1. Java dünyasındaki framework'ler ve çözdükleri problemler nedir?(Spring MVC, JSP, Struts, Vaadin). Örnekler ile açıklayın.

Cevap:

Spring Framework: Java ile geliştirme yapmayı kolaylaştıran modülleri içeren bir framework'tür. Açık kaynak kodludur. Spring kullanıcılara OOP tabanlı bir tasarım sağlar.

Spring MVC: Spring Framework'u içerisinde web tabanlı projeler geliştirilmesini sağlar. MVC'nin açılımı Model View Controller'dır. Kodu farklı parçalara ayırarak geliştirme yapılır bu da projenin kolay yönetilmesini sağlar. Karmaşıklığı azaltır.

Model Katmanı -> Verilerin modellendiği kısımdır.

View Katmanı -> Kullanıcı ile etkileşimde bulunulan katmandır.

Controller -> Model ve View arasında kalan katmandır. Verilerin Model'den View'a aktarılmasını sağlar.

JSP: Açılımı Java Server ages'dır. Uzun HTML kodları proje yönetimini zorlaştırabilir bunun önüne geçmek için JSP kullanılabilir. JSP ile HTML kodlarının içerisine java komutları yerleştirilir. Böylece kod karmaşıklığı azaltılır.

Apache Struts: MVC mimarisini kullanan, web geliştirmede kullanılan bir framework'tür. Java server pages ile uyumlu çalışır.

Vaadin: Java web uygulamaları için UI sunan bir framewrok'tür. Projeye Javascript kütüphaneleri eklenmesine olanak sağlar. Açık kaynak kodlu bir framework'tür. Java masaüstü uygulaması gibi kodlama yapılarak web uygulaması geliştirilmesine olanak sağlar.

2. Katmanlı mimari nedir?

Cevap: Kod yazarken bazı durumlara dikkat edilmesi gerekir. Okunabilirlik, tekrar kullanılabilirlik ve anlaşılabilirlik önemli kavramlardır. Proje üzerinde yapılacak herhangi bir değişikliğin zaman ve iş yükü açısından az maliyetli olması programcı için iyi bir şeydir. Bu nedenle katmanlı mimariler geliştirilmiştir.

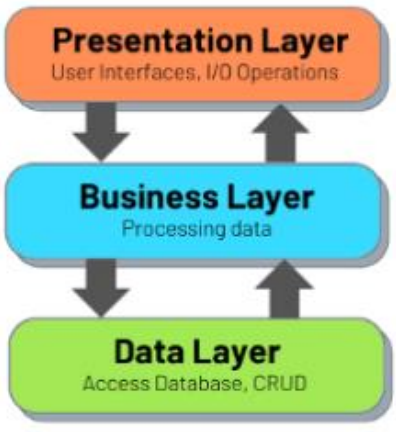
Katmanlı mimari, karmaşık ve büyük projelerde karmaşıklığı azaltır. Katmanlı mimari kullanılan bir projede belli standartlar vardır ve bu standartlara göre kodlama yapılır. Böylece kod okunabilirliği artar ve hata yönetimi kolaylaşır.

Katmanlı mimarinin 3 katmanı vardır:

Sunum Katmanı (Presentation Layer)

İş Katmanı (Business Layer)

Veri Katmanı (Data Access Layer)



Katmanlı Mimari

Veri Katmanı: Veri tabanı bağlantılarının yapıldığı katmandır. Veri erişimi bu katmanda yapılır. Veri ekleme, silme, güncelleme ve ekleme gibi temel veri tabanı işlemleri gerçekleştirilir.

İş Katmanı: Yapılacak iş ile ilgili kurallar burada tanımlanır. Veri katmanından alınan veriler iş katmanında işlenir. Ya da kullanıcıdan gelen veriler iş katmanında işlenerek veri katmanına iletilir. Böylece projedeki bağımlılıklar azaltılmış olur ve kodun okunabilirliği artar. Verilere erişime kimler tarafından açık olup olmayacağı da iş katmanında belirlenir.

Sunum Katmanı: Kullanıcının program ile etkileşime girdiği katman, sunum katmanıdır. MVC uygulaması, Windows form uygulaması, android mobil uygulaması, konsol uygulaması gibi uygulamaların arayüz katmanıdır. Sunum Katmanında kullanıcılara veriler gösterilebilir ya da kullanıcından veriler alınarak iş katmanına oradan da veri katmanına iletilebilir.

3. Garbage collector nedir, nasıl çalışır? Diğer C++ ile karşılaştırın.

Cevap:

Nedir? Garbage collector bir hafıza yönetim mekanizmasıdır. Düşük öncelikli bir iş parçasıdır (low priority thread). Geri planda Java platformunda sürekli olarak çalışır. Amacı belleği daha etkin kullanmak için işi biten nesnelerin kapladığı alanları temizlemektir.

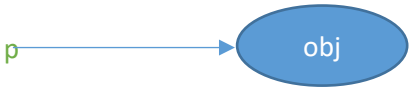
Nesne yönelimli programlama dillerinde bellek dinamik olarak kullanılır. Garbage collector, dinamik bellek temizliği yaptığı için belleğin heap kısmını temizler (Heap Stack ->). Java platformu sayesinde, nesneler kullanıldıktan sonra işlevi kalmadığında kapladıkları yerler belleğe iade edilir. Bu tür framework'lere "Manage Environment" denir. Bu mekanizma ile yazılım geliştirici kullandığı nesneyi kendisi iade etmek zorunda değildir. Java kullanılmayan nesneleri kendi tespit ediyor ve bellekten atıyor.

Nasıl Çalışır? Düzenli aralıklarla devreye girer (Genellikle sistemin çok yoğun olmadığı anlarda). Devreye girdiğinde o anda çalışan kaç tane java programı varsa o programların üretmiş olduğu ancak kullanılmayan nesneleri tespit eder. Tespit ettiği nesneleri kullanıcı için bellekten atar.

Bellekten atılacak nesneye garbage denir. Garbage collector bir nesnenin garbage olup olmadığını şu şekilde anlar:

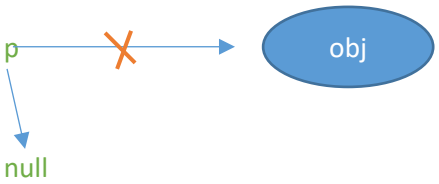
Bir nesneyi bellekte gösteren hiçbir değişken yoksa, yani o nesne bellekte erişilebilir durumda değilse kullanılmıyor demektir. O nesne çöp nesne olur.

Gösterim:



Bu gösterimde obj nesnesi garbage değildir çünkü p obj'yi referans etmektedir. objj'ye ulaşılabilir.

p = null yapılırsa;



Bu gösterimde obj nesnesi garbage'dır çünkü p obj'yi referans etmemektedir. obj'ye ulaşım sağlayabileceğimiz bir yol yoktur. Bu nedenle garbace collector obj'yi çöp olarak algılar ve bellekte tuttuğu alanı temizler.

C++ ile Karşılaştırma: Kullanılmayan nesnelerin bellekten temizlenme işlemi kullanıcı tarafından gerçekleştirilir. Fakat büyük projelerde ve karmaşık yazılımlarda kullanıcının bu işlemi yapması ona zorluk yaratabilir. free(), delete() gibi komutlarla dolu olan bellek alanının temizlenmesi gerçekleştirilebilir.

4. Spring frameworkünün kullandığı design patternlar neler?

Cevap:

Factory Method: Nesne oluşturmak için ara yüz oluşturuluyor. Fakat oluşturulacak nesnelerin türü farklı olabiliyor.

Singleton: Sistemin herhangi bir yerinde bir instance oluşturulmak istendiğinde ilk oluşturulan instance'ın yerine oluşturulur. Yani sistemde tek bir nesne olur ve bununla işlem yaparız.

Prototype: Sistemin herhangi bir yerinde instance oluşturulmak istenirse yeni bir nesne oluşturulur.

Proxy: Bir nesne için yedek tutulmasını sağlar. Orijinal nesneye erişimi kontrol edilebilir.

Template Method: Sıralı operasyonları içeren fonksiyonelliklerle ilgilenir.

Observer: Bir nesnede değişiklik meydana gelebilir. Bu değişiklikler diğer nesnelere bildirilir.

Mediator: Birbirleriyle ilişkisi bulunan nesne grubunu bir merkezden yönetmek için kullanılır.

Front Controller: Oluşan isteklerin bir yerde toplanması ve daha sonra isteklerin karşılanıp ilgili yerlere gönderilmesi ile ilgilidir.

5. Creational Patterns neler? Önceki ödevde oluşturulan nesnelerinizi Factory Design Patterni ile oluşacak şekilde düzenleyin.

Cevap: Creational Patterns, nesne oluşturulmasını sağlayan, durumlara göre farklı nesneleri oluşturmak için geliştirilmiş tasarım kalıplarıdır. 5 tane creational pattern vardır.

Singleton Design Pattern: Bir class'ın sadece bir instance'sının oluşturulmasını sağlar. Çalışma zamanında sadece 1 nesne yaratılmasına izin verir. Birden fazla sınıf aynı instance'yi kullanmak isteyebilir ya da sadece tek bir instance gerekebilir, her biri için new anahtar sözcüğü ile instance yaratmak maliyetlidir. Bu gibi durumlarda singleton design pattern kullanılır. Böylece bir instance'ın unique'liği sağlanmış olur ve erişim kontrolü sağlanır. Singleton Pattern'de bir nesne sadece ona ihtiyaç duyulduğunda yaratılır.

Factory Method Design Pattern: Program da birbirine benzeyen birden fazla sınıf olabilir. Bu tür sınıfları oluştururken her seferinde new keywordünü kullanmak istenen bir durum değildir. Birden çok aynı özelliği gösterebilecek sınıflar için gerekli nesne üretiminin kalıtım yolu ile yapılmasını sağlar.

Factory design pattern, class yaratma üzerine kurulu bir yapıdır. Benzer yapıda nesne üretimlerini gerçekleştiren sınıfları bir soyut sınıftan türetilir.

Abstarct Factory Design Pattern: Factory Design Pattern yapısında, birbirleri ile benzer nesnelerin üretimini tek bir sınıfa bağlıdır ve tek bir Interface üzerinden işlemler gerçekleştirilir. Abstract factory yapısında ise benzer nesne üretimlerini gerçekleştirebilmek adına her nesne için ayrı bir fabrika sınıfı oluşturulur. Birden fazla Interface kullanımı ile gerçekleştirilir.

Builder Design Pattern: Nesne yönelimi programlamda temel unsurlardan biri class'lardır. Class'lar ile nesneler yaratılır. Bunun için de class'ların constructor'ları kullanılır. Bir class için farklı parametreler ile çalışan constructor'lar yazılabilir. Eğer fazla parametre olursa constructor kullanımında karışıklık meydana gelebilir. Bunun önüne geçilmesi için Builder design pattern'ı kullanılır. Builder design pattern, nesnenin constructor kodunu kendine has bir sınıfa dönüştürür. Böylece karmaşık constructor yapılarından kurtulunur.

Prototype Design Pattern: Benzer bir instance kullanıp gereksinimlere göre değişiklik yaparak yeni bir instance oluşturulmasını sağlar. Var olan nesnelerin kopyalanmasına izin verir. Projede bir nesneden birden fazla kez oluşturulması gerekirse new keyword'u yerine daha önceden oluşturulmuş bir nesnenin klonu oluşturulur.