Soru-1: Infoset olarak backend tarafında OOP, frontend tarafında ise fonksiyonel programlama paradigmalarını kullanıyoruz. Bunları birbirinden ayıran şeyler nelerdir? Birbirlerine göre avantajlarını-dezavantajlarını basit birkaç kod bloğuyla açıklayabilir misin?

-----------------------------------------------------------------------------------------------------------

Fonksiyonel programlamada değişkenler ve fonksiyonlar, programlamanın temel öğeleridir, OOP'de ise temel öğeler metotlar, nesne ve class kavramlarıdır.

Fonksiyonel programlama ile yazılan kodların okunup anlaşılması daha basitken OOP yaklaşımı ile yazılan kodların okunması daha zorlayıcı olabilir.

OOP'deki temel öğelerin, nesnelerin durumu (state), özellikleri vardır. Nesnenin durumu, özelliği bir etkileşim sonucunda değişebilir. Fonksiyonel programlamanın temel öğelerinde, fonksiyonlarda bir durum, özellik söz konusu değildir.

Fonksiyonel programlamada değişmez (immutable) veriler kullanılır, OOP'de ise değişmez(immutable) ve değişken(mutable) veriler birlikte kullanılır.

Fonksiyonel programlama karmaşık işlemler tasarlarken , büyük verilerle çalışırken OOP'e göre daha iyi performans sağlar.

OOP, fonksiyonel programlamaya göre daha fazla bellek kullanır.

Karmaşık kullanılan sınıflarda bir hata meydana geldiğinde debug etmesi zordur. Fonksiyonel programlamada debug daha kolaydır.

OOP'de veriler, nesneler içerisinde erişim belirleyiciler (access modifiers) aracılığıyla programın diğer ögelerinden saklanabilir, kullanımı kısıtlanabilir. (Encapsulation) OOP'de veriye erişim kısıtlanabilirken fonksiyonel programlamada ise bu şekilde bir veri erişimi kısıtlaması durumu yoktur, fonksiyonlar aynı verileri paylaşabilir ve kullanabilir.

Fonksiyonel programlamada modifikasyon işlemi daha zordur; bir fonksiyonda yapılan değişiklik, diğer fonksiyonları ve programın çalışma mantığını etkileyebilir. OOP'de ise nesneler gevşek bağlılık (loose coupling) prensibi ile oluşturulur ve bir nesnede yapılan değişiklik diğer nesnelerin işlevini ve programın çalışma mantığını etkilemez.

İki yaklaşımın da birbirlerine karşı avantajlı ve dezavantajlı olduğu durumlar var. Hangi yaklaşımın kullanılacağı yapmak istediğimiz işin gereksinimlerine ve uygunluğuna göre tercih edilmelidir.

-----------------------------------------------------------------------------------------------------------

Önce OOP yaklaşımıyla geometrik şekiller ile işlemler yapan bir uygulama geliştirdiğimizi düşünelim. Ardından fonksiyonel yaklaşım ile bir örnek yapalım.

Örnek kod (OOP) :

OOP özelliklerinden inheritance kullanarak abstract bir Shape class oluşturup Circle ve Rectangle şekillerimizi Shape sınıfımızdan türettik. Shape classımıza alan hesabı yapan bir Area() metodu ekledik ve türettiğimiz classlarda bu metodu override ederek propertylerimizi uygun formüllerin içerisine ekledik.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Artık bu sınıflardan instance alarak şekillerimiz için alan hesabı yapabiliriz.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

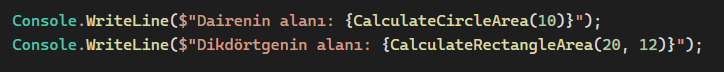
Aynı işlemi fonksiyonel programlama ile yapalım.

Daire ve dikdörtgen şekilleri için uygun parametreleri ve formülleri barındıran iki tane fonksiyon oluşturduk.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Artık bu fonksiyonları kullanarak şekillerimiz için alan hesabı yapabiliriz.



-----------------------------------------------------------------------------------------------------------

Dediğim gibi hangi yaklaşımı kullanacağımızı ihtiyaçlarımıza göre seçmeliyiz. İkisinin de birbirine karşı üstün olduğu durumlar var. Bu örneği ele alırsak, uygulamada sadece alan hesabı veya başka hesaplar yapmak istiyorsak fonksiyonlarımızı kullanmak uygun olacaktır.

Uygulamada ‘birden fazla şekli bir arada kullanayım, bu dikdörtgen küçülsün şu dairenin içine girsin’ gibi, şekillerle farklı etkileşimlere girme, özelliklerini değiştirme gibi ihtiyaçlarımız olursa OOP yaklaşımı daha uygun olacaktır.