**2024 YAZILIM GELİŞTİRİCİ YETİŞRİME KAPMPI İÇİNDEKİLER KISMI**

* **HAFTA-1 = İlk Canlı yayın yapıldı ve 2 kez izlendi.**
* **İlk Ödev Yapıldı (Kodlama.io üzerinden istenilen yapılar bulundu**
* **5 Saatlik C# Programlama Kampı izlendi**
* **HAFTA-2 = İlk Canlı yayın yapıldı ve 2 kez izlendi.**
* **İlk Ödev Yapıldı (Kodlama.io üzerinden istenilen yapılar bulundu**
* **5 Saatlik C# Programlama Kampı izlendi**
* **HAFTA-3 = İlk Canlı yayın yapıldı ve 2 kez izlendi.**
* **İlk Ödev Yapıldı (Kodlama.io üzerinden istenilen yapılar bulundu**
* **5 Saatlik C# Programlama Kampı izlendi**
* **HAFTA-4 = İlk Canlı yayın yapıldı ve 2 kez izlendi.**
* **İlk Ödev Yapıldı (Kodlama.io üzerinden istenilen yapılar bulundu**
* **5 Saatlik C# Programlama Kampı izlendi**
* **HAFTA-5 = İlk Canlı yayın yapıldı ve 2 kez izlendi.**
* **İlk Ödev Yapıldı (Kodlama.io üzerinden istenilen yapılar bulundu**
* **5 Saatlik C# Programlama Kampı izlendi**
* **Dsf**

**HAFTA-1 =**

**1)Classlarda property = Şimdi bir class içinde bir özellik tanımlarsan eğer bu default olarak “private” yani yanlızca class içinde kullanılacak dışarıdan erişilemeyecek şekilde tasarlanır. Siz bunu dışarıya erişilebilir kılmak için public olarak tanımlananız gerekir**

**2)SOLID olayı =Şimdi bir proje eğer bir değişikliğe direnç gösteriyorsa bilki SOLID e uymadan yapılmıştır yada yeterince uyamamıştır.**

1. **S (Single Responsibility Principle - Tek Sorumluluk Prensibi):**
   * **Bir sınıfın veya modülün sadece bir tek sorumluluğu olmalıdır.**
   * **Bu prensip, her bir sınıfın sadece tek bir işi yapması gerektiğini belirtir. Bir sınıfın birden fazla sorumluluğu olduğunda, kod karmaşık hale gelir, bakımı zorlaşır ve hatalara yol açabilir.**
2. **O (Open/Closed Principle - Açık/Kapalı Prensibi):**
   * **Yazılım varlıkları (sınıflar, modüller, fonksiyonlar) genişlemeye açık ancak değişikliğe kapalı olmalıdır.**
   * **Kod, yeni işlevsellik eklemek için açık olmalı, ancak mevcut işlevselliği değiştirmek için kapalı olmalıdır. Bunu genellikle soyutlamayla ve kalıtım/uyumluluk gibi tasarım kalıplarıyla sağlarız.**
3. **L (Liskov Substitution Principle - Liskov'un Yerine Koyma Prensibi):**
   * **Bir üst tür (super-type) nesnesinin her zaman alt tür (sub-type) nesnesiyle yer değiştirilebilir olması gerekmektedir.**
   * **Yani, bir alt sınıf, üst sınıfın tüm davranışlarını sergileyebilmelidir. Bu prensip, kalıtım hiyerarşisinde tutarlılık sağlamayı ve beklenmedik davranışlardan kaçınmayı amaçlar.**
4. **I (Interface Segregation Principle - Arayüz Ayırma Prensibi):**
   * **Kullanıcılar için gereksiz ayrıntıları içermeyen, spesifik arayüzler tanımlanmalıdır.**
   * **Yani, istemciler sadece ihtiyaç duydukları yöntemlere sahip olan arayüzleri kullanmalıdır. Bu prensip, gereksiz bağımlılıkları azaltır ve sınıflar arasındaki bağlantıyı gevşetir.**
5. **D (Dependency Inversion Principle - Bağımlılıkları Tersine Çevirme Prensibi):**
   * **Yüksek seviyeli modüller, düşük seviyeli modüllere bağımlı olmamalıdır. Her ikisi de soyutlamalara bağlı olmalıdır.**
   * **Soyutlamalar (interface, abstract class vb.) üzerinde bağımlılıklar oluşturarak, modüller arasındaki bağımlılıkları azaltırız. Bu sayede kodun değiştirilmesi ve yeniden kullanılabilirliği kolaylaşır.**

**NOT: string aslına bir char Array dir. AKLINDA BULUSUN!!!**

**3)Genel İsimlendirmeler ve İşlemleri =**

**Abstracks = Soyut , Concrete= Somut**

**Entities => Varlık Nesnelerini Tutar**

**Business => Kuralları Tutar**

**Bu Katmanları işledikçe ne işe yaradıklarını yazacam**

**NOT: Classlar özellik tutabildiği gibi metotları da barındırabilirler FAKAT kilit nokta projenin SOLİD e uyması için bir classta “özellik varsa metot yok metot varsa özellik” kuralına uymamız gerekir**

**ÖNEMLİ NOT: Bir iş classıı yani business class diğer bir iş classını kullanırken new’leyerek kullanıyorsa bu program yeniliğe karşı bir direnç gösterir.**

**NOT: Bu dediğimi öyle çok net bir ifade olarak algılama ama bir projede ne kadar çok if varsa o projeye geliştirmeye bir o kadar uzaktır.**

**NOT: Bir Projeye yeni bir özellik geldiği zaman bu seni eski bir kodu değiştirmeye itiyorsa eğer geçmiş olsun demektir bunu kod üzerindeki bir güncelleme ile karıştırma bu dediğim mesela 50 yaşında emekli için ikramiye hesaplayan bir sistemin bunu 60 yaşına çekmesi değil. Atıyorum sen SGK ile hesaplıyordun artık Bağkuruda hesaplaman lazım senin bağkur kısmını projeye ekliyor olman seni sgkda bir özelliğe yönlendiriyorsa bil ki bir sorun var**

**3)Intarface ile class Bağımlılığından kurtulma**

**(LÜTFEN Mor kısmı dikkatlice oku )**

**Şimdi yukarıda “önemli not” denilen kısımda anlattığım ola burda kafana daha iyi oturacak.**

**Ben bunu Örnek üzerinden anlatacağım şimdi diyelim Maske Dağıtım uygulaması yapıyorsun ve business katmanında da “PttManager” adlı bir katmanın kullanıcıya ait kontrolleri “PersonManager” adlı bir class dan kontroller yaparak ve kendisi de kontroller uygulayarak maske verecek. Buna Örnek Sallıyorum “PersonManager da türkiye vatandaşımı ona bakacak değilse vermeyecek ha eğer türkiyeliyse ve atıyorum Ptt kartı varsa 5 maske fazla verecek”(Lütfen örneğe takılma sadece şuna odaklan “PttManager” adlı business classı “PersonManager” adlı business classını bir kontrol için kullanacak.**

**Eğer bu kullanma işini o classı newleyerek yapmaya kalkarsa ona bağımlı hale gelecek şimdi bu dediğimi 1 metot olarak düşünme düşünki 100 farklı işlemde PttManager PersonManagerda bulunan kişinin türkiyeli olup olmadığını sorgulayan bir metodu kullanıyor ve türkiye vatandaşı olmayana maske vermiyor**

**DEVLETTE BİR ANDA DİYORKİ**

**“sadece Türkiyeliler değil ülkedeki herkes maske alabilir”…**

**Bu ne demek biliyor musun sen ya yeni bir ForeignerManager(Yabancı Manager) adlı bir class daha tanımlayıp 100 yere gidip işte sallıyorum “|| operatörüyle yeni classı işe dahil etmeye çalışacaksın(ki bence bunu yapmaya üşeneceksin:☹ )”**

**Ya da gidip PersonManger a if lerle işte yabancıysa şunu yap türkse tcsini sor türk ve gençse şunu yap gibi gibi iflerle PersonManager ı Sebzeli Çorbaya çevirecen.**

**İşte Bu işin Süpermeni Interface**

**Interface ile PersonManager gibi kontroller yapan diğer classları Interface bölümüne mirasçı kılarsan senin classları kullanman için newlemene gerek kalmaz bir string değeri tanımlayıp kullanır gibi interface değerini tanımlar kullanırsın ve diyelim yeni sisteme geçilecek bu seni zerre zorlamaz hemen classını açarsın ve Interface deki imzaları çeker gerekli kuralları yazarsın.**

**4) Başka Bir Micro Service i projene dahil etme**

**Şimdi burda Başka bir microservice i projene dahil edip ona belirli işler yaptırabiliyoruz mesela biz dersteki projede tckimlik kontrolü ile vatandaşa maske dağıttık bunun için “Business” üzerine gelip ->add -> Connected Service -> WCF seçip sitenin linkini vererek sisteme ekliyoruz.**