Concurrent programlama ve Parallel Programlama nedir ? Aralarında çalışma şekli olarak nasıl bir fark bulunmaktadır ?

Paralel hesaplama, ya da **Koşut hesaplama**, aynı görevin (parçalara bölünmüş ve uyarlanmış), sonuçları daha hızlı elde etmek için çoklu işlemcilerde eş zamanlı olarak işletilmesidir. Bu fikir, problemlerin çözümünün ufak görev parçalarına bölünmesi ve bunların eş zamanlı olarak koordine edilmesine dayanır. Paralel hesaplama ile performans artar, büyük sorunlar daha az sürede çözülür ve bilimdeki gelişmeler paralel hesaplamaya gereksinim duyar.

Bu programlamayı kullanmanın iki önemli avantajı vardır. Birincisi, paralel yapılar kullanılırken tüm süreçlerin hızlandırılması, hızlı sonuçlara ulaşmak için hem verimliliği hem de kullanılan kaynakları artırmasıdır. Diğer bir yararı da paralel hesaplamanın eş zamanlı programlamadan daha uygun maliyetli olmasıdır çünkü aynı sonuçları elde etmek daha az zaman alır. Bu inanılmaz derecede önemlidir, çünkü çok büyük miktarda veriyi işlenmesi kolay veri kümelerinde toplamak veya karmaşık sorunları çözmek için paralel işlemler gereklidir.

Seri hesaplama olarak da bilinen sıralı hesaplama, tek bir işlemcinin, herhangi bir zamanda, her biri birbiri ardına üst üste binme olmaksızın yürütülen bir dizi ayrı talimatlara bölünen bir programı yürütmek için kullanılmasını ifade eder. Yazılım geleneksel olarak sıralı olarak programlanmıştır, bu daha basit bir yaklaşım sağlar, ancak işlemcinin hızı ve her bir talimat dizisini yürütme yeteneği ile önemli ölçüde sınırlıdır.

Mutex ve Semaphore kavramlarını açıklayınız.

Mutex Kullanımı:
Bir mutex karşılıklı dışlanma sağlar; ya üretici ya da tüketici anahtarı (mutex) alabilir ve çalışmalarına devam edebilir. Tampon üretici tarafından doldurulduğu sürece, tüketicinin beklemesi gerekir ve bunun tersi de geçerlidir. Herhangi bir zamanda, yalnızca bir iş parçacığı tüm arabellekte çalışabilir. Konsept semafor kullanılarak genelleştirilebilir.
Semafor kullanarak:
Bir semafor genelleştirilmiş bir mutextir. Tek tampon yerine, 4 KB tamponu dört 1 KB tampona (özdeş kaynaklar) ayırabiliriz. Bir semafor bu dört tamponla ilişkilendirilebilir. Tüketici ve üretici aynı anda farklı tamponlar üzerinde çalışabilir.

Java'da Error ve Exception arasındaki fark nedir ? Throwable ile ilişkileri nasıldır ? Hangi tür durumlarda Error hangi tür durumlarda Exception meydana gelebilir ?

Errorlar handle edilemez. Ancak exceptionları try-catch ile handle edebiliriz.
Exception istisnai durum, bu durumda aslında beklenen uygulamanın devam edebileceği ama ara sıra yaşansa da sorun olmayacağı yönünde.

Error yani hata ise bu durum yaşandığı zaman uygulamanın/işlemin devam edemeyeceği burada kalması gerekiyor dediğimiz durumlarda kullanılmalı.

Spring'te yer alan @Scheduled anotasyonunun kullanım amaçlarını ve kullanım şeklini açıklayınız.

Scheduling, belirli bir zaman dilimi için taskları(görev-işlem) yürütme işlemidir. İşletim sistemi bazında düşündüğümüzde bu threadler ile sağlanır. Threadler, processlerin altında çalışan işlem parçalarıdır. Bilgisayarda yapacağımız işlemleri concurrent ya da paralel şekilde çalıştırmamıza yararlar. Tek işlemci, tek core bir bilgisayarda çalışan threadler belirli bir sırayla time sharing yaparak işlemleri yürütürler. Buna da concurrency denir. Bu yapılan işlemlerin sıralamasının düzenlenmesi işlemi de bilgisayar bilimlerinde cpu scheduling olarak geçer. Spring Framework bize anotasyonları kullanarak, neredeyse hiç threadler için kod yazmadan tasklarımızı yazmamızı sağlar. Yani bizi daha da soyutlar, altta yer alan detaylar ile uğraştırmaz. Periyodik olarak yapmak istediğimiz metodun başına @ Scheduled anotasyonunu eklememiz yeterli olur.

Uygulamamıza @ EnableScheduling anotasyonunu ekleriz. Zamanlamak istediğimiz metoda da @ Scheduled anotasyonunu ekleriz. Burada 3 tip Schedule bulunmakta. Bunlar:

- fixedDelay : Sabit gecikme süresi ile, taskımız bir önceki taskın bitiminden sonra belirlediğimiz süre kadar bekler. Yani 'fixedDelay=5000' dediğimizde önceki taskın bitiminden 5 saniye sonra yeni task başlar. Tasklar arasında 5 saniye olduğundan emin oluruz.

- fixedRate: Sabit belirlenen süre ile, taskın gerçekleşeceği zaman bir önceki taskın başladığı zamandan hesaplanır. Yani 'fixedRate=5000' dediğimizde önceki task başladıktan sonra 5 saniye içerisinde bu task gerçekleşecek demektir.

- cron: Aynı Linux'da bulunan cron yapısını burada da kullanabiliriz. Basit bir cron expression ile işlemlerimizi istediğimiz belli bir periyoda zamanlayabiliriz.

Spring'te yer alan @Async anotasyonunun kullanım amaçlarını ve kullanım şeklini açıklayınız.

Bir bean'in methoduna @Async annotationu eklemek onun main thread'den farklı olarak ayrı bir threadde çalıştırılmasını sağlar.
Yani call edilen methodun tamamlanmasını call eden kısım beklemez. Kod böylece async olarak çalışmış olur.

High Availability (HA) kavramını kısa açıklayınız.

Entity ve Value Object kavramlarını Domain Driven Design (DDD) kapsamında açıklayınız.

DDD, geliştirilmiş bir teknoloji veya belirli yöntem değildir. DDD, karmaşık yazılım sistemlerinin geliştirilme aşamasında ve bu karşmaşık projeler hayata geçtikten sonra uygulamalarımızın sürekliliğini sağlamakta sıkça yaşanan temel problemlere çözümler getirmeye çalışan bir yaklaşımdır.

Ubiquitous Language kavramını DDD kapsamında açıklayınız.

Core Domain, Supporting Domain, Generic Domain kavramlarını DDD kapsamında açıklayınız.

Anemic Domain Model ve Rich Domain Model kavramlarını kıyaslayarak açıklayınız.