SAMSUN ÜNİVERSİTESİ
MÜHENDİSLİK ve DOĞA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ
2023 – 2024 Akademik Yılı Bahar Dönemi
MYAZ204 – Veri Yapıları ve Algoritmalar Dersi

16/04/2024
Ara Sınav

- Sınav süresi 60 dakikadır. Sınav süresince bilgisayar kullanımını sadece Visual Studio/Visual Studio Code ile sınırlandırılmıştır. Sınavda internet kullanılmayacaktır.
- Cevap kağıdınıza öğrenci numaranızı, şubenizi, adınızı ve soyadınızı yazınız. Birden fazla cevap kağıdı kullanmanız durumunda gözetmenlerden kağıtlarınızı zımbalamasını isteyiniz. Cevap kağıtlarına sadece sizden istenen bilgileri yazınız.

Doç. Dr. Zafer CÖMERT

SORULAR

1. Lütfen boş bırakılan yerlere gelmesi gereken ifadeleri cevap kâğıdınıza geçiriniz.

Veri yapıları, bilgisayar programlarında verilerin organizasyonu, depolanması ve yönetilmesi için kullanılan yapısal düzenlemelerdir. Veri yapıları, verileri bir araya getirirken, bu verilere nasıl erişileceğini, nasıl değiştirileceğini ve nasıl işleneceğini belirler. Bir veri yapısı modellenirken verinin ___(a)___ doğru gidebileceği ya da genişleyebileceği düşünülür.

___(b)___ bir veri yapısının matematiksel bir modelidir ve bu veri yapısının operasyonları ile bu veri yapısında saklanan verilere nasıl erişileceği, nasıl değiştirileceği ve nasıl kullanılacağı gibi özellikleri tanımlar. Bu kavram, bir veri yapısının davranışlarını ve kullanıcıların bu davranışlarla etkileşimini belirler, ancak içsel implementasyon (uygulama) detaylarını gizler.

Doğrusal veri türlerinden dizi (array), aynı türdeki verilerin bir araya getirilmesi için kullanılan bir veri yapısıdır. Temel olarak bir dizi ___(c)___ boyutta, ardışık bellek alanında depolanan ve benzer türdeki verileri tutan bir koleksiyondur. Dizilerin en büyük avantajlarından biri, ___(d)___ erişim zamanına sahip olmalarıdır. Bu, dizilerin elemanlarına belirli bir indeks kullanarak doğrudan erişilebileceği anlamına gelir. Dolayısıyla, dizilerdeki herhangi bir elemana erişmek için, elemanın indeksini bilmek yeterlidir.

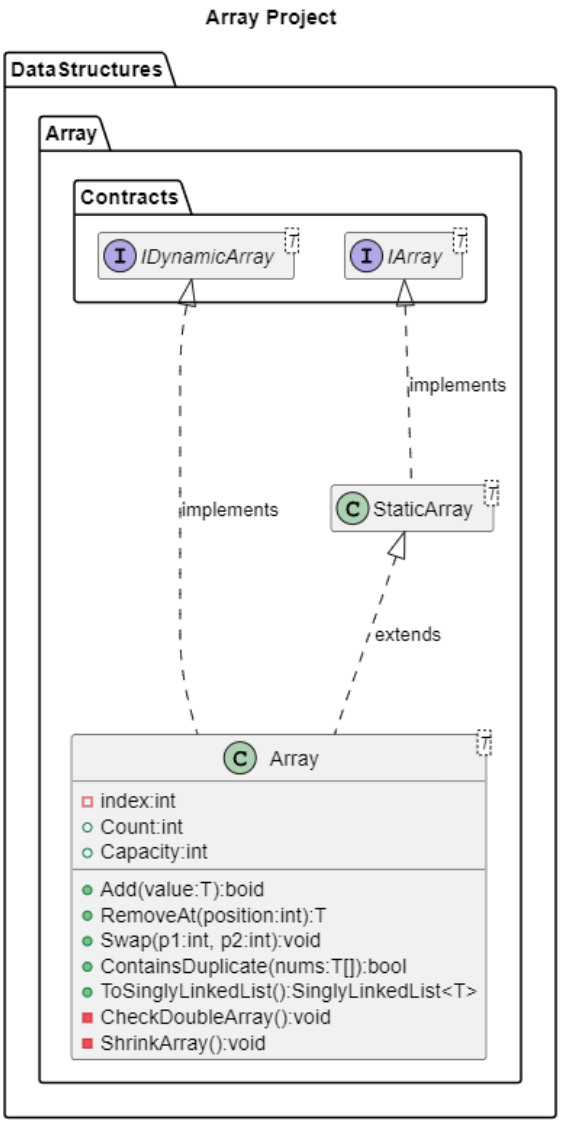
Bağlı liste, verileri bağlı düğümler veya düğüm adı verilen veri yapıları aracılığıyla depolayan bir veri yapısıdır. Her düğüm, verinin yanı sıra bir sonraki düğümün referansını da içerir. Bağlı listeler, veri eklemek veya çıkarmak için ___(e)___ boyuta sahiptir. Bağlı listede saklı olan elemanlara erişmek üzere koleksiyon giriş noktası ___(f)___ olarak ifade edilir. Bağlı listede arama yapmanın maliyeti ___(g)___'dir. Ayrıca, bağlı listeler, her düğümün referansını depolamak için ___(h)___ kullanır. Bu, dizilere göre daha fazla bellek tüketimine neden olabilir.

Yığın (stack), verilerin, ___(i)___ prensibiyle çalışan bir veri yapısıdır. Yığın, verilerin üst üste konulduğu ve yalnızca en üstteki öğenin erişilebilir olduğu bir yapıdır. Yığına yeni bir öğe eklenirken bu öğe, yığının en üstüne yerleştirilir ve yığından bir öğe çıkarıldığında ise en üstteki öğe çıkarılır. Yığındaki en üste yer alan ve son eklenen elemanı ne olduğunu öğrenmek üzere ___(j)___ işlevi kullanılır.

2. "`Contains duplicate`" problemi, verilen bir dizide veya koleksiyonda herhangi bir öğenin birden fazla kez bulunup bulunmadığını belirlemeyi amaçlar. Örneğin, bir dizi içinde aynı değere sahip iki veya daha fazla öğe varsa, bu dizi "`contains duplicate`" olarak kabul edilir.

   Bu problem genellikle bir programlama görevi olarak karşımıza çıkar ve verilerin benzersiz olup olmadığını kontrol etmemiz gereken birçok senaryoda önemli bir rol oynar. Örneğin, bir koleksiyon içinde benzersiz bir liste oluşturmak veya bir veri tabanında yinelenen kayıtları bulmak gibi durumlarda bu problemle karşılaşabiliriz.

   Bu problemi çözmek için genellikle verilen veri yapısını (örneğin, bir dizi veya bir liste) tararız ve her öğeyi bir kümeye ekleriz. Eğer bir öğeyi eklerken o öğe zaten kümeye dahil ise, bu durumda yinelenen bir öğe olduğunu anlarız ve işlemi sonlandırırız. Bu şekilde, yinelenen öğeleri tespit edebiliriz.



   UML diyagramını dikkate alarak ilgili metodu uygun olan sınıfa ekleyiniz.

Örnekler

Örnek-1

Giriş: nums : [1,2,3,1]

Çıkış: true

Örnek-2

Giriş: nums : [1,2,3,4]

Çıkış: false

Buna göre sadece, aşağıdaki kod bloğunun gövdesini cevap kağıdınıza geçiriniz.

```
1  public bool ContainsDuplicate(T[] nums)
2      {
3          throw new NotImplementedException();
4      }
```

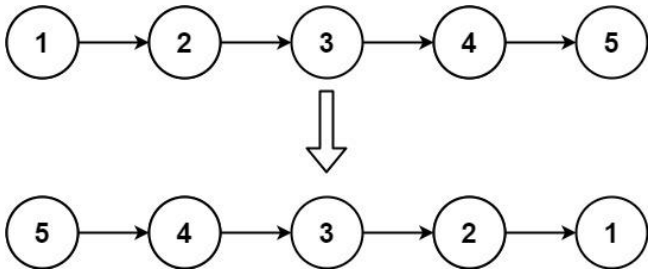
3. "**`Reverse Linked List`**" problemi, verilen bir bağlı listedeki düğümlerin sıralamasını tersine çevirmeyi amaçlar. Bu problem, bir bağlı listeyi tersten yazmak veya ters yönde gezinmek gibi senaryolar için önemlidir.

Bağlı listede düğümler sıralı bir şekilde birbirine bağlıdır ve genellikle her düğümün bir sonraki düğümü gösteren bir "next" referansı vardır. "Reverse Linked List" problemi, bu "next" referanslarını kullanarak bağlı listedeki düğümlerin sıralamasını tersine çevirmeyi gerektirir.

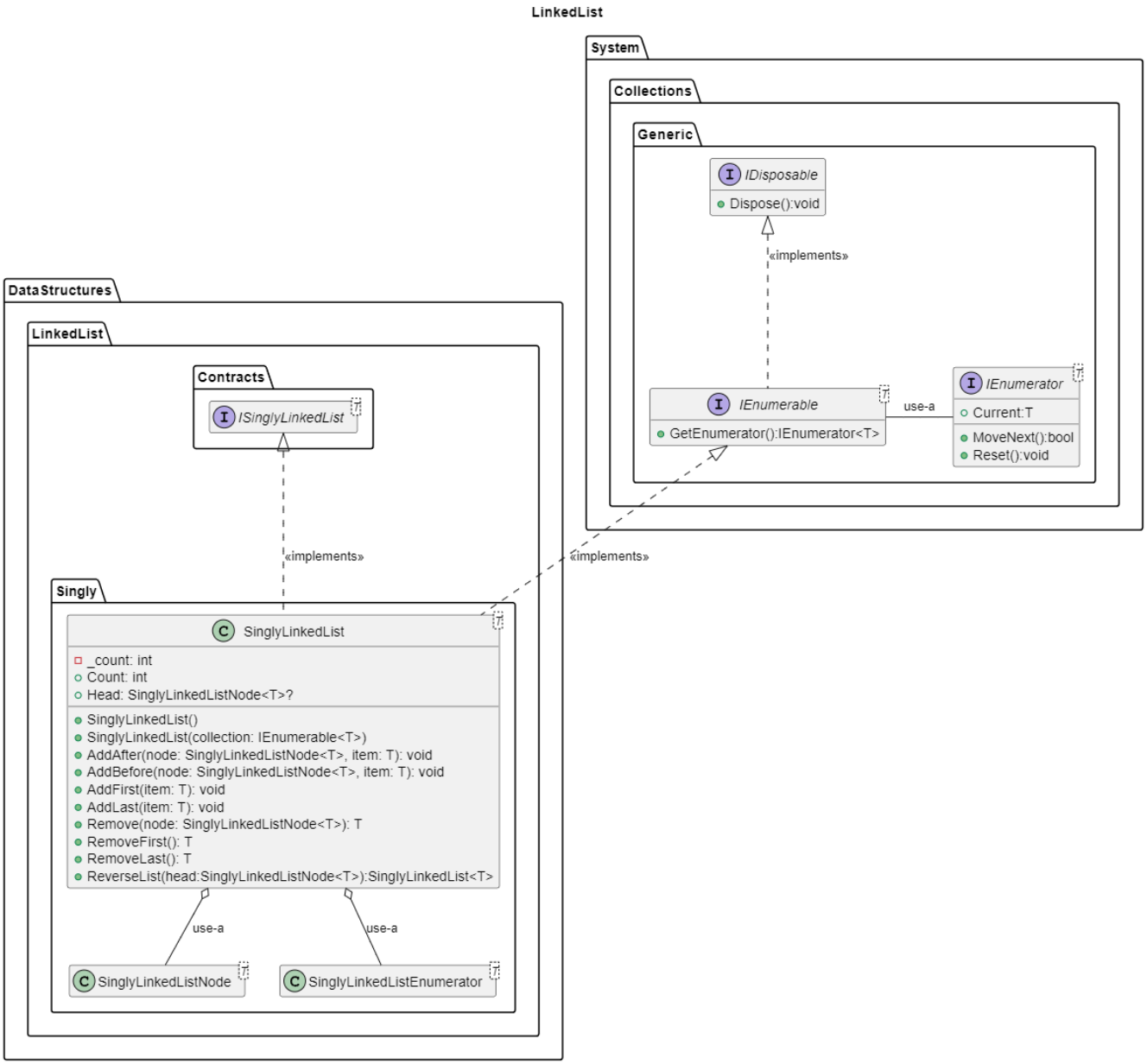
Bu problemi çözmek için genellikle iki düğümlü bir yaklaşım kullanılır. Her adımda, bir düğümün "next" referansını bir önceki düğüme doğru çevirirız. Bu şekilde, her düğümü takip eden düğüm bir önceki düğüm olur ve böylece bağlı liste tersine çevrilir.

Problem çözüldüğünde, bağlı listenin ilk düğümü artık eski son düğüm olur ve bağlı listenin son düğümü eski ilk düğüm olur. Bu şekilde, bağlı listenin başlangıç ve bitiş noktaları değişir ve bağlı liste tersine çevrilmiş olur.

Örnek



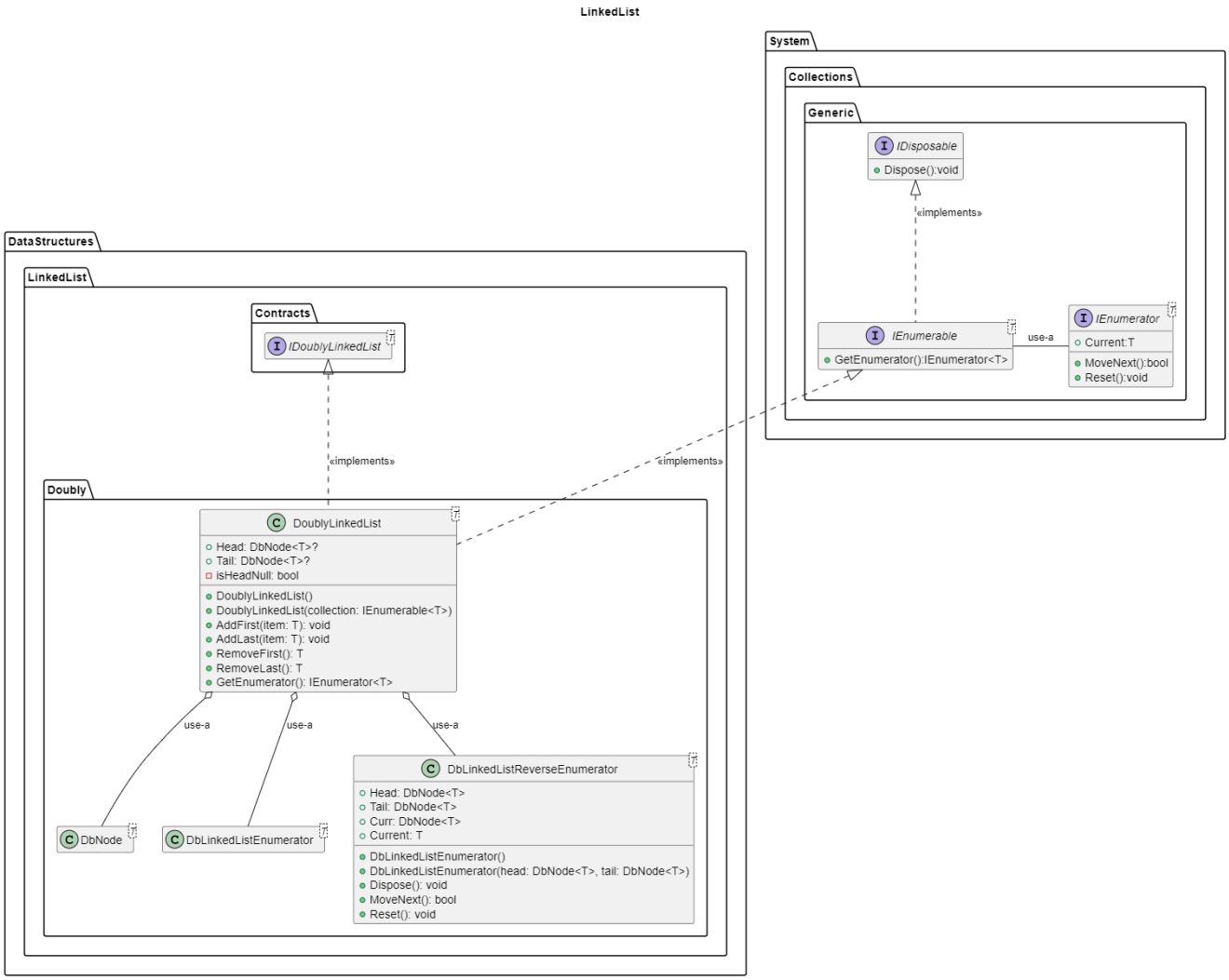
Şekilde görüldüğü gibi ilgili fonksiyonun çalışması sonrası bağlı liste tersine çevrilmiştir. UML diyagramını dikkate alarak ilgili fonksiyonu uygun olan sınıfa ekleyiniz.



Buna sadece aşağıdaki kod bloğunu cevap kağıdınıza geçiriniz.

```
public SinglyLinkedList<T> ReverseList(SinglyLinkedListNode<T> head)
    {
        throw new NotImplementedException();
    }
```

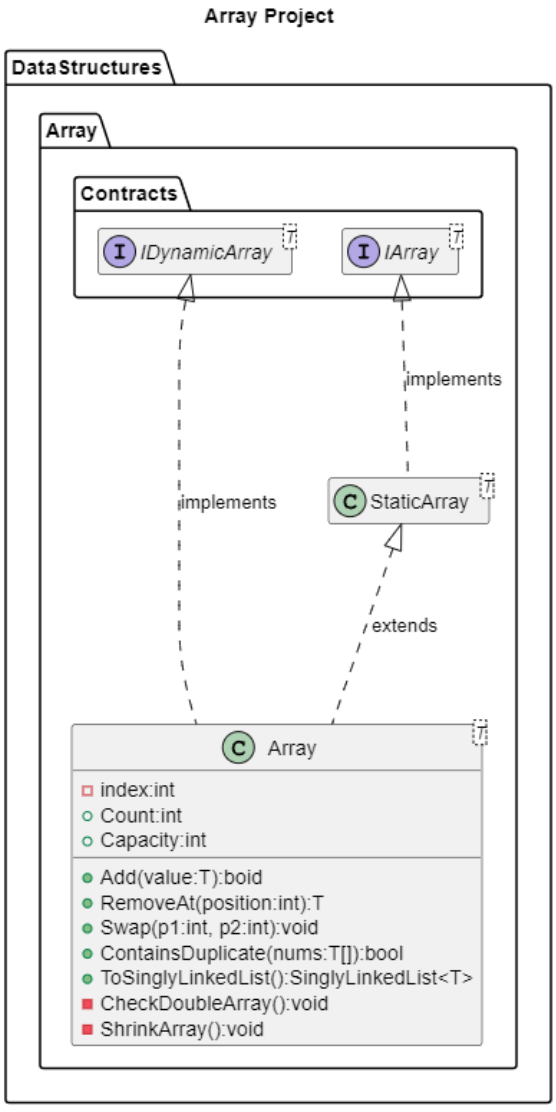
4. Çift yönlü bağlı listede (`DoublyLinkedList<T>`) liste sonundan liste başına doğru gezinme sağlayan `DbLinkedListReverseEnumerator` sınıfının tasarımı gerçekleştiriniz. Bir başka ifadeyle bu bağlı liste (koleksiyon) üzerinde `foreach` döngüsünün çalıştırılması durumunda; liste sonundaki elemandan (`Tail`), liste başına doğru (`Head`) elemanların gezinmesi beklenir.



Sadece `MoveNext()` metodunun içeriğini cevap kağıdına geçirmeniz beklenmektedir.

```
public bool MoveNext()
        {
            throw new NotImplementedException();
        }
```

5. Bir diziyi bağlı listeye çeviren metot tasarımı gerçekleştiriniz. UML diyagramında metodun yazılması gereken sınıf verilmiştir.



Sadece aşağıda verilen bölümün cevap kâğıdına geçirilmesi beklenmektedir.

```
1  public SinglyLinkedList<T> ToSinglyLinkedList()
2      {
3          throw new NotImplementedException();
4      }
```