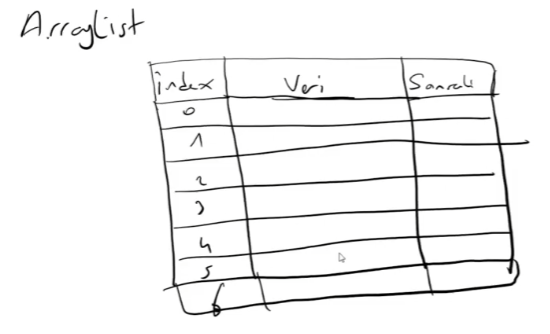
**Listenin Dizi Üzerinde Gerçekleştirimi**

* Anahtar kelime ArrayList’tir.
* Eğer Arraylist görürsen bu veri yapısının bir liste olduğunu ama arka planda dizi ile gerçekleştiğini bilmelisin.

**Düğüm Listesi (ArrayList)**

* Düğüm dizisi oluşturursan arraylist oluşturmuş olursun
* Aslında dizilerde index ve veri vardır, sonraki diye bir şey yoktur. Sonraki’ni veri ile birlikte gömmemiz gerekir



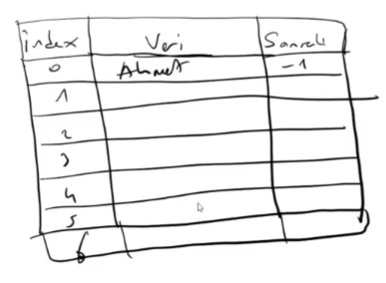
* Sonraki ve veri yi bir class, struct olarak ifade edebiliriz. Veri herhangi bir değişken tipinde olabilir ama sonraki int olmak zorunda

metin içeren bir resim

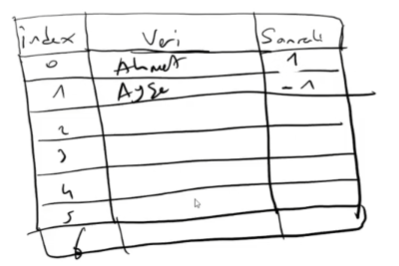
Açıklama otomatik olarak oluşturuldu

* Amacımız 2.yoldaki dezavantajları ortadan kaldırmak

Dizideki en son elemanın sonraki elemanı -1 olur. Ahmet’i diziye ekledik.



Ayşe’yi de eklediğimizde Ayşe -1 Ahmet 1 olacak



Murat’ı eklemek istiyoruz. İlk elemandan başlayarak tarayıp nerden sonra geleceğimizi bilmeliyiz. İlk eleman 0.index o zaman LB1=0 diye bir şey tutmamış gerek. Murat Ahmet’e gitti sonraki ifadesi 1 ve en son eleman olmadığını anladı. Ayşe’ye gitti sonraki ifadesi -1 liste sonu olduğunu anladı Ayşe’nin sonraki 2 yapıldı Ahmet kendine -1 yazdı



Araya Dilek’i ekleyelim. Kaydırma işlemi yapmıyoruz. Dilek bir sonraki hücreye yazılır. Ahmet’in sonraki ifadesi Dilek’i göstermeli aynı zamanda Ayşe de kopmamalı. Ahmetin sonraki ifadesini Dilek’e yazarız Ahmet’e de 3 yazarız

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yukarıdaki dizi üzerinde gerçekleştirim ama dizinin nimetlerinden yararlanamaz. 3.indexe git dediğinde mantık olarak 3.elemanı almak isteriz ama 3.index te dizinin 1.elemanı var. Ama bu senaryonun diğer avantajı da aynı dizi üzerine birden çok listeyi tutabiliriz.



Yukarıda LB1=0 için Ahmet, Ayşe, Murat’ı okudu. Murat=-1 orada durdu. LB2=4 için de -1’i görene kadar okur.

Diyelim ki Ayşe’yi silmemiz istendi. Ayşe’yi silmek için Ayşe’nin bir öncesine yerleşmemiz gerekir. Yani Dilek. Dilek in ilerisi Ayşe olmayacak, Ayşe’nin ilerisi olacak. Yani Ayşe’yi silmek için Ayşe’nin bir öncesini bulduk: Dilek, Dilek’in sonrakini de Ayşe’nin sonrakisine bağladık: Ahmet. (bu arada Dilek, Ayşe’nin yerini aldı). Ayşe dizide kalır ama onu referans alan kimse olmadığı için Ayşe silinmiş gibi olur



* Eleman ekleme ve silme de veri kayması olmaz.
* Liste veri yapısından bahsetsek de arka planda veriler dizilerdedir ama diziyi dışarıya açmıyoruz.

dizi[2]=”Dilek” gibi bir şey yok.

* Diziyi arka planda bırakıp metotlar yardımıyla yeni bir veri yapısı ortaya çıkardık: Liste. Listeler için ekle, sil, insert, remove metotları vardır bunlar üzerinden işlem yapılır. Bir dizi adı verilmez
* Sonuç olarak veri dizide saklanıyor, biz metotlar yardımıyla diziye mesaj gönderip liste veri yapısını oluşturuyoruz.

Yukarıdakinin kodunu yazmayacağız. Kodunu yazacağımız senaryo:

Sonraki diye bir şey yok. Elimizde bir dizi var ve ona modülerlik kazandırmak için metot yazacağız ve ortaya liste veri türü çıkacak. Dizi sabittir araya eleman ekleme, eleman silme gibi özellikleri yoktur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Ekle Metodu //void add()**

metin, spor içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eleman eklediğimizde en sona ekleyeceğiz. Şuan length=0 o zaman 0. İndexe ekledik ve sonra length=1 oldu

metin, spor içeren bir resim

Açıklama otomatik olarak oluşturuldu

0.indexe eleman ekledik length=1, bir daha eleman eklediğimizde 1.indexe eklicez ve sonra length=2 olacak bir daha eklersek 2.indexe. yani her serefinde length in gösterdiği yere ekliyoruz

**Araya Ekle //void insert()**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

2. indexe 25 ekleyelim. Direkt üzerine yazarsak 21 i kaybederiz. 2. indexten başlayarak sağa kaydırırsak 50’yi kaybederiz. En sondan başlayarak sağa kaydıralım. Önce 50yi sağa sonra 21’i sağa. 25’i de boşluğa ekleyelim

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sonuç olarak en sonran başlayıp eklenecek indexe kadar (o indiste dahil olmak üzere) tüm elemanlar ir sağa kayar. Eğer eleman sayısı çoook büyükse her elema nı sağa kaydırmak büyük performans kaybıdır. Eğer sürekli insert kullanılacaksa arraylist doğru bir seçim değildir. Ama genelde arraylist e add() (sona ekelme) daha çok kullanılır. İnsert nadir kullanılır.

**NOT**: Aradan eleman silmek ve araya eleman eklemek dizinin dezavantajlarından biridir. Çünkü eleman silmek diye bir şey yoktur elemanları kaydırırsın. Büyük dizilerde baya maliyetli olur. Bağıl liste bu konuda avantajlıdır.

----------------------------------------------------------------------------------------------------------------------------------------------

**NOT:**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

----------------------------------------------------------------------------------------------------------------------------------------------

NOT: sadece kütüphane dosyası yazarsan (.hpp) kaynak dosyası yoksa o zaman make de onu derlemene gerek yok. Sadece Test’i derlesen olur. Zaten include ediyorsun ve bulamadığı include ları nerede arayacağını söylüyorsun.

NOT: hata aldığında en baştan itibaren bak

NOT: temizleme çok sık yapılan bir şey ise bağıl liste yerine array list tercih etmelisin. Çünkü 100 bin tane elemanın varsa length=0 diyerek llisteyi silmiş oluyor. Bağıl listede ise çöp olmasın diye her birini tek tek siler. Bu konuda da arraylist linked liste göre daha avantajlıdır, hızlıdır

NOT: programlama dillerinin dökümanlararına bakmalısın. Hepsi aynı değil. Arraylist arkada dizi mi kullanıyor falan incemelisin

NOT: arryalist de iki elemanı değiştirme yok. Çünkü dizi private alanda erişemiyoruz. Ama ödevlerde standart veri yapıları dışında ek özellikler yazman beklenir. Mesela sen bu durumda ne yapabilirsin? Sınıf oluşturup arraylisten kalıtım alabilirsin private alandaki diziye erişemezsin, diziyi komple tekrar yazacaksın ve içine swap metodunda gönderilen ikili indexteki elemanların yerini değiştirecek. Bağılliste de bu bir tık daha zordur.

NOT: listenin dizi üzerinde gerçekleştiriminde iki farklı durumu bilmen gerekir. Birinin kodunu yazmadık (en baştaki) diğerinin kodunu yazdık.

**Dizi Hangi Durumlarda Avantajlı ?**

Add ---- evet

İnsert --- kesinlikle hayır (insert çok kullanılıyorsa bağıl liste kullan)

Remove – hayır (her zman sondan mı silenecek belli değil. Yine kaydırma var)

removeAt -- hayır

first -- farketmez

last – evet (sürekli son elemana gidiyorsak direkt bulabiliriz)

clear – çok evet

size – farketmez (ikisinde de eleman sayısı tutulur)

indexOf – farketmez. (ikisinde de tek tek dolaşıp elemanı buluyorsun)

print -- farketmez

elementAt --- indexi biliyorsan kesinlikle arraylist

* Baskın bir şekilde evet yazanlar kullanılıyorsa arraylist kullanmalısın