birden fazla Müşteri den sorumludur. Bu durumu aşağıdaki gibi belirtebiliriz.

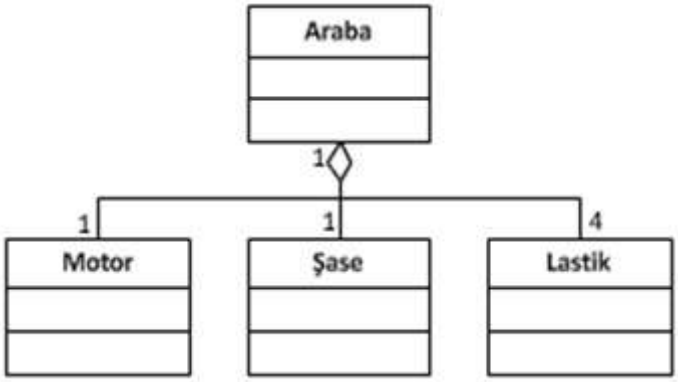
3. Türetme (Inheritance)

Örneğin projemizde Müşteri ve Yönetici sınıflarımız var ve bu sınıfları Kullanıcı sınıfından türettik. Bu yapıyı aşağıdaki gibi gösterebiliriz.

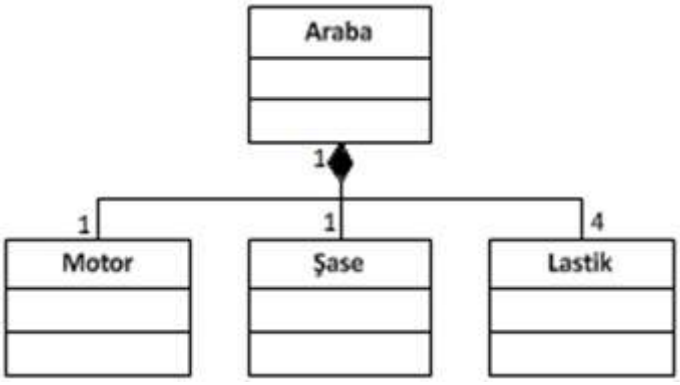


4. İçerme (Aggregations)

Bir sınıf başka parçalardan oluşuyor ise aralarındaki ilişkiye Aggregation denir. Örneğin Araba sınıfını 1 Motor, 1 Şase ve 4 Lastik sınıflarının oluşturduğunu düşünürsek aralarındaki ilişki aşağıdaki gibi gösterilir.

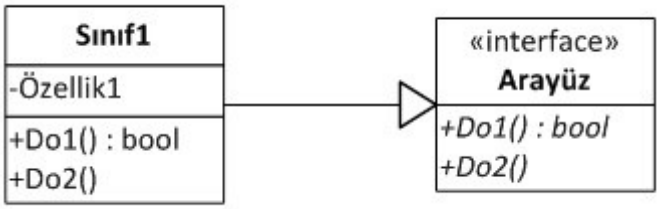


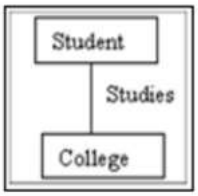
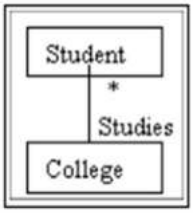
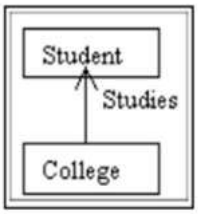
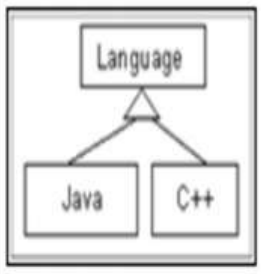
Asıl sınıf üretildiğinde parçaları da üretilecek ise bu ilişkiye Composite denir. Eğer Araba sınıfı oluşturulduğunda Motor, Şase ve Lastik sınıfları da oluşturulacak ise içi boş dörtgen dolu olarak gösterilir.



4. Arayüz(Interface)

Arayüzler de sınıfa benzer şekilde gösterilir. Fakat arayüzlerde özellik tanımlaması yoktur ve <interface> ile arayüz olduğu belirtilir. Sınıf ve arayüz arasındaki ilişki kesik çizgi ve arayüzün olduğu kısımda içi boş üçgen şeklinde gösterilir.



No	İlişki	Örnek Gösterim	Açıklama
1	Bağıntı(Association)		İki sınıfın herhangi bir şekilde birbirleriyle iletişim kurmasıdır. Örneğin : Öğrenci ve Okul arasındaki ilişki
1.a	Çokluk (Multiplicity)		İki sınıf arasındaki 1:n ilişkidir. Birden fazla öğrencinin okul ile olan ilişkisi olarak çıklanabilir. Şekildeki * işareti 1:n ilişkiyi göstermektedir.
1.b	Yönlü Bağıntı (Directed Association)		Sınıflar arasındaki tek yönlü ilişkiyi gösterir. Ok işareti ile gösterilir.
1.c	Dönüşlü Bağıntı(Reflexive Association)	Belirgin bir çizimi yoktur.	Bir sınıfın kendisiyle kurduğu ilişkidir.Bu tür ilişkiler genellikle bir sınıfın sistemde birden fazla rolü varsa ortaya çıkar
2	Kalıtım/Genelleme (Inheritance/Generalization)		Bu ilişki, bir nesnenin bir diğerinin özel bir türü olduğu gerçeğini modellemek için kullanılır.

Kaynaklar

http://harunozer.com/makale/uml_sinif_diyagrami_class_diagram.htm

https://www.tutorialspoint.com/uml/uml_class_diagram.htm

Selic, Bran. "Using UML for modeling complex real-time systems." Languages, compilers, and tools for embedded systems. Springer Berlin Heidelberg, 1998.

http://muhendislik.istanbul.edu.tr/endustri/wp-content/uploads/2014/10/Ders3.pdf