

Estate agent and Costumer



NAME SURNAME: BILAL KHANZAR

NO: 231041079

Emlak Yönetim Sistemi

- **Amaç:** Müşteriler ve emlakçılar arasında etkili bir iletişim ve işlem kolaylığı sağlamak.
- **Özellikler:**
 - Müşteri bilgileri alınarak uygun emlak sunuluyor.
 - Emlakçı, ödeme işlemleri ve emlak seçimleri konusunda yardımcı oluyor.

Proje Genel Yapısı

01

Başlık: Proje
Yapısı ve İşlev

02

Müşteri: İsim,
telefon numarası
ve istenilen emlak
türünü girer.

03

Emlakçı: Emlak
türlerini sunar ve
ödeme işlemini
yönetir.

04

Hata Yönetimi:
Kullanıcı hatalı
veri girişi
yaptığında hata
mesajı gösterilir.

05

Polimorfizm: Aynı
işlev (metot) farklı
nesnelerde farklı
şekillerde çalışır.

Temel Özellikler

Başlık: Temel Özellikler ve Fonksiyonlar

Müşteri Bilgileri:

- İsim ve telefon numarası alınır.
- İstenilen emlak türü girilir.

Emlakçı Bilgileri:

- İsim ve telefon numarası alınır.
- Emlak türlerini sunar.
- Ödeme işlemi kontrol edilir.



Polimorfizm (Polymorphism)

- Polimorfizm Nedir?
- **Tanım:** Aynı metot adı, farklı nesnelerde farklı işlevler gösterir. Çağırılması normal method çağılmaktan farklı A a = new B mesela
- **Örnek:**
 - **BilgileriGoster** metodu hem **Musteri** hem de **Emlakci** sınıflarında kullanılır, fakat her sınıf için farklı bilgileri ekrana yazdırır.
- Musteri muster1 = BilgiAlMusteri(); // Polimorfizm (Polymorphism): oluyor
- Emlakci emlakci1 = BilgiAlEmlakci();

TryParse Kullanımı



TryParse ile Hata Kontrolü



Tryparse Nedir ? bir değerin belirli bir türde (örneğin, int, double, DateTime gibi) geçerli olup olmadığını kontrol etmek için kullanılan bir metoddur



Amaç: Kullanıcıdan alınan verilerin geçerliliğini kontrol etmek.



Örnek: Telefon numarasının sayısal olup olmadığını kontrol etmek için TryParse kullanılır.

Eğer geçerli bir telefon numarası girilmezse, hata mesajı gösterilir.



Example :

```
if (!long.TryParse(Console.ReadLine(), out musteriTelefon))
{
    throw new Exception("Telefon numarası sadece sayısal olmalıdır!");
}

Console.Write("İstenen Emlak Türü (örn. Daire): ");
string musteriTurlakTuru = Console.ReadLine();

return new Musteri(musteriIsim, musteriTurlakon, musteriTurlakTuru);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

Kullanıcı Akışı

- **Müşteri Bilgileri Alınır:** Müşteri ismi, telefon numarası ve emlak türü girer.
- **Emlakçı Bilgileri Alınır:** Emlakçı ismi ve telefon numarası girilir.
- **Emlak Türü Seçimi:** Emlakçı, 3 farklı emlak türü sunar (Daire, Villa, Apartman). Müşteri birini seçer.
- **Ödeme İşlemi:** Müşteri, ödeme yapar. Emlakçı ödeme tutarını kontrol eder ve işlemi onaylar.

Hata Yönetimi

1

Try-Catch Blokları:
Kullanıcı yanlış bir veri girdiğinde hata mesajı gösterilir.

2

Ödeme Hatası: Ödeme tutarı yetersizse, hata mesajı gösterilir.

3

Veri Hataları: İsim veya telefon numarası gibi veriler geçerli değilse, kullanıcı bilgilendirilir.

Sonuçlar ve İleriye Dönük Gelişmeler

Projenin Sonuçları ve Gelecek Planları

Sonuç: Müşteriler ve emlakçılar arasında daha hızlı ve güvenli işlemler yapabiliriz .

İleriye Dönük Planlar: belki gelecekte

Veritabanı Entegrasyonu: Müşteri ve emlakçı bilgilerini bir veritabanına kaydedebiliriz .

Web/Mobil Uygulama: Sistemin daha geniş bir kitleye ulaşabilmesi için web ya da mobil uygulama geliştirme.



BU kadar yapabildim