A red text on a white background

Description automatically generated

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BYM304 YAZILIM MİMARİSİ VE TASARIMI ÖDEV-2

HAZIRLAYAN

MERT TOSUN

210601027

BİLGİSAYAR MÜHENDİSLİĞİ

TESLİM TARİHİ

02.06.2024

DERSİN EĞİTMENİ

Assist. Prof. Dr. Kemal Çağrı Serdaroğlu

PROJENİN AMACI: Belirli istatistiksel işlemleri yapabilen bir yazılım geliştirmektir. Bu yazılım, bir klasörde bulunan sıcaklık ve hava bağıl nem ölçümlerini işleyerek kullanıcıya çeşitli istatistiksel veriler sunmayı hedeflemektedir. Ancak bu bize verilen örneğin amaçlarıdır burada asıl amaçlanan bizim bu projeyi geliştirirken kullandığımız desenler ve SOLID ilkelerini uygulayıp uygulamamamız olacaktır.

KULLANDIĞIM İDE VE KODLARIM

Projeyi kodlarken Visual Studio idesinden faydalandım. Burada bir Windows forms uygulaması ile arayüz tasarımını yaptım. Arayüzü tasarlarken checkbox,textbox, button ve labellar kullandım.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada arayüzümüzü görüyoruz. Arayüzü tasarlarken ödevde istenenlere göre birebir tasarladım. Üstte bir buton ve bu buton sayesinde klasör seçme işlemini gerçekleştiriyoruz. Butonun sağında bir adet textbox var burada seçtiğimiz klasörün yolu gözüküyor. Ortalama bul,maxbul, minbul,st sapma bul , frekansı bul ve medyan bul checkboxları ile dosyamızdaki değerlere hangi işlemi uygulayacağımızı işaretleyip altta bulunan hesapla butonuna bastığımızda alttaki mesaj labelinin yanındaki textboxa gerekli hesaplamalardan geçirilen dosya içindeki değerlerin sonucu veya hesaplama yapılamadıysa hesaplama yapılamadı uyarısı geliyor. Global seçilirse işaretlenen fiziksel ölçüme ait tüm değerler hesaplanacak. Ayrıca işlemleri yaptıktan sonra sonuçları bir dosyaya yazıyoruz ve bu dosyanın yolu da textboxta XXX yerine konulmuştur olarak gözüküyor. Şimdi detaylı olarak kodumuzun açıklamasını, kullandığımız desenleri, solid ilkelerini ve use case diyagramlarını gösterelim.

metin, yazı tipi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kullandığım kütüphaneler yukarıda görüldüğü gibidir bunlar genelde c# programlama dilinde kullanılan temel kütüphanelerdir forms arayüz tasarımı için, IO dosya işlemleri ve girdi çıktı işlemleri vs. için kullanılır. İlk olarak hesaplamaları yaptığım metotları tutan classtaki metotları anlatacağım önce metotlarda kullandığım linq kavramından bahsetmem gerekirse:

**LINQ NEDİR?**

LINQ, "Language Integrated Query" kelimelerinin kısaltmasıdır ve Microsoft tarafından .NET Framework'e entegre edilmiş bir sorgu dilidir. LINQ, veriler üzerinde yapılan sorguları, filtrelemeleri, eşleştirmeleri ve dönüşümleri yapmak için kullanılır. Sorgular doğrudan C# veya diğer .NET dilleri içine entegre edilmiştir. Bu, veri üzerinde sorgular yaparken, dilin kendisine aşina olduğunuz için daha kolay ve doğal bir şekilde kod yazmanıza olanak sağlıyor.

Kullandığım metotlar:

**ortalamabul Metodu:**

Bu metod, bir double türünden oluşan bir liste alır ve listenin ortalamasını hesaplar. İlk olarak, gelen liste kontrol edilir ve eğer null ise veya içinde hiç eleman yoksa 0 döndürülür. Ardından, LINQ kullanılarak, listenin içindeki değerlerin ortalaması alınır ve bu değer döndürülür. Eğer listede elemanlar varsa, bu elemanların ortalaması hesaplanır ve geri döndürülür. Eğer liste boş veya null ise, 0 değeri döndürülür.

**maximumBul Metodu:**

Bu metod, bir double türünden oluşan bir liste alır ve listenin en büyük değerini bulur. Eğer gelen liste null ise veya içinde hiç eleman yoksa, double türünün NaN (Not a Number) değeri döndürülür. Değerler içinde en büyük değer LINQ kullanılarak bulunur ve döndürülür.

**minimumBul Metodu:**

Bu metod, bir double türünden oluşan bir liste alır ve listenin en küçük değerini bulur. Eğer gelen liste null ise veya içinde hiç eleman yoksa, double türünün NaN değeri döndürülür. Değerler içinde en küçük değer LINQ kullanılarak bulunur ve döndürülür.

**standartSapmaBul Metodu:**

Bu metod, bir double türünden oluşan bir liste alır ve listenin standart sapmasını hesaplar. Eğer gelen liste null ise veya içinde hiç eleman yoksa, double türünün NaN (Not a Number) değeri döndürülür. Listenin standart sapması, öncelikle ortalama hesaplanarak ve ardından her elemanın ortalama ile farkının karesinin toplamı hesaplanarak elde edilir. Son olarak, toplamın eleman sayısına bölünüp karekökü alınarak standart sapma hesaplanır ve döndürülür.

**frekansBul Metodu:**

Bu metod, bir double türünden oluşan bir liste alır ve listenin içindeki değerlerin frekanslarını hesaplar. Eğer gelen liste null ise veya içinde hiç eleman yoksa, boş bir Dictionary döndürülür. Değerler gruplanarak her bir değerin frekansı bulunur ve bir Dictionary içinde saklanır. Bu Dictionary, değerlerin frekanslarını anahtar-değer çiftleri olarak tutar.

**medianBul Metodu:**

Bu metod, bir double türünden oluşan bir liste alır ve listenin medyanını hesaplar. Eğer gelen liste null ise veya içinde hiç eleman yoksa, double türünün NaN (Not a Number) değeri döndürülür. Değerler küçükten büyüğe sıralanır ve listenin eleman sayısına göre ortadaki değer veya ortadaki iki değerin ortalaması medyan olarak döndürülür.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kodumuzu incelemeye devam ettiğimizde klasör seçme ve hesaplama yaparken kullandığımız butonların kodlarını görüyoruz.

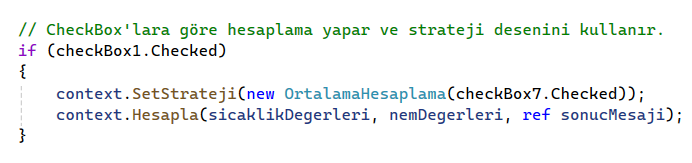


Dosyalardan aldığımız değerleri hesaplama işlemlerinde kullanmak için birer double liste dönüştürüyoruz.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yukarıdaki kod parçasında klasörlerimiz içindeki tüm ".txt" dosyalarını okuyarak içerdikleri sıcaklık ve nem verilerini işler. İşlenen her dosyanın içeriği satır satır okunur ve her bir satır virgülle ayrılmış parçalara ayrılır. Bu parçalardan birincisi zamanı, ikincisi ise değeri temsil eder. Bu değerler daha sonra ilgili ölçüm türüne (sıcaklık veya nem) göre ayrı listelere eklenir.



Üstte gördüğümüz kısımda checkbox’ın click eventini görüyoruz burada checkboxlar ile işaretleme yaptığımız kutucuğun işlemi neyse ona göre işlem yapıyoruz. Burada dikkat çekmemiz gereken şey ise programlama esnasında kullandığımız yazılım deseni strateji yazılım desenidir.

**STRATEJİ YAZILIM DESENİ**

Strateji tasarım deseni, bir algoritma ailesini tanımlar her birini kapsüller ve bu algoritmaları birbirinin yerine kullanabilmenizi sağlar. Strateji deseni, algoritmaların kullanıcılardan bağımsız olarak değiştirilebilmesini sağlar. Bu desen, genellikle bir dizi ilgili algoritmanın bir sınıf ailesi olarak tanımlandığı ve bu algoritmalar arasında seçim yapılmasına olanak tanıyan davranışsal bir desen olarak kategorize edilir. Bu uygulamamızda farklı hesaplamalar yapacağımız için bu deseni kullandık.

Strateji Deseninin Avantajları

* Esneklik: Algoritmalar bağımsız sınıflarda tanımlandığı için, yeni bir algoritma eklemek veya mevcut bir algoritmayı değiştirmek kolaydır.
* Bakım Kolaylığı: Her algoritma kendi sınıfında tanımlandığı için, kodun okunabilirliği ve bakımı daha kolaydır.
* Yeniden Kullanılabilirlik: Aynı algoritma farklı bağlamlarda tekrar kullanılabilir.

Strateji Deseninin Bileşenleri

* Strateji Arayüzü (Strategy Interface): Algoritmaların uyması gereken ortak arayüzdür. Bu arayüz, strateji ailesindeki tüm algoritmaların uygulaması gereken bir veya daha fazla metodu tanımlar.
* Somut Stratejiler (Concrete Strategies): Strateji arayüzünü uygulayan ve belirli algoritmaları gerçekleştiren sınıflardır.
* Bağlam (Context): Strateji arayüzüne bir referans içerir ve bu referans aracılığıyla algoritmaları yürütür.

Strateji Arayüzü (IHesaplamaStratejisi)

metin, çizgi, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada kodumuzdaki strateji arayüzünü görmekteyiz hesapla fonksiyonundaki 7 başvuru işlemlerimizin tümünün buradaki arayüzden faydalandığını gösteriyor. Bunlarda yani max, min, frekans vs. bulma metotlarımız ise somut Stratejiler (Concrete Strategies) olarak geçer. Aşağıda onunda bir örneğini koyacağım.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bağlam (Context)

Context sınıfı hangi stratejinin kullanılacağını belirler ve bu stratejiyi çalıştırır.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Context sınıfı, strateji arayüzüne bir referans içerir ve SetStrateji metodu aracılığıyla hangi stratejinin kullanılacağını belirler. Hesapla metodu, belirtilen stratejiye göre hesaplamayı gerçekleştirir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kullanıcı hangi hesaplamaların yapılacağını belirlemek için çeşitli checkbox'ları işaretler. Bu işaretlemelere göre, uygun strateji sınıfları oluşturulur ve context sınıfına atanır. Ardından context sınıfı, belirtilen stratejiye göre hesaplamayı gerçekleştirir ve sonucu ekranda göstermemize olanak sağlar.

**KODUMUZDA UYGULADIĞIMIZ SOLİD İLKELERİ**

Tek Sorumluluk Prensibi (Single Responsibility Principle):

* Hesaplamalar sınıfı, yalnızca matematiksel hesaplamaları gerçekleştirmek için kullanılır.OrtalamaHesaplama,MaximumHesaplama,MinimumHesaplama,StandartSapmaHesaplama, FrekansHesaplama ve MedianHesaplama sınıfları, belirli bir hesaplama türünü gerçekleştirmek için kullanılır.
* Context sınıfı, stratejiyi seçmek ve uygulamak için kullanılır.

Açık/Kapalı Prensibi (Open/Closed Principle):

Açık/Kapalı Prensibi, bir sınıfın veya metodun genişletilmeye açık, ancak değiştirmeye kapalı olması gerektiğini belirtir. Bu kodda, yeni bir hesaplama stratejisi eklemek oldukça kolaydır

* Yeni bir hesaplama stratejisi eklemek için IHesaplamaStratejisi arayüzünü uygulayan yeni bir sınıf oluşturmanız yeterlidir. Mevcut kodda herhangi bir değişiklik yapmanıza gerek yoktur.

Liskov Yerine Geçme Prensibi (Liskov Substitution Principle)

Liskov Yerine Geçme Prensibi, bir sınıfın alt sınıflarının, üst sınıfın yerine kullanılabilmesi gerektiğini belirtir. Bu kodda, her bir hesaplama stratejisi (OrtalamaHesaplama, MaximumHesaplama, vb.) IHesaplamaStratejisi arayüzünü uygular.

* IHesaplamaStratejisi arayüzünü uygulayan herhangi bir sınıf, strateji olarak kullanılabilir ve Context sınıfında sorunsuz bir şekilde çalışabilir.

Arayüz Ayrımı Prensibi (Interface Segregation Principle):

Arayüz Ayrımı Prensibi, müşterilerin (client) kullanmadıkları arayüzlere bağlı kalmaması gerektiğini belirtir. Kodumuzda:

* IHesaplamaStratejisi arayüzü, yalnızca gerekli olan Hesapla metodunu içerir ve her strateji bu metodun farklı bir uygulamasını sağlar.

Bu prensip, IHesaplamaStratejisi arayüzü ile uygulanmıştır, çünkü arayüz sadece gerekli olan metodları tanımlar ve bu sayede istemciler gereksiz metodlara bağlı kalmazlar.

Bağımlılıkların Ters Çevrilmesi Prensibi (Dependency Inversion Principle):

Bağımlılıkların Ters Çevrilmesi Prensibi, yüksek seviyeli modüllerin düşük seviyeli modüllere değil, her ikisinin de soyutlamalara bağımlı olması gerektiğini belirtir.

* Context sınıfı, doğrudan strateji sınıflarına bağımlı değildir, bunun yerine IHesaplamaStratejisi arayüzüne bağımlıdır. Bu, yüksek seviyeli modül olan Context sınıfının, düşük seviyeli modüller olan hesaplama stratejilerine doğrudan bağımlı olmamasını sağlar.



İşlemleri yaptıktan sonra gerekli sonuçları bir txt dosyasına kaydediyoruz ve mesaj textboxımıza klasörün olduğu dosya yolunu yazıyoruz

**CLASS DİYAGRAMI**

metin, ekran görüntüsü, yazı tipi, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

Class diyagramımız yukarıda görüldüğü gibidir. Bu projede yazılım geliştirme desenlerini kullanarak bir problemi çözmüş olduk.

NOT: Hocam kodum ilk başta raporda gösterdiğim gibi çalışıyordu daha sonra geliştirme yaparken visual stuido kaynaklı bir problem yaşadım ve değişiklik yapmak zorunda kaldım. Dosya okuma ve yazma yaparken sorun yaşadığım için dıştaki klasörden hesaplama yapmak zorunda kaldım. İçteki nem ve sıcaklık klasörlerini seçtiğimde hata mesajı aldım uğraşmama rağmen ideden kaynaklı olduğunu düşündüğüm hatayı çözemedim. Classları kodun içine alma sebebim bundan ötürü.