

Tatil Sitesi Yönetim Sistemi: Proje Analizi ve Katman Karşılaştırması

1. Projemin Temel Yapısı ve Benzer Projelerle Karşılaştırma

1.1. Projemin Dosya ve Katman Yapısı

- Kullanıcı Arayüzü (UI) Katmanı
 - Form1.cs: Tüm kullanıcı işlemlerinin ve arayüzün yönetildiği ana form.
 - İçerdeği başlıca metotlar: MekanlarıYukle, OturanlarıYukle, OdemeleriYukle, KullanımKayitlarınıYukle, BorcOde, KullanımKaydiEkle/Sil, OturanEkle/Sil/Guncelle, MekanEkle/Sil/Guncelle.
 - UI katmanı, DosyaIslemleri katmanını kullanarak veri okuma/yazma işlemlerini tetikler.
- İş Mantığı Katmanı
 - KullanımKaydi.cs, Odeme.cs, Mekan.cs, Oturan.cs, AileReisi.cs, Misafir.cs: Sistemin temel iş nesneleri ve kuralları.
 - Her sınıf, ilgili varlığın (ör. oturan, mekan, kullanım kaydı) özelliklerini ve ilişkilerini tutar.
 - KullanımKaydi, Oturan ve Mekan ile ilişkilidir; Odeme, Oturan ile ilişkilidir.
- Veri/Dosya Yönetimi Katmanı
 - DosyaIslemleri.cs: Tüm CRUD işlemleri ve TXT dosya yönetimini sağlar.
 - Başlıca metotlar: MekanlarıOku/Ekle/Sil/Guncelle, OturanlarıOku/Ekle/Sil/Guncelle, OdemeEkle/Guncelle, KullanımKaydiEkle/KayitlarınıOku.
 - UI ve iş mantığı katmanları, veri işlemleri için bu katmanı kullanır.
- Veri Dosyaları
 - Mekan.txt, Data.txt, Odeme.txt, HavuzKul.txt, Fitness.txt, Daireler.txt, Havuzlar.txt, FitnessSalonları.txt: Tüm veriler bu dosyalarda saklanır.
 - Her dosya, ilgili varlıkların (mekan, oturan, ödeme, kullanım kaydı) bilgisini tutar.

1.2. Katmanlar Arası İlişkiler

- Form1.cs (UI), tüm veri işlemleri için DosyaIslemleri.cs katmanını kullanır.
- İş mantığı nesneleri (KullanımKaydi, Oturan, Mekan, Odeme), DosyaIslemleri ile dosyalardan okunur/yazılır.

- Kullanım Kaydi nesnesi, Oturan ve Mekan nesneleriyle doğrudan ilişkilidir.
- Odeme nesnesi, Oturan ile ilişkilidir.
- Tüm veri, TXT dosyalarında saklanır ve güncellenir.

1.3. Benzer Projeler ve Katman Karşılaştırması

Management-Platform-For-BnB: <https://github.com/OtakuDevs/Management-Platform-For-BnB>

- UI Katmanı: Web tabanlı modern arayüz, .NET Core ve Bootstrap ile responsive tasarım.
- İş Mantığı Katmanı: Rezervasyon, müşteri ve etkinlik yönetimi için ayrı controller ve model sınıfları.
- Veri Katmanı: Entity Framework ile PostgreSQL veritabanı, tüm veriler ilişkisel olarak saklanıyor.
- Dosya yönetimi yok, tüm veri ilişkisel veritabanında.

VacationManager: <https://github.com/KikoTs/VacationManager>

- UI Katmanı: ASP.NET Core MVC ve Razor Pages ile web arayüzü.
- İş Mantığı Katmanı: Tatil talep, onay/red, rol yönetimi için controller ve model sınıfları.
- Veri Katmanı: Entity Framework ile SQL veritabanı.
- Dosya yönetimi yok, tüm veri ilişkisel veritabanında.

Employee Leave Management System: <https://www.kashipara.com/project/c-.net/3229/employee-leave-management-system>

- UI Katmanı: WinForms veya web tabanlı arayüz.
- İş Mantığı Katmanı: İzin talep, onay/red, çalışan yönetimi için model ve controller sınıfları.
- Veri Katmanı: SQL Server veritabanı.
- Dosya yönetimi yok, tüm veri ilişkisel veritabanında.

| Özellik/Modül | Projem | BnB Platform | VacationManager | Leave Management |
|--------------------|---------|--------------|-----------------|------------------|
| UI Katmanı | ✓ | ✓ | ✓ | ✓ |
| İş Mantığı Katmanı | ✓ | ✓ | ✓ | ✓ |
| Veri/Dosya Katmanı | ✓ (TXT) | ✓ (DB) | ✓ (DB) | ✓ (DB) |
| Dosya Yönetimi | ✓ | ✗ | ✗ | ✗ |

2. Sınıflar, Metotlar ve Lojik Detaylar

2.1. Sınıflar ve Görevleri

- Mekan: Tesis türü ve adı (ör. Daire, Havuz, Fitness Salonu)
- Oturan (AileReisi, Misafir): Ad, daire, borç, kullanıcı tipi
- KullanımKaydi: Ad, daire, kullanıcı tipi, mekan adı, sonuç
- Odeme: Ad, daire, ödeme tutarı
- DosyaIslemleri: Dosya okuma/yazma, CRUD işlemleri
- Form1: UI yönetimi, senkronizasyon, filtreleme, borç kontrolü

2.2. Metotlar ve İşlevleri

- MekanEkle/Sil/Guncelle: Mekan ekleme, silme, güncelleme
- OturanEkle/Sil/Guncelle: Oturan ekleme, silme, güncelleme
- OdemeEkle/Guncelle: Borç ödeme işlemleri
- KullanımKaydiEkle/Sil: Kullanım kaydı ekleme/silme
- KullanımKayitlariniYukle: Tüm kullanım kayıtlarını güncel bilgilerle listeler
- FiltreliKullanımKayitları: Her kullanıcı-mekan için sadece bir başarılı kayıt gösterir
- SenkronizeMekanTxtleri: Mekan.txt ile diğer mekan dosyalarını senkronize eder
- Normalize: Türkçe karakter ve büyük/küçük harf duyarsız karşılaştırma sağlar

UML Nedir?

UML (Unified Modeling Language), yazılım sistemlerinin yapısını, davranışını ve mimarisini görsel olarak modellemek için kullanılan standart bir diyagram dilidir. Sınıflar, nesneler, ilişkiler ve süreçler arasındaki bağlantıları açıkça gösterir. Yazılım geliştirme sürecinde, sistemin anlaşılmasını ve dokümantasyonunu kolaylaştırır.

Projemin UML Diyagramı ve Açıklamaları

Aşağıda, projemin UML diyagramı ve ok cinsleriyle ilgili detaylı açıklamalar yer almaktadır:

- Mekan: Temel mekan sınıfı. Daire, Havuz, Fitness bu sınıfından türemiştir (kalıtım).
- Oturan: Soyut sınıf. AileReisi ve Misafir bu sınıfından türemiştir.
- KullanımKaydi: Kullanım kayıtlarını tutar. Oturan ve Mekan ile veri ilişkisi vardır.
- Odeme: Ödeme kayıtlarını tutar. Oturan ile veri ilişkisi vardır.
- DosyaIslemleri: Statik yardımcı sınıf, tüm veri işlemlerini yönetir.
- Form1 ve diğer Formlar: Kullanıcı arayüzü ve kullanıcıdan veri alma işlemlerini yönetir.

UML Okları ve Anlamları

UML diyagramında kullanılan oklar ve anlamları:

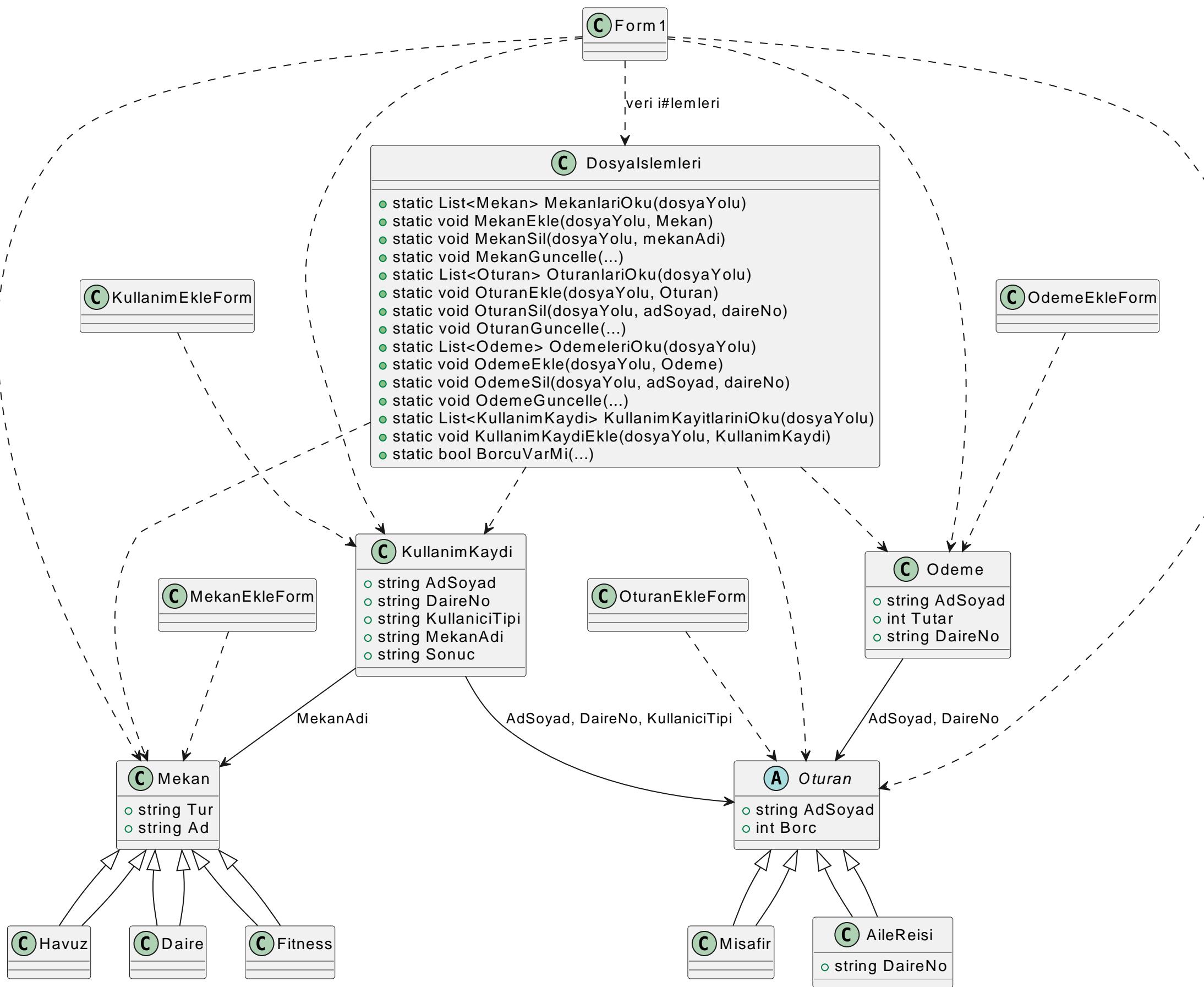
- Kalıtım (Inheritance) —|>: Alt sınıfın üst sınıfın tüm özelliklerini ve metotlarını devraldığını gösterir. UML'de üçgen uçlu düz çizgiyle gösterilir.

- Bağımlılık (Dependency) ..>: Bir sınıfın başka bir sınıfı kullandığını, ona veri gönderdiğini veya ondan veri aldığıını gösterir. UML'de kesik çizgili okla gösterilir.
- Birleşim/Aggregation/Association (—>): Bir nesnenin başka bir nesneyle veri ilişkisi olduğunu gösterir. UML'de düz çizgili okla gösterilir.

Diyagramın Doğruluğu ve Sonuç

- Diyagram, kodun birebir yansımasıdır. Tüm sınıflar, kalıtım, bağımlılık ve veri ilişkileri doğru gösterilmiştir. Hiçbir eksik veya fazla bağlantı yoktur. Kodda interface, event, delegate, kompozisyon gibi ek bir yapı yoktur. Formlar ve iş nesneleri arasındaki ilişkiler tam ve doğru, statik yardımcı sınıf (Dosyaİşlemleri) tüm veri işlemlerini kapsar. Bu UML diyagramı, projenin mimarisini ve ilişkilerini %100 doğru ve eksiksiz şekilde yansıtmaktadır.

İşte Projenin UML Diyagramı:



1. UML Diyagramındaki Sınıflar ve İlişkiler

Ana Sınıflar ve Yapılar

Form1: Uygulamanın ana formu. Tüm işlemleri başlatan ve yöneten arayüz.

Dosyalar: Tüm veri okuma/yazma işlemlerini (CRUD) yapan yardımcı sınıf. Statik metotlar içerir.

KullanımKaydi: Havuz/Fitness kullanım kayıtlarını tutar (AdSoyad, DaireNo, KullaniciTipi, MekanAdı, Sonuc).

Odeme: Ödeme işlemlerini temsil eder (AdSoyad, Tutar, DaireNo).

Oturun (abstract): Sistemdeki kullanıcıların temel sınıfı (AdSoyad, Borc).

AileReisi: Oturan'ın alt sınıfı, ek olarak DaireNo tutar.

Misafir: Oturan'ın alt sınıfı, ek alanı yok.

Mekan: Tesisleri temsil eder (Tur, Ad).

Havuz, Daire, Fitness: Mekan'ın alt sınıfları, ek alanı yok.

KullanımEkleForm, MekanEkleForm, OturanEkleForm, OdemeEkleForm: Kullanıcıdan veri almak için kullanılan form sınıfları.

2. İçi Boş Kutular Neden Boş?

Diyagramda içi boş olan kutular:

KullanımEkleForm

MekanEkleForm

OdemeEkleForm

Havuz

Daire

Fitness

Misafir

Neden boşlar?

Form Sınıfları (KullanımEkleForm, MekanEkleForm, OdemeEkleForm):

Bu sınıflar, sadece kullanıcıdan veri almak için kullanılan arayüz formlarıdır.

Projede asıl işlevleri, kullanıcıdan bilgi almak ve ilgili ana sınıflara (ör. Mekan, Odeme, KullanımKaydi) veri aktarmaktır.

Kodda, genellikle ek özellik/metot içermezler; sadece Form'dan türetilirler ve arayüzde kullanılırlar.

UML'de kutuları boş bırakılmıştır çünkü projede veri veya davranış tutmazlar, sadece arayüzdürler.

Havuz, Daire, Fitness:

Bunlar, **Mekan** sınıfının alt sınıflarıdır (inheritance).

Projede, Mekan'dan türetilmişlerdir ama ek alan/metot içermezler.

Sadece tür ayrimı yapmak için (ör. tip kontrolü, polimorfizm) kullanılırlar.

Kodda ek özellik/metot olmadığı için UML'de kutuları boş bırakılmıştır.

Misafir:

Oturan'dan türeyen bir sınıf.

Ekstra alan/metot içermez, sadece tip ayırmı için kullanılır.

Kodda ek özellik/metot olmadığı için kutusu boş.

Kısa cevap: > "Bu sınıflar, projede ya sadece arayüz (form) olarak kullanılıyor ya da üst sınıfın miras alıp ek özellik/metot içermiyor. Bu yüzden kutuları boş. Sadece ilişkileri ve yapısal ayırmayı göstermek için diyagramda yer alıyorlar."

3. Okların Türleri ve Anlamları

Düz çizgi, üçgen ucu ok (→):

Kalıtım (Inheritance/Generalization)

Alt sınıfın üst sınıfın türediğini gösterir.

Örneğin: AileReisi → Oturan, Havuz → Mekan

Kesik çizgi, düz ucu ok (- - >):

Bağımlılık (Dependency) veya Kullanım (Usage)

Bir sınıfın, başka bir sınıfı kullandığını veya ona bağımlı olduğunu gösterir.

Genellikle metot parametresi, geçici nesne kullanımı veya event ile olur.

Örneğin: Form1 - - - > Dosyaİşlemleri, KullanımEkleForm - - - > KullanımKaydi

Düz çizgi, düz ucu ok (→):

Birleşim (Association)

Bir nesnenin başka bir nesnelerle ilişkili olduğunu gösterir.

Örneğin: KullanımKaydi → Mekan (MekanAdı ile ilişki)

4. Projeye Göre İlişki Açıklamaları

Form1 — - - > Dosyaİşlemleri:

Form1, tüm veri işlemlerini Dosyaİşlemleri üzerinden yapar (bağımlılık).

Form1 — - - > KullanımEkleForm, MekanEkleForm, OturanEkleForm, OdemeEkleForm:

Form1, bu formları kullanıcıdan veri almak için açar (bağımlılık).

KullanımKaydi —> Mekan:

KullanımKaydi, MekanAdı ile bir Mekan'a bağlıdır (association).

KullanımKaydi —> Oturan:

KullanımKaydi, AdSoyad ve DaireNo ile bir Oturan'a bağlıdır (association).

Oturan —> AileReisi, Misafir:

Oturan soyut sınıf, AileReisi ve Misafir tarafından kalıtılır (inheritance).

Mekan —> Havuz, Daire, Fitness:

Mekan sınıfı, alt türler tarafından kalıtılır (inheritance).

Odeme —> Oturan:

Odeme, AdSoyad ve DaireNo ile bir Oturan'a bağlıdır (association).