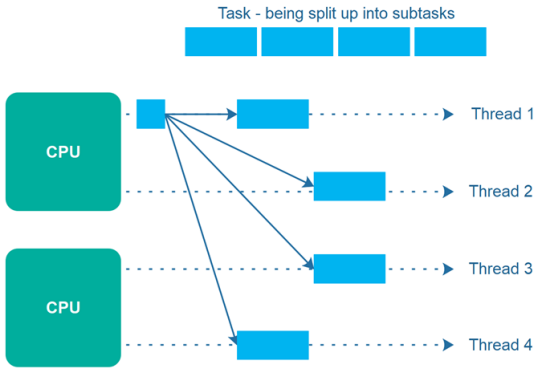
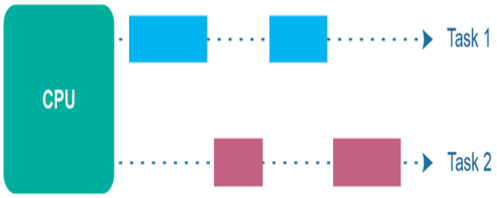
1. Concurrent programlama ve Parallel Programlama nedir? Aralarında çalışma şekli olarak nasıl bir fark bulunmaktadır?

Concurrent, aynı anda birden fazla işin tamamlanmaya çalışılmasıdır. Uygulamanın içinde verilen görevleri tek tek yapmak yerine, hepsini bir düzen içinde yapmaya çalışır. Tek bir işlem birimi kullanılır, sistemin yanıt verme süresi arttırılmaya çalışılır.

Parallelism, uygulamanın içindeki bir görevin birkaç ayrı alt görevlere bölünmesiyle yapılır. Bur da birkaç işlemci çekirdeği kullanılarak, görevler bu işlemci çekirdeklerine dağıtılır. Resimde de gösterildiği gibi bir görev 4 ayrı göreve bölünmüş ve bu alt görevlerde 2 farklı işlemciye dağıtılmıştır.



Parallelism



Concurrent

2. Mutex ve Semaphore kavramlarını açıklayınız. Hangi tür durumlarda bunlara başvurmamız gerekir?

Mutex bir kilitleme mekanizmasıdır. Bir kaynağa erişimi senkronize eder. Bir iş parçacığı o kaynağa erişmek istediğinde, bir sinyal gönderir ve kaynağa erişebilir, thereadin kullanıma giren bu kaynak thread işini bitirene kadar tekrar erişilemez, thread işini bitirdiğinde erişilebilir hale gelir. Bir mutex yanlızca bir thread tarafından sinyallenebilir.

```
wait (mutex);
…..
Critical Section
…..
signal (mutex);
```

Semaphore; Birden fazla sürecin(process) eş zamanlı çalışması halinde birbirleri için risk teşkil ettikleri kritik zamanlarda (critical sections) birbirlerini beklemesini sağlayan bir mekanizmadır. Semaforlar üzerinde kilitleme ve kilidi kaldırma yerine, semafor değerini artırma ve azaltma şeklinde işlemler yapılabilir.

Mutex önem arz eden bir kaynağı threadların sırayla kullanması gereken yerlerde kullanır.

```
mutexWait(mutex_mens_room);
      // Safely use
mutexRelease(mutex_mens_room);
```

Semaphore ise threadlar arasında sinyal göndemek veya sinyal almak için kullanılır. Ayı zamanda bekleme işlemlerinde, sayma işlemlerinde kullanılır.

```
* Uygulama makineye başlama komutunu verir*/
   semPost(sem_power_btn);   // Send the signal

/* Task 2 – Kullanıcı uygulamaya kapatma komutunu verir */
   semPend(sem_power_btn);  // Wait for signal
```

Kapatma sinyali gelene kadar sinyal beklenir. Böylece threadlar arasında sinyal alişverirşi sağlanmış olur.

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

Uygulamada bir bilgini, bir kod parçasının, bir dosyanın olmaması bariz bir şekilde uygulamayı çalıştırmayacaksa error fırlatılır ve uygulamanın akışı durdurulur. Örneğin bir configiration dosyası olmadan uygulama aya kalkamaz. Exception ise bilgini olmaması uygulamanın çalışmasını etkilemiyorsa kullanılır. Exception uygulama çalişken tolere edilebilir hatalardır, uygulamanın çalışmasını etkilemezler. Örneğin kullanıcı id bilgisi vererek, kişinin ismini veri tabanından almak istiyor. Girilen id ile veri tabanında bir eşleşme sağlamadığında excepiton kullanılır.

ThrowableException bir süpercllass'tır. Exception ve Error sınıfları Throwable sınıfından türerler.

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

Projede belli periyod ile düzenli olarak tekrar eden işlemler yapmak için kullanılır.

3 farklı şekilde kullanılabilir,

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.

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

Bu anatasyonla işaretlediğimiz metodlar, bir threadın içinde çalışır ve bu metodu çağıran metod işaretlediğimiz metodun bitmesini beklemez, yani arka planda çalışır.

Anatasyon public metodlara uygulanmalıdır.

6. High Availability (HA) kavramını kısa açıklayınız.

High Availability (HA), bir sistemin belirli bir süre boyunca arızalanmadan sürekli çalışma yeteneğidir. Birkaç karakteristik özelliği vardır.

Redundancy: bir yedekleme bileşeninin başarısız olanın yerine geçmesini sağlar. Bu olduğunda, veri kaybetmeden veya performansı etkilemeden X bileşeninden Y bileşenine geçiş eylemi olan güvenilir geçiş veya yük devretme sağlamak gerekir.

Failure detectability: Arızalar görünür olmalıdır ve ideal olarak sistemlerde arızayı kendi başlarına halletmek için yerleşik otomasyon bulunur.

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

Bir kimliği olan objelere entity denir. Bu objelerin bir feature leri olmalıdır ve bu feature eşşiz olmalıdır. Örneğin bir kişiler için bir class oluşturulduğunda, TC kimlik numarası bir eşsiz özelliktir. Bu class böylece bir entity olacaktır.

Bir entity'in ıd leri kullanıcı tarafından veya otomatik olarak sistem tarafından oluşturulabilir. Aynı zamanda bir entity birkaç ıd ye de sahip olabilir. Örneğin sosyal güvenlik numarası.

Entity yapmak maliyetlidir. Tüm objeleri entity yapmamak gerekir. Bu nedenle value object kullanılır.

Value Object'ler'in identity'si yoktur. Sadece data transferinde kullanılan dummy objelerden ziyade içerisinde bazı business'lar bulundurabilir. Entity'leri bu tarz hesaplamalardan ve business logic'lerden arındırmak için kullanılırlar. Bu sayede entity'ler kendi yapması gereken işlere odaklanır. Ancak işin kötü yanı value object'ler içerisinde bir miktar functional programming yapılabilmektedir.

8. Ubiquitous Language kavramını DDD kapsamında açıklayınız. Sizce neden önemli olabileceğini belirtiniz.

Bir proje de uzmanlar ve geliştiriciler vardır. Projenin uzmanı, o alandaki her şeyi ayrıntılı olarak bilir, mesleğinin getirdiği kendi dilinde konuşur. Örneğin bir doktor bir hastalığı tanımlarken latince ifadeler kullanabilir. Aynı şekilde uzmanlar da geliştiricilerin kullandığı ifadeleri bilmeyebilir. Uzman kişi kütüphaneler veya kullanılacak framework hakkında bir bilgiye sahip olmayabilir.

Bu nedenle geliştiriciler ile proje uzmanları arasında ortak bir dile ihtiyaç vardır. Bu ortak dil oluşturulurken dikkat edilmelidir ve aceleye getirilmemelidir. Kullanılan ortak dil geliştirmeye açık olmalıdır.

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

Core domain bir organizasyonu diğer bir organizasyondan farklı yapandır. Core domain en önemli domaindir ve en çok zaman harcanması gereken domaindir. Bu domain olmadan veya başarılı olmadan organizasyon var olamaz. Subdomain lerin tek bir core domainin vardır, daha fazla subdomainleri olamaz.

Supporting domain bir subdomaindir ve uygulamanın başarılı olması için gerekli olan ancak core domain kategorisine girmeyen bir alt etki alanıdır. Organizasyondaki sorun için az seviyede uzmanlık gerektirir. Mevcut bir çözümle başlayabilir ve ince ayarlar yaparak geliştirebilir veya özel ihtiyaçlara göre genişletilebilir.

generic subdomain, organizasyona özel hiçbir şey içermeyen ancak genel çözümün çalışması için yine de gerekli olan bir subdomaindir. Generic domain için kullanıma hazır yazılımları kullanmaya çalışarak çok zaman kazanabilir . Tipik bir örnek, kullanıcı kimliği yönetimi.

http://tutorials.jenkov.com/java-concurrency/concurrency-vs-parallelism.html#concurrency

geeksforgeeks.org/difference-between-concurrency-and-parallelism/#:~:text=Concurrency%20is%20the%20task%20of,of%20running%20multiple%20computations%20simultaneously.