1. **MVC kavramını açıklar mısınız? Neden ihtiyaç duyuluyor? Java’da nasıl kurgulanıyor? Object** **Oriented katmanları nelerdir?**

MVC (Model, View, Controller), yazılım geliştirme sürecinde kullanılan bir tasarım mirarisidir. Bu mimaride uygulama, Model, View ve Controller olmak üzere üç parçaya ayrılır.

**Model:** Uygulamada kullanılan verileri temsil eder. İş mantığının derlendiği kısımdır. Verileri tutar ve gerektiğinde güncellenmesini sağlar.

**View:** Kullanıcının gördüğü arayüz kısmıdır.

**Controller:** Model ile View arasında bağlantı kurar. 

* MVC, uygulamayı üç farklı parçaya ayırır. Bu sayede her parça kendi işlevselliğini sağlar ve bağımsız olarak test edilebilir.
* Uygulamanın farklı platformlarda kullanılmasını kolaylaştırır. x
* Her parça, diğer parçalardan bağımsız olarak geliştirilir ve sonunda birleştirilebilir. Böylelikle *spagetti* kodun önüne geçilmiş olur.
* Test edilebilir kod yazımını da bize sunar. Bakımı kolay hale getirir. Bu gibi sebeplerden kaynaklı, MVC’ye ihtiyaç duyulmaktadır.

Java’da ise şu şekilde kullanılmaktadır:

* Model: Verilerin tutulduğu kısımdır. Bu kısım, Java sınıfları şeklinde oluşturulur.
* View: Arayüzün tasarlandığı bölümdür.
* Controller: Model ile View arasındaki bağlantıyı sağlar. Veri işlemesini yapar ve sonucu View’e gönderir. Genellikle Servlet, Spring MVC ve JSF Managed Bean tercih edilir.

Object Oriented katmanları ise Encapsulation, Abstraction, Polymorphism ve Intheritance’dir. Java da OOP, sınıflar ve nesnelerin üzerine kurulur.

**Encapsulation** (Kapsülleme), bir sınıfın içerisindeki metot ve değişkenleri korumasıdır. Yani veri gizliliği diyebiliriz. Java, tanımladığımız değişkeni bizlere default olarak *public* sunar. Bu sayede her yerden erişilebilir hale gelir. Bu da uygulama için tehlike arz eder. Bunun önüne geçmek için kapsülleme kullanılır.

**Abstraction** (Soyutlama), önemsiz verilerin kullanıcıya gönderilmemesidir. Yani yalnızca temel ayrıntıların verilmesidir. Bir örnek verecek olursak, bir kullanıcı bilgisayarı çalıştırmıştır fakat nasıl kullanılacağını bilmemektedir. Nasıl çalıştığı, kullanımı hakkında bilgi sahibi değildir.

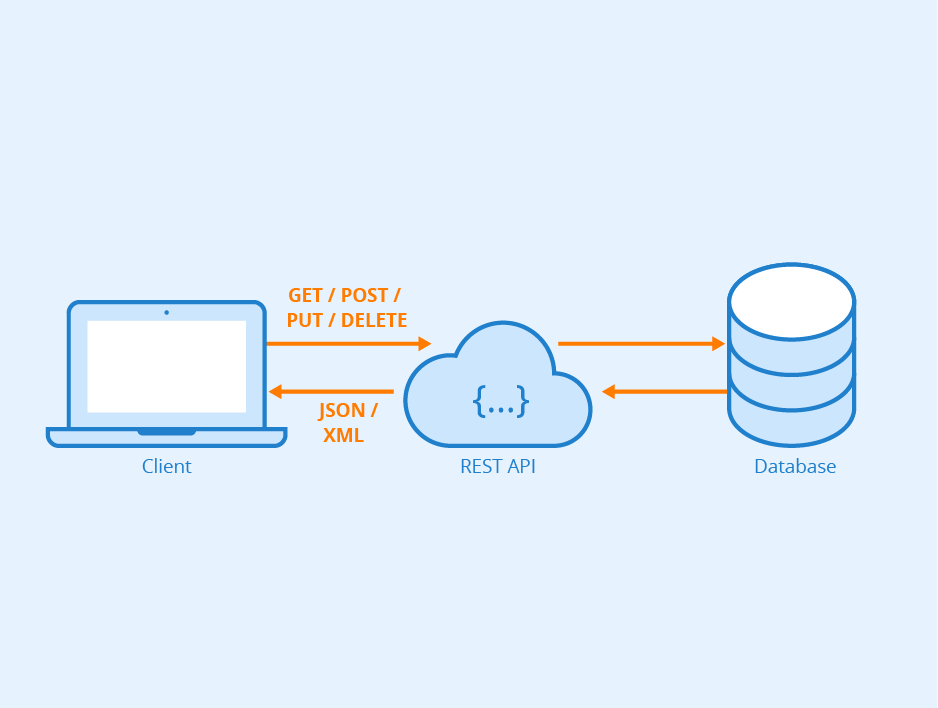
**Polymorphism** (Çok biçimlilik), yapılacak olan eylemi birçok farklı şekilde gerçekleştirmemizi sağlayan özelliktir.

**Inheritance** (Miras, kalıtım), bir sınıfın başka bir sınıftan özelliklerini miras almasıdır. Örnek verecek olursak, bir ‘Customer’ sınıfı düşünelim. Bu sınıfa, ‘int id’, ‘String name’ ve ‘String mail’ değişkenlerini tanımlayalım. Bir de ‘Employee’ isimli sınıf oluşturalım. Bu sınıfa ise yine ‘int id’, ‘String name’ ve ‘double salary’ değişkeni atayalım. Gördüğünüz üzere ikisinin birleştiği noktalar (id ve name) ve birleşmediği (mail ve salary) noktalar mevcut. Fakat ikisi de bir birey, yani kişi. Bu yüzden Person isimli bir class daha oluşturabiliriz. Böylelikle Customer ve Employee sınıflarını Person’a extends edersek, intehirtance gerçekleştirmiş oluruz.

1. **Birbirinden bağımsız iki platformun birbiriyle haberleşmesi nasıl sağlanabilir? Örneğin, X platformu Java ile yazılmış olsun, Y platform u C# ile. Bu iki platformun bir biri ile iletişim halinde request-response ilişkisi kurması gerekiyor. Bu yapıyı nasıl sağlarız?**

Bu platformların haberleşebilmesi için Restful API kullanabiliriz. Programlama dilinden bağımsız olarak, web servis oluşturmak için kullanılan web servis yöntemlerinden biridir. HTTP üzerinde çalışır ve diğer alternatiflerine göre daha basittir. İstemci ve sunucu arasında XML ve JSON verilerini taşıyarak uygulamaların haberleşmesini sağlar.

Rest mimarisine şöyle bir göz atalım:



**İstek ve Yanıt (HTTP Request — Response):** İstek, bir web hizmetinin girdisidir; yanıt ise çıktısıdır.

**Mesaj Değişim Formatı** (**Message Exchange Format):**İstek ve yanıtın formatıdır. İki adet popüler mesaj alışveriş formatı vardır. XML — JASON.

**Servis Sağlayıcı — Sunucu (Service Provider-Service):**Web servisini barındırandır.

**Tüketici — Müşteri (Consumer-Client):**Web hizmetini kullanan kişidir.

1. **Bir web sayfasında ekran sürekli Backend’ den veya bir başka yapı tarafından güncelleniyor. Siz, web sayfasını refresh etmeden bu güncel bilgiyi anlık ekrana nasıl yansıtırsınız?**

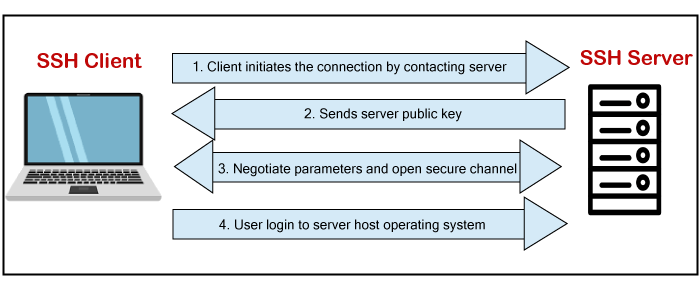
**WebSocket** ve **SSE** (Server-Sent-Events) ile bunu yapmak mümkündür.

WebSocket API, sunucu ve istemci arasında gerçek zamanlı ve iki yönlü bir işlem sağlar. WebSocket kullanmak, sunucu ve istemci arasında yüksek ölçekli veri aktarımlarını işlemenin iyi bir yoludur. WebSocket’in temel farklarından biri, HTTP’de olduğu gibi ayrı bir istek göndermek zorunda kalmadan veri almanıza izin vermesidir. Yine HTTP’nin aksine *ws://* veya *wss://* ile başlar. WebSocket, durum bilgisi olan bir protokoldür. Yani istemci ile sunucu arasındaki bağlantı, taraflardan biri oturumu sonlandırıncaya kadar canlı kalır. Sunucu, istemciye anlık olarak güncellenen verileri gönderir ve istemci bu verileri alarak web sayfasında anında görüntüleyebilir. Bu yüzden web sayfasını refresh etmeden bu güncel bilgiyi anlık ekranında görebilir. Gerçek zamanlı web uygulamaları, değişim platformları, oyun uygulamaları, sohbet uygulamaları, push bildirimleri gibi platformlar, yaygın kullanım örneklerindendir.

SSE ise, sunucunun istemciye tek yönlü olarak veri göndermesini sağlar. İstemciye anlık olarak güncel veri gönderilir ve web sayfasında bu veriler anlık olarak görüntülenebilir. Bu yöntem WebSocket’e göre daha az esnek olsa da bir çok durumda yeterli olduğu söylenebilir..

1. **Firmada çalışman için sana remote bir linux server verildi. Elinde ip adresi port bilgisi kullanıcı adi ve şifren var. Servera erişimi nasıl test edersin? Server a nasıl erişirsin? Server a nasıl dosya atarsın? Serverdan nasıl dosya çekersin?**

SSH (Secure Shell) ile bu mümkün. SSH, sunucularını internet üzerinden kontrol etmesini ve düzenlemesini sağlayan, uzak yöntem protokolüdür. Linux ve macOs kullanıcıları, işletim sistemlerindeki terminal penceresi ile uzak sunucularına SSH bağlantısı kurabilirler. Shell komutları ile sanki fiziksel olarak bağlıymış gibi control edebilirsiniz.



Örnek olarak, “ssh kullanici\_adi@ip\_adresi -p port\_numarasi”. Burada "kullanici\_adi" ve "ip\_adresi" verilen bilgilerle, "port\_numarasi" ise SSH bağlantısı için kullanılan port numarasıdır.

Dosya yükleme işlemi için, "scp" (secure copy) komutu kullanılabilir. Örneğin, scp file.zip root@192.168.10.10:/backup/file.zip, bu komut sunucudaki "/home/kullanici\_adi/" klasörüne "local\_dosya.txt" dosyasını yükleyecektir.

Dosya alma için, “scp [options] [username@]remote\_host:remote\_file local\_file”. Bu komut ile dosya alma işlemi gerçekleştirilebilir.

1. **Apache Solr servisine yazılacak bir query örneği Apache Solr kullanılan sql programlarından daha farklı runtime bir database. Solr a hali hazırda kayıtlı bir alan olduğunu düşünelim. Alanın ismi “updatedAt” long tipinde tutulan bir alan. Ben 2020 Ocak ayından sonraki verileri getir dediğimde solr a nasıl bir query yazılmalı. http://example?query= kısmını nasıl doldurmalıyım?**

2020 Ocak ayından sonraki verileri getirmek için, *“updatedAt:[1577836800000 TO \*]”* bu soru kullanılır. 1577836800000, 2020 Ocak ayının başlangıcına denk gelen Unix zaman damgası değeridir. “TO”, iki tarih arasındaki zaman aralığını belirtmek için kullanılır. “\*” Karakteri ise wildcard karakteridir. Sorgu esnekliğini sağlamak için kullanılır ve belirli bir kelime öbeği içerisindeki herhangi bir karakter ya da kelimenin yerini alabilir.

1. **Bir for döngüsü ile aşağıdaki çıktıyı yazar mısınız:**

