Ismet Mertkan YILMAZ

Payten Java Spring Bootcamp HW1.1

First Encounters

Unix pipeline, smalltalk ve ruby pipeline dizaynı fonksiyonların lineer bir şekile sıralanmış halde verilen nesneye tanımlanan işlemleri yapmalarıdır. Bu dizayn bir boru hattında soldan giren bir objenin sağdan çıkışıtan gerekli işlemleri görüp çıkması gibi düşünülebilir. Eskiden insanların collection pipelineların OOP'nin bir elemanı olmadığını söyleemesinin sebebi, Smalltalk'ın lambda fonksiyonları kullanımının öenmli OOP dillerinde kendine yer bulamamasıydı. Bugün ise collection pipelibe yapısı C#, Java gibi dillerde mevcuttur.

Defining Collection Pipeline

Collection pipeline bir adet collection yapısını alır ve fonsiyonları uygulayarak başka bir colletiona çevirir. Bu yapı üzerinde basit fonksiyonlar ile karışık yapılı birçok işi yapan fonksiyonlar kurulabilir. Farklı dillerde collection yapıları farklı olabilir ama temel mantık aynıdır. Fonsiyonel diller ve OOP diller farklı itemleri bu yapıda kullanabilirler. Mesela OOP'de objeler üzerinde işlem yaparken, fonksiyonel dillerde hashmap'ler üzerinde bu işlemler yapılır.

Exploring more pipelines and operations

Farklı dillerde, benzer/aynı collection pipeline yapıları vardır. Bunlarda bazıları şöyledir;

Map & Reduce ➔ Map ile farklı collection yapısı içinde yer alan elemanlara istediğimiz işlemi uygularız. Reduce ile de elde ettiğimiz (mesela map ile her elemanına işlem yaptığımız bir collection düşünelim) yapıyı tek elemana düşürürüz. (bu collection içindeki sayıları toplayabiliriz ve tek sonuç alırız.)

Group-by ➔ Bu operasyon sayesinde bir collection yapısı içinde yer alan objeleri, tiplerine belli özelliklerine göre sınıflandırıp key:value yapısına dönüştürebiliriz. Bu yapıda genel problem genelde input yapı listedir ve hash yapısı olabilir. Eğer böyle olursa bu yapıyı ikili liste gibi düşünüp gerekli çevrim yapılıp devam edilir.

Laziness

Laziness konsepti sayesinde bir collection üzeride sadece gerekli olan objeler için işlem yapılır. Normal koşullarda bi collecionun tamamı işlemden geçirilip sadece işimize yarayacak olanlar alınacaksa, bu konsept sayesinde daha performanslı bir method elde ederiz. Bazı dillerde bu konsept direkt olarak dilde hazır implementasyonlar ile kulllanılırken bazılarında bunun kullanımı mmkün değildir.

Parallelism

Parallelism sayesinde eğer çok çekirdekli bir yapıda çalışıyorsak ve özellikle büyük setler üzerinde çalışıyorsak, işlemleri threadlere bölerek aynı anda bu setin objeleri üzerinde paralel işlem yapıp performans sağlanabilir. Fakat bazen bu yapıyı kurmak daha maliyetli olacağı için perormansı düşürebilir. Kesinlik için performans testlerinin yapılması gereklidir.

Immutability

Colletion pipelinelarda her işlem aslında bir nevi sonuç olarak başka bir collection oluşturur. Doğal olarak bu sonuç çok fazla kopyalama ve büyük datalarla çalışırken sorunlara neden olur. Bu çoğunlukla pointerlar kopyalandığı için bir sıkıntı yaratmayacaz ama yaratırsa da immutable olmasını sağlayacak data yapıları ile çalışılarak sorun giderilebilir. Bazı durumlarda da eğer gerekli ise immutablity korumak yerine, collectionu sürekli update eden işlemler kullanılabilir.

Debugging

Debug işlemi için birkaç yöntem uygulanabilir. IDEnin bunu çözmesi sıkıntı olduğunda bunları biz uygulayabiliriz. Pipelineın yarısında sonucu alabiliriz. İşlemlerden sonra print ettirebiliriz. Tabi ki IDE içinde debug uygulamak da kullanılır ve break-pointler ile izlenir.

Alternatives

Collection pipeline yapısı ile yaptığımız işlemler başka şekillerde de icra edilebiliyor. Looplar, kıyas işlemleri/operatörleri kullanılarak collection pipeline'a alternatifleri yapılabilir. Fakat bu uygulamalrda göze çarpan nokta, collection pipeline kullanmak temiz, anlaşılır ve kısa kod konusunda daha fazla öne çıkıyor.

When To Use It

Collection pipelinenın her design patterned olduğu gibi kullanılması makul ve makul olmayan yerler vardır. Mesela Java'da eskiden lambda fonksiyonların olmadığı zamanlarda bu yapıyı kurmak temiz olmayan syntax olarak karıiık bir hal alan kodlar oluşmasına her yerde loop kullanımına sebep oluyordu. Bu durumlarda astarı, yüzünden pahalıya geldiği için kullanmak makul değildir. Dil buna el verişli olsa bile boyut ve karışıklık sebebi ile bazen kulllanmak mantıklı olmayabilir. Büyük ve karışık pipeline yapıları ayrı ayrı methodlara bölünebilir.