COLLECTIONS&LINKEDLIST (45.VİDEO ÖZETi) 17 ARALIK 2021

Önceki Dersten Aklımızda Kalanlar

1) Iterator: Collection’larda elementler arasında gezinmemizi sağlar, normalde index kullanan yapilar icin index ve loop ile elemanlar üzerinde gezinebilir ve değişiklikler yapabiliriz, ancak collection’da her yapı index kullanmadığından iterator kullanmamız gerekir

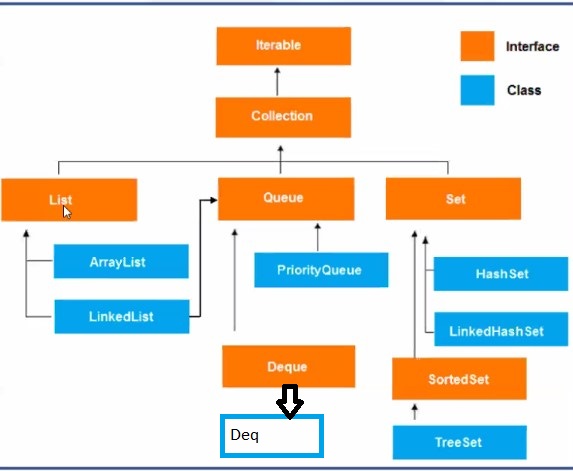
2) Iterator interface’dir dolayısıyla kullanacağımız collection class’inda iterator method’lari override edilmiştir. Biz collectionIsmi.iteratorMethod() diyerek kullanacağımız collection class’inda bulunan ilgili iterator method’unu çağırmış oluruz.

3) Iterator interface’inde sadece 3 method hasNext( ) : iterator’in bulunduğu konumdan hemen sonra eleman var mi ? Diye sorar ve true veya false döner next( ) : iterator’i bir sonraki konumuna hareket ettirir ve üzerinden geçtiği elemanı bize dondurur remove( ) : son olarak üzerinden geçilen elemanı siler.

4) ListIterator da Iterator interface’inin child interface’dir. ListIterator parent’I olan Iterator’a göre daha fazla method’a sahiptir. Set ( ) method’u üzerinde olduğumuz collection elementini kalıcı olarak değiştirmemizi sağlar Iterator ’da hep ileri doğru hareket ve bununla ilgili method’lar olmasına karşılık ListIterator’da next ile birlikte previous method’lari da vardır. Dolayısıyla ListIterator kullanarak ileri ve geri hareket edebiliriz.

5) Iterator ile çalışırken dikkat etmemiz gereken en önemli konu biz ileri veya geri demedikçe iterator son biriktiğimiz yerde bekler

Collections nesnelerden oluşan bir topluluğu bir arada tutan yapıdır.



Bu yapıları Java farklı farklı biçimlerde yapılandırmıştır. Collection Iterator’ın child’dır.

Örneğin bir eczane de sürekli yeni ilaç gelir şayet hep en son gelen satılırsa ilk gelenlerin son kullanma tarihi geçer. Başka bir durumda ise mesela lokantada domates hep en son gelen kullanılırsa ilk gelenler bozulur, bu durumda ilk geleni kullanmak en avantajlı olan tercihtir.

Bazı durumlarda ise; dayanıklı ürünleri örn. Yangın tüpü gibi bir malzeme olduğunu düşünürsek gelenin sırlaması değil varlığı yokluğu önemlidir. Mesela çelik boru yeni ya da eski olanın kullanılmasının bir önemi yoktur.

Farklı iş kolları için farklı yapılar vardır. Java kullanılacak programın alt yapısına bağlı olarak alternatif sunar. Varlık ve yokluğu kontrol edilir ister sondan kullanılır ister baştan..

Collection konusunda odaklanmamız gereken nokta hangi soruyu çözeceğimizden ziyade bunların birbirlerinden farkları

Collections altında 11 yapı vardır,5’i interface’dir dolayısıyla hepsiyle obje üretilmez.

Interface ve class’ları nasıl ayırt edersiniz?

Iterable yapısında interface ve calass’ları tek tek ezberlemek zor olabilir. Interface’den obje oluşturulmaz. Şayet bir constructor oluşturulduğunda hata veriyorsa o bir interface’dir,onun yerine child’ları kullanılır

Collection yapısı 3 çeşittir.

1-LIST : kullanılan yapıda bazen listelemek işinizi kolaylaştırır. Kullanmaya en aşina olduğumuz yapıdır.

2- QUEUE: Kuyruk demektir. Son gelen en arkaya gider. Eczane örneğinde olduğu gibi önce gelen önce kullanılır

3-SET : Elimizdeki malzemenin varlık ya da yokluğuna bakar, kümenin içinde varsa bir daha eklemez . Yangın tüpü gibi örneğindeki gibi. Bu 3 yapı temelde özellikleri belirler.



1-SET(Küme) : Listede varsa bir daha eklemeye gerek yok. Yangın tüpü örneğinde olduğu gibi varsa bir tane daha almaya gerek yok.. mantığıyla çalışır

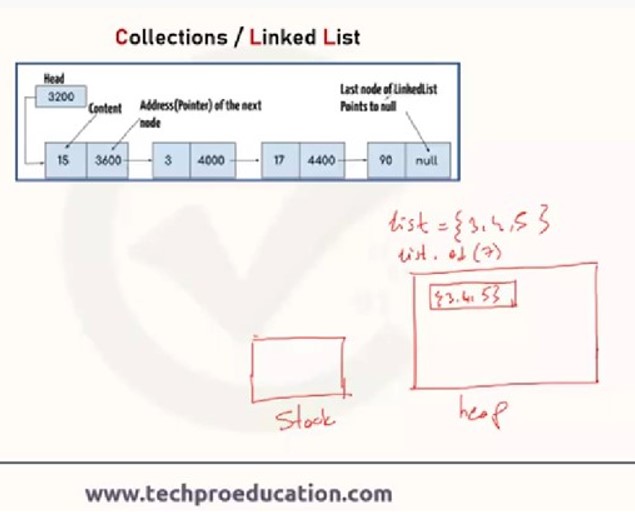
2-QUEUE(Sıra) : Sıralı olarak bir kuyruk vardır. Yapı itibariyle ister baştan ister sondan kullanılır.

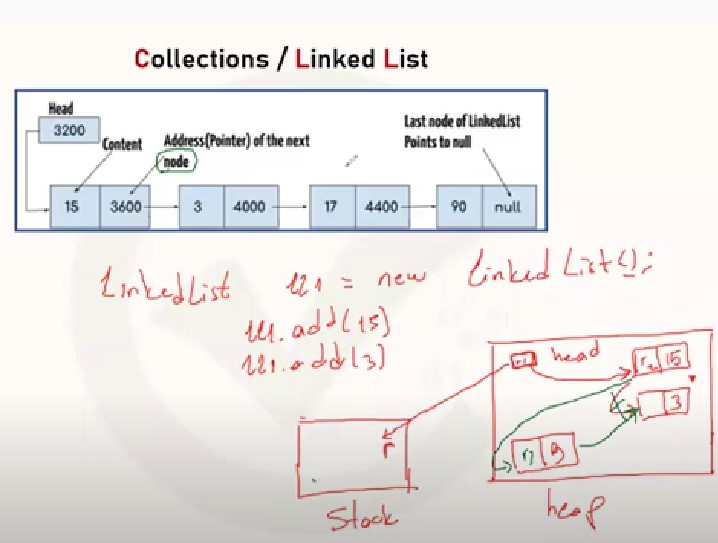
3-LINKED(Bağlı) : Birbirine bağlı demektir. Araya giren eleman hemen oraya entegre olur. Halay örneğinde olduğu gibi,baş-orta-son her yere olabilir. Neticede birbirine bağlı olacaktır.

4-TREE(Doğal Sıralı) : bir collection’ın içinde tree varsa orada doğal bir sıralama vardır. Örneğin a,d,z sıralamasında b gelirse a ve d’nin ortasına alır, eklerken index’e bakmaz, koyması gereken yere koyar

5- HASH : mantığı da tüm elementleri beli bir formüle göre , kendi çerisindeki mantığa göre basite indirger, bunları depolar. Ve bu mantıkla çabuk bulunur. Hash mantığı ile tek tek bakılmaz ..burada bir örnek vermek gerekirse karanlık bir sınıfta yerini bildiğiniz bir öğrencinin yerini hemen buluruz, şayet öğrencinin yeri bilinmiyorsa tek tek herkesi yoklamak gerekir. Hash kodu ve aranacağı yer belli ise ihtiyaç olan verinin bulunması çok kolaydır. Dolayısıyla neyin nereye koyulduğu çok önemlidir.

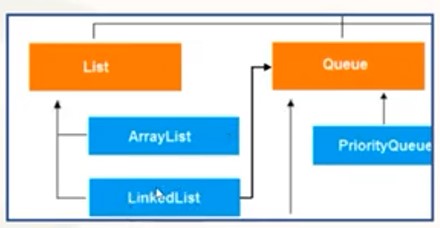
LINKEDLIST





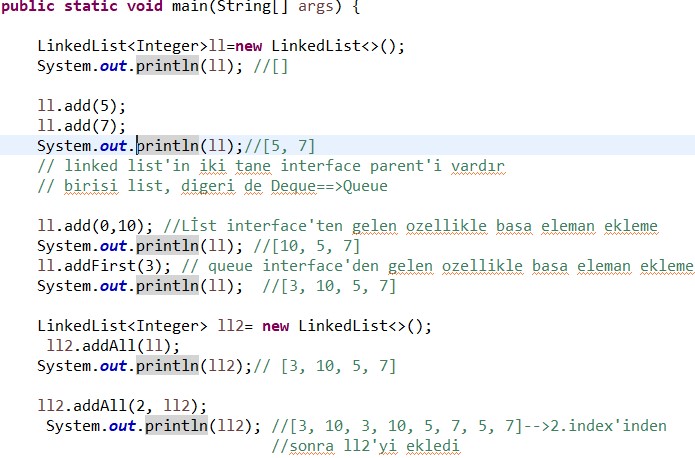
LinkedList her geleni sona ekler en büyük avantajı araya eleman eklemesidir. Linked list class yapısına sahiptir.

1. İlk eleman her zaman head’dir ve head’de data yoktur, sadece adres vardır. Head ‘de data yoktur. Head 2 değil tek bölümü vardır o da adrestir.
2. Son eleman (tail) null’ı point eder,adres kısmı null olur. Sona bir tane daha eleman eklenirse pointer kısmını null yapar.
3. Her eleman içinde data ve adress kısmı olmak üzere 2 kısım vardır.
4. Tüm elemanlar pointer’lar ve adresler kullanılarak birbirine bağlanır.
5. Her eleman NODE olarak adlandırılır.
6. Array’lerden daha dinamiktirler. Insert ve delete işlemlerinde başarılı bir yapıdır.
7. Pointer yapısından dolayı bir elemana ulaşmada yavaştırlar.. List’te örneğin 5. Elemana hemen gider. LinkedList’te ise 3. Elemana git denildiğinde önce 1.-2. Sonra 3.’ye gider. Dezavantajı da bu yönüdür.

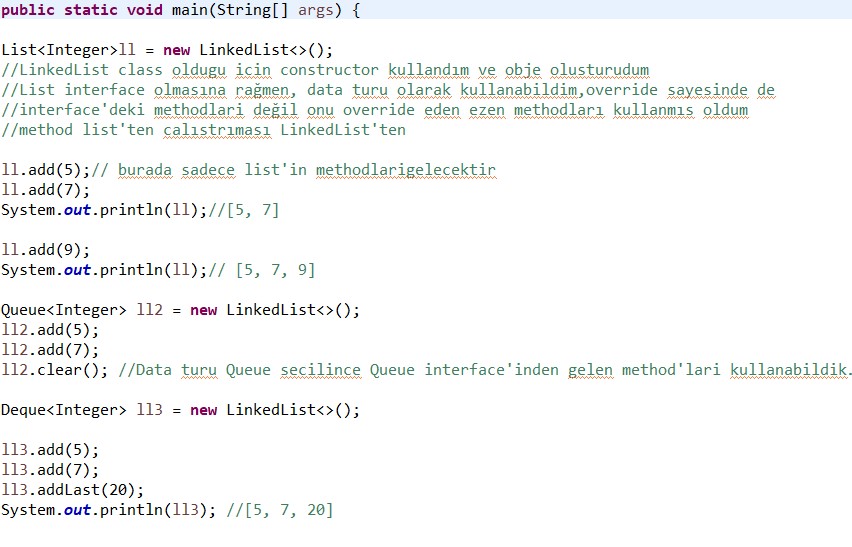


Linked List’in 2 tane pointer’ı vardır. Arrray List, parent’i olan Interface List’in tüm methodlarını override etmek zorunda ve sadece onları kullanabilir, o methodlara sahip. Linked list ise hem List’teki hem de Queue’deki methodlara sahip. Dolayısıyla daha fazla method ve özelliğe sahip.

Class1



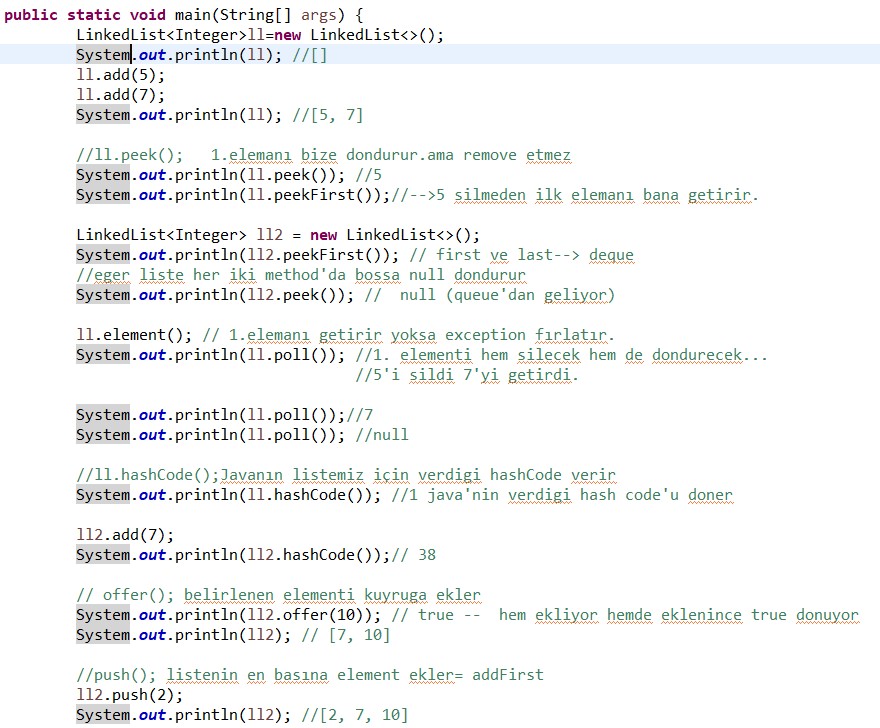
Class2



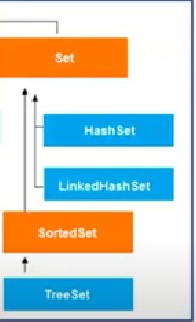
Yapacak olduğumuz proje de, kullanacağımız yapıyı belirleme özgürlüğü sağlar. Hangisi lazımsa onu kullanabililriz.Queue’nun özellikleri lazımsa data türü Queue seçilir. Data türündeki method’a gider, eğer o method override edildiyse yani ezildiyse, ezilen method’u değil, ezen method’u çalıştırır.

Deque interface olduğu için o method zaten abstract yani override edilmiş geçersiz kılınmış. Geçersiz kılan method LinkedList’te gidip o method’u çalıştırır. Anvantajı şudur ; LİnkedList’te olup List2te olanları çalıştırmaz. Deque’dan gelen methodları çalıştırır.

CLASS 3



SET



Küme mantığı ile çalışır. Var mı yok mu diye kontrol eder, yangın tüpü örneğinde olduğu gibi varsa almaya gerek yok. Örneğin takımda kaleci varsa 2.yi almaya gerek yoktur.

Bir eleman varsa 2.yi eklemez,yani unique’dir. Tek olmak zorundadır.

Bir list’te aynı elemanlardan birden fazla olanı çıkarmak istersek List ile bu uzun bir yöntemdir. Ama set’te elementleri tek tek al, Set’in içine atıp yoksa ekler varsa eklemez.

Dezavantajı da vardır, yığındır. Index’i yoktur. List’te index’le hallettiğimiz bazı işlemleri halledemeyiz.

Java elementleri unique yapmak için HASH ALGORİTMASI kullanır.

Index olmadan, set’in elemanlarının hepsini değiştirmeyi Iterator ile yapabiliriz. For each loop ile yazdırabiliriz.

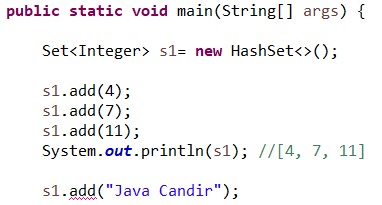
SET’ten obje üretemediğimiz için 3 seçenek vardır. Bu 3 yapı arasında hangisi bizim işimizi kolaylaştıracaksa tercihimizi ondan yana kullanırız

1. **HashSet :** Linked ve tree olması zaman kaybettirir. hashSet hızlıdır. hem map hem de Set’te HashSet kullanmayı tercih ederiz. Sıralama işlemi yazdıracağımız zaman önemlidir. Bir proje de bir çok kez ekleme-çıkarma-update buna benzer bir çok özellik yapılacaktır. Bu tür durumlarda şu şekilde bir çözüm üretilebilir; tüm işlemler HashSet olarak yapılır, yazacağımız zaman tree Set’e aktarılır. Sadece yazdırmada sıralama isteniyorsa bu şekilde hızlı bir yöntem ile denemek uygun olacaktır.

Bir method oluşturulup ihtiyaç halinde o method çağırılıp HASHSET ,TREESET’e çevirilip yazdırılır.

1. **LinkedList :** Elementlerin birbirine bağlılığı önemli ise kullanabiliriz. Ama işleyiş yavaştır. Elemente ulaşmak için 1.2.3.2ye tek tek bakmak vakit kaybettirir.
2. **TreeSet: sıralama** önemli ise kullanılabilir fakat ulaşmak yavaştır.

3 Child Class’dan bizim için önemli olan özelliğe göre istediğimizi kullanabiliriz.



Collcetions’ ın bir özelliği de farklı data türünden elementleri ekleyebilmemizdir. Bunun için eşitliğin sol tarafındaki <>(data türü) kaldırılabilir veya data türü olarak Object yazılabilir. Ancak bu çok tavsiye edilmez, çünkü Java’nın çok fazla Casting yapması gerekir.

