

Gebze Technical University  
Computer Engineering

CSE 222  
2017 Spring

HOMEWORK 03 REPORT

Elif Şeyma ARMAĞAN  
151044042

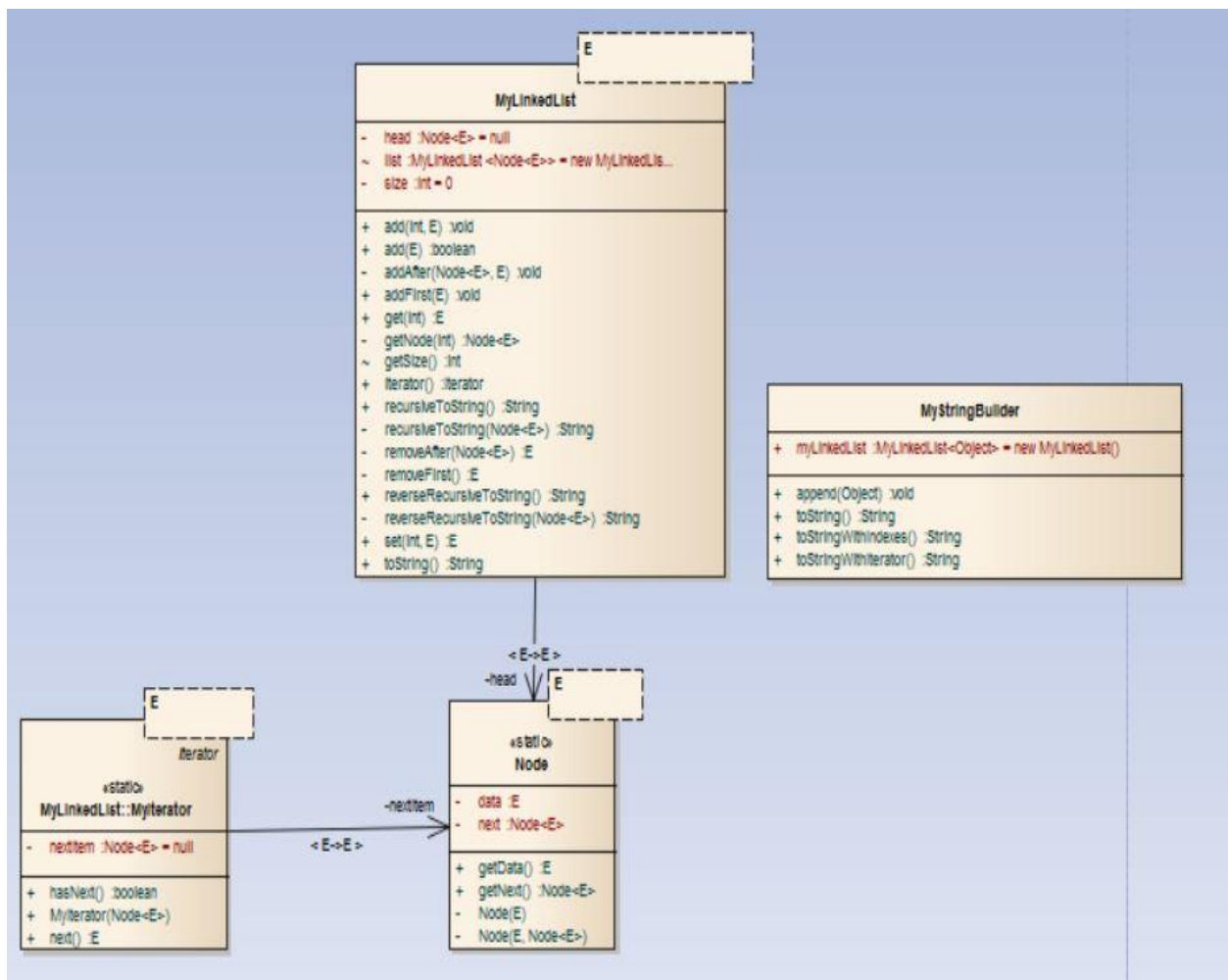
Course Assistant:  
Nur Banu ALBAYRAK

# Q1)

## 1. System Requirements

Kodun çalışabilmesi için içinde integerlar olan bir numbers.txt dosyası gereklidir.

## 2. Class Diagrams



### 3. Problem Solutions Approach

İlk başta MyStringBuilder class ımın kullanabilmesi için MyLinkedList class ımı yazdım. MyLinkedList class ı Java'da tanımlı olan LinkedList class ı ile aynı metodlara ve aynı işleve sahip. MyLinkedList class ımı yazdıktan sonra toString metodunu override edebilmek için istenen 3 yöntemden ilki olan indeksleri ve get metodunu kullanarak yapabilmek için MyLinkedList class ımın get fonksiyonunu kullanarak teker teker her elemana erişip onları sıra ile birbirlerine ekledim ve sonuçta bunu return ettim

2. metod olan iteratör yardımıyla yapma kısmında ise kendi yazmış olduğum MyLinkedList class ımda yazdığım iteratör ü kullandım ve iteratör yardımıyla node larım üzerinde dolaştım ve bunları birbirlerine ekleyerek oluşturdum.

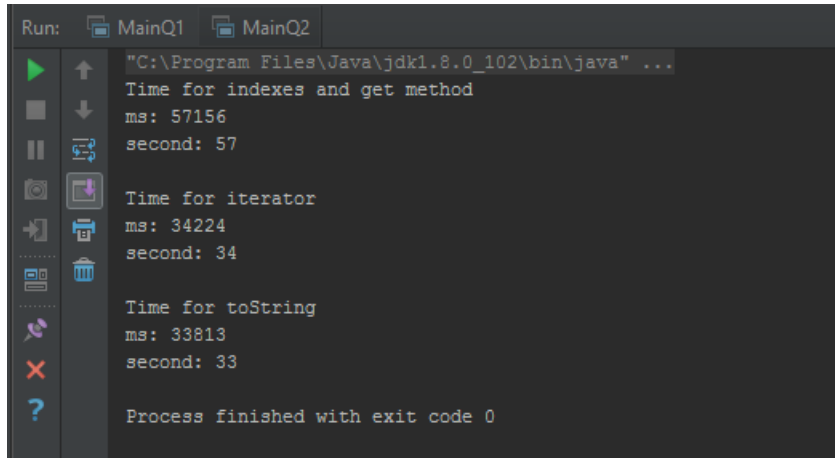
3. metod olan toString metodunun override edildiği halinde ise yazmış olduğum MyLinkedList class ında toString metodunu override ettim. Override etmiş olduğum toString metodu her bir node un datasının toString ini çağırıyor ve bunları birbirlerine ekleyip nodelar üzerinde ilerleyerek string oluşturmakta.

### 4. Test Cases

Yazmış olduğum toString metodlarını test edebilmek için öncelikle ödev pdf inde istendiği üzere 100.000 integer içeren bir dosya üzerindeki verileri okuyup bunları yazmış olduğum 3 farklı toString metodunu kullanarak farklı dosyalara yazdım ve bunların çalışma sürelerini karşılaştırdım. 3 toString metodu da farklı yöntemler ile çalıştığı için testler sonucu çalışma süreleri birbirinden farklı çıktı. Çalışma sürelerini kıyasladığımda en hızlı sürede tamamlanan birbirleri ile neredeyse aynı sürede çalışan, iteratör kullandığım yöntem ile MyLinkedList class ımın override ettiğim toString metodunu kullandığım yöntem oldu. İkisinin birbirine benzer çıkmasının nedeni ikisinde de head node dan başlanıp listenin sonuna kadar ilerleniyor oluşuydu. Get metodunu ve indeksleri kullandığım yöntemin diğer iki yönteme göre yavaş çalışmasının nedeni ise her seferinde baştan başlayarak bulmam gerekeni aradığım bir döngü ile yapmam oldu.

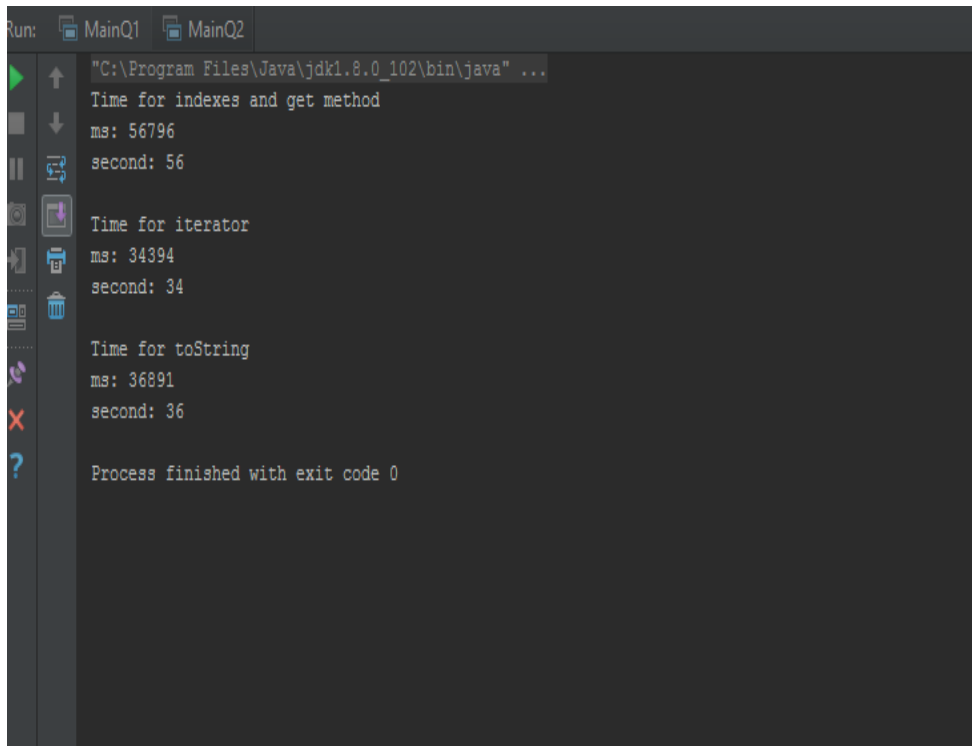
## 5. Running and Result

Kodumu farklı toStringleri kullanarak çalıştırdığımda çalışma süreleri aşağıdaki gibidir.



The screenshot shows an IDE console window with a dark theme. At the top, there are tabs for 'MainQ1' and 'MainQ2'. The console output for 'MainQ1' is as follows:

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...  
Time for indexes and get method  
ms: 57156  
second: 57  
  
Time for iterator  
ms: 34224  
second: 34  
.....  
Time for toString  
ms: 33813  
second: 33  
  
Process finished with exit code 0
```



The screenshot shows an IDE console window with a dark theme. At the top, there are tabs for 'MainQ1' and 'MainQ2'. The console output for 'MainQ2' is as follows:

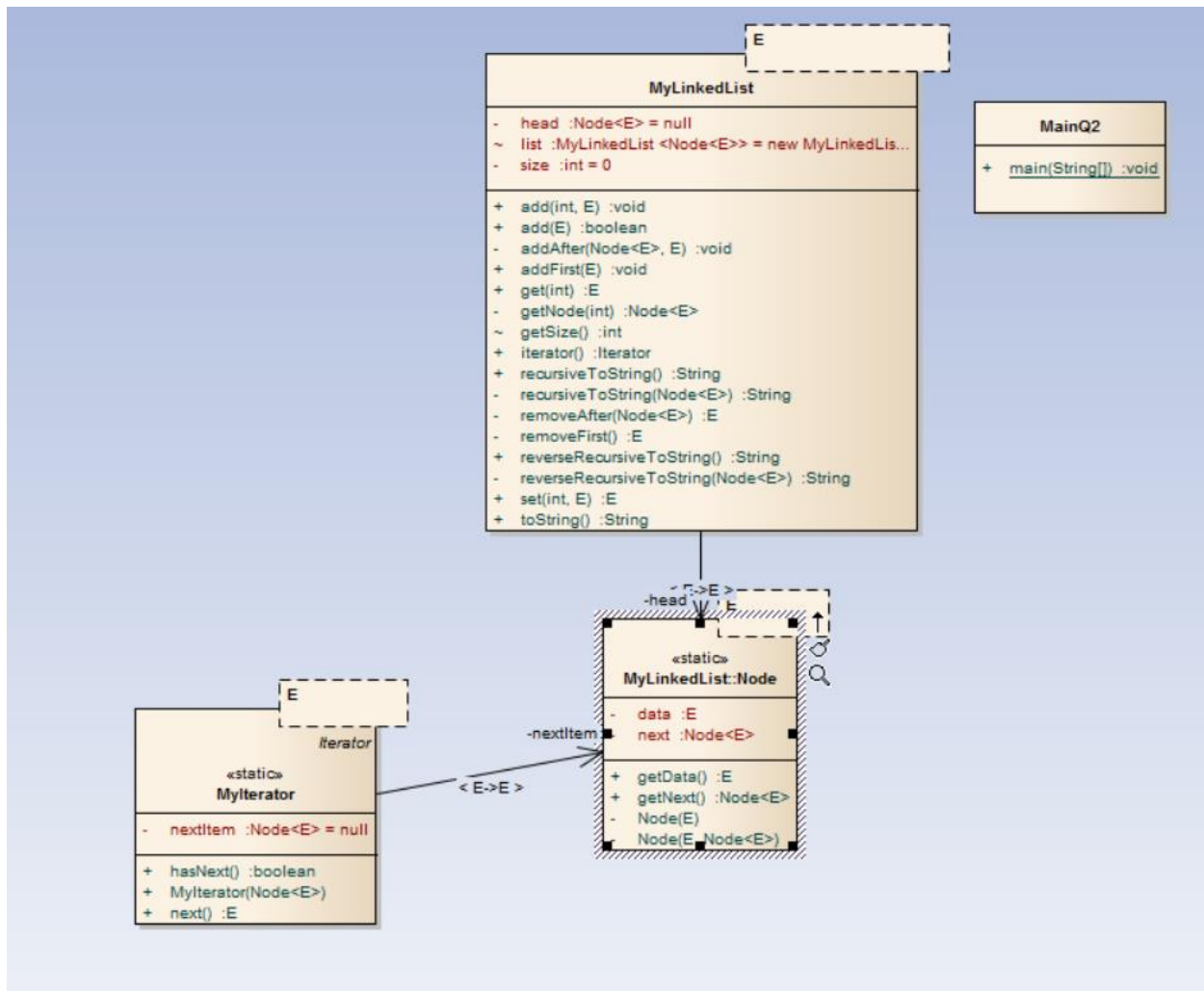
```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...  
Time for indexes and get method  
ms: 56796  
second: 56  
  
Time for iterator  
ms: 34394  
second: 34  
.....  
Time for toString  
ms: 36891  
second: 36  
  
Process finished with exit code 0
```

## Q2)

### 1. System Requirements

Kodun çalışabilmesi için içinde elemanlar olan bir list olması gerekir.

### 2. Class Diagrams



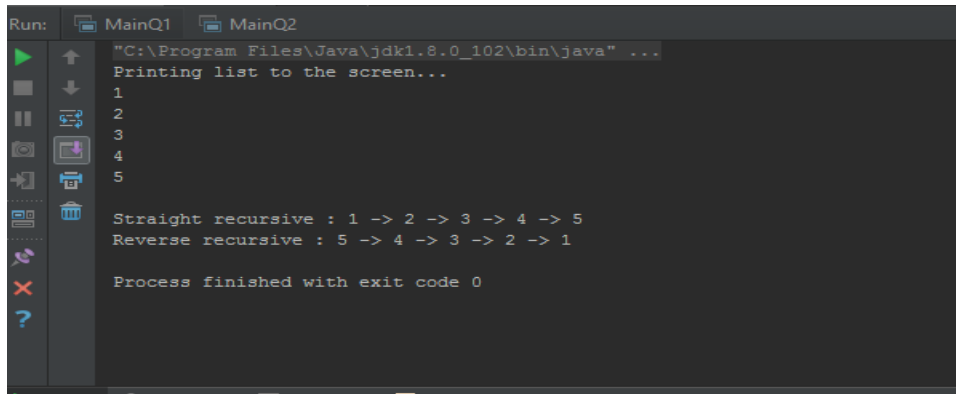
### 3. Problem Solutions Approach

Headden başlanarak null bulana kadar recursive ilerlenir. Daha sonra geri dönerken nodeların toString lerı toplanır.

## 4. Test Cases

Kodumu test etmek için MainQ2 isimli bir main yazdım. Bu main de öncelikle bir list oluşturdum ve bu list e elemanlar ekledim. Daha sonra bu elemanları ilk başta sıralı bir şekilde ekrana bastım daha sonra reverseToString metodu ile çağırıp reversed hallerini ekrana bastım.

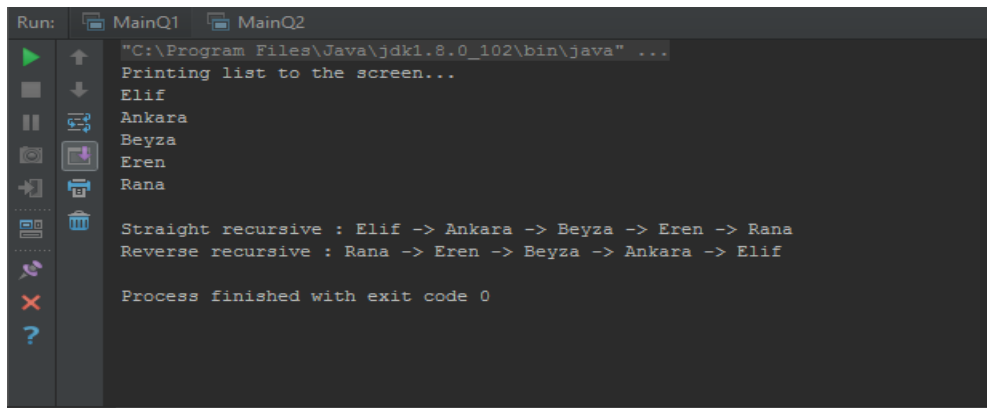
## 5. Running and Result



```
Run: MainQ1 MainQ2
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...
Printing list to the screen...
1
2
3
4
5

Straight recursive : 1 -> 2 -> 3 -> 4 -> 5
Reverse recursive : 5 -> 4 -> 3 -> 2 -> 1

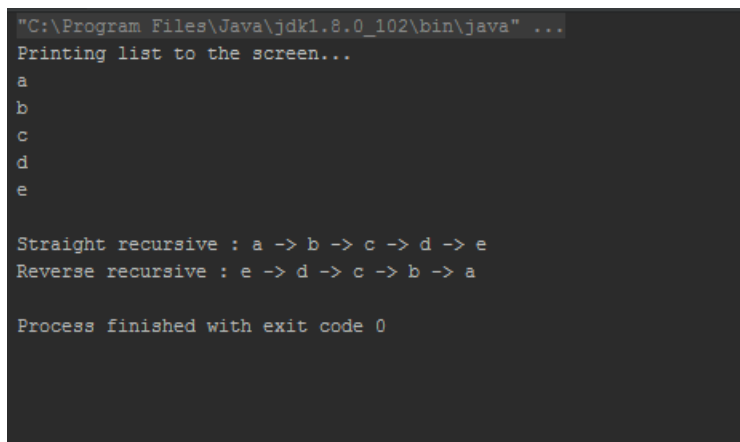
Process finished with exit code 0
```



```
Run: MainQ1 MainQ2
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...
Printing list to the screen...
Elif
Ankara
Beyza
Eren
Rana

Straight recursive : Elif -> Ankara -> Beyza -> Eren -> Rana
Reverse recursive : Rana -> Eren -> Beyza -> Ankara -> Elif

Process finished with exit code 0
```



```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...
Printing list to the screen...
a
b
c
d
e

Straight recursive : a -> b -> c -> d -> e
Reverse recursive : e -> d -> c -> b -> a

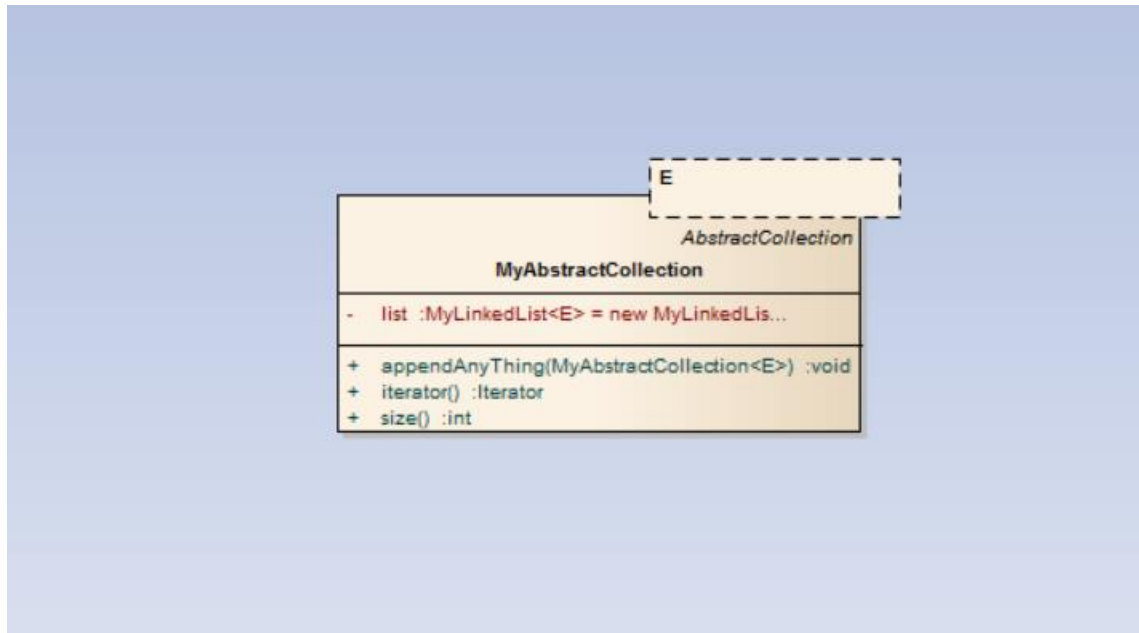
Process finished with exit code 0
```

## Q3)

### 1. System Requirements

Kodun çalışabilmesi için eklenebilecek bir abstract collection objesi lazımdır.

### 2. Class Diagrams



### 3. Problem Solutions Approach

`appendAnything` metodu aldığı objenin iteratörü yardımıyla tüm objeyi dolaşıp elemanları `add` metodu ile `MyAbstraction` a ekler. Burdaki `add` metodu `AbstractCollection` dan gelmektedir. `MyAbstractCollection` ın instance ı oluşturulamaz çünkü abstract tır.

### 4. Test Cases && Running and Result

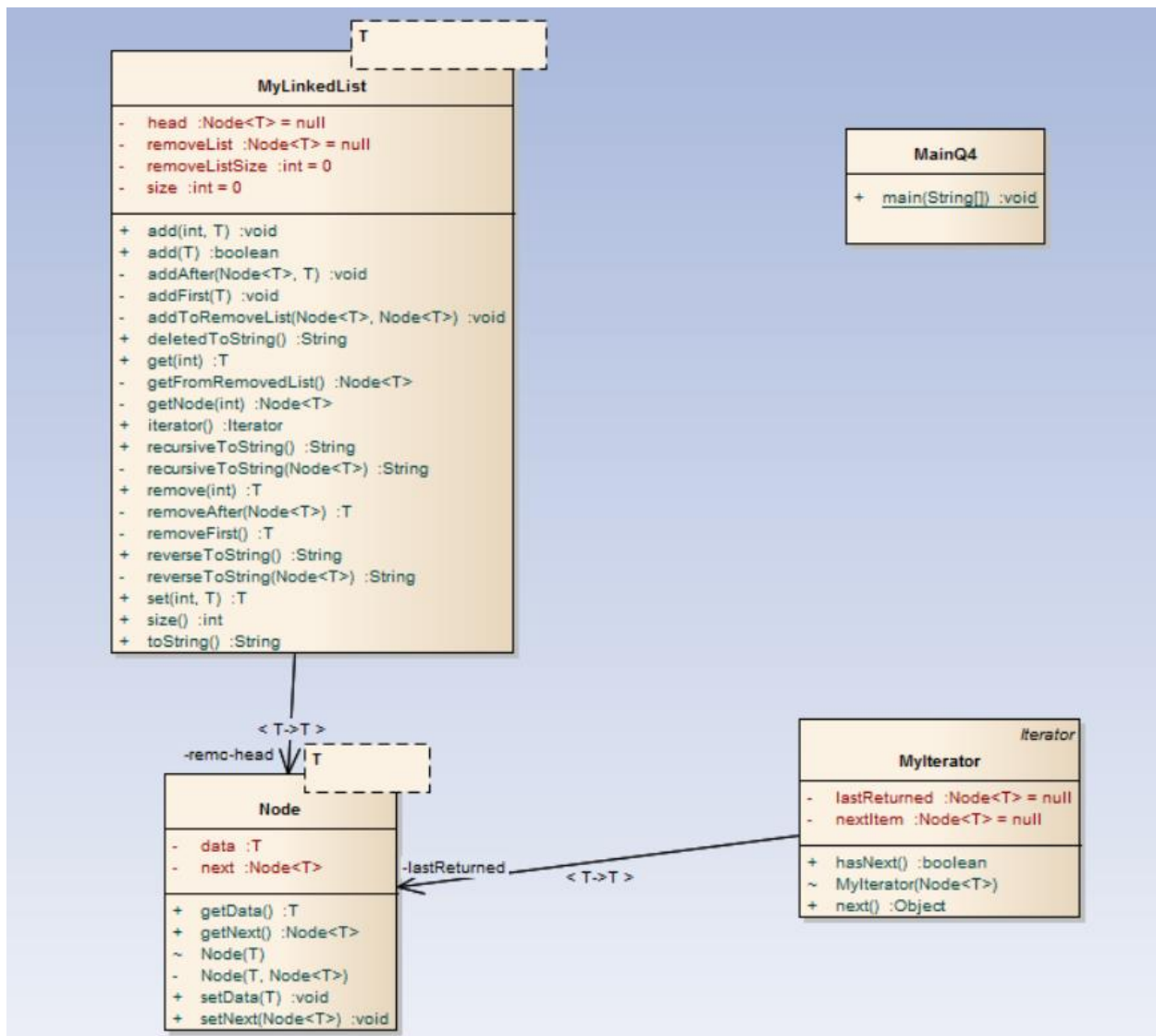
Ödev pdf inde yapılması istenmediği için bu kısımlar yapılmamıştır.

## Q4)

### 1. System Requirements

Kodun çalışabilmesi için içinde elemanlar olan bir test list i olmalıdır.

### 2. Class Diagrams





### 3. Problem Solutions Approach

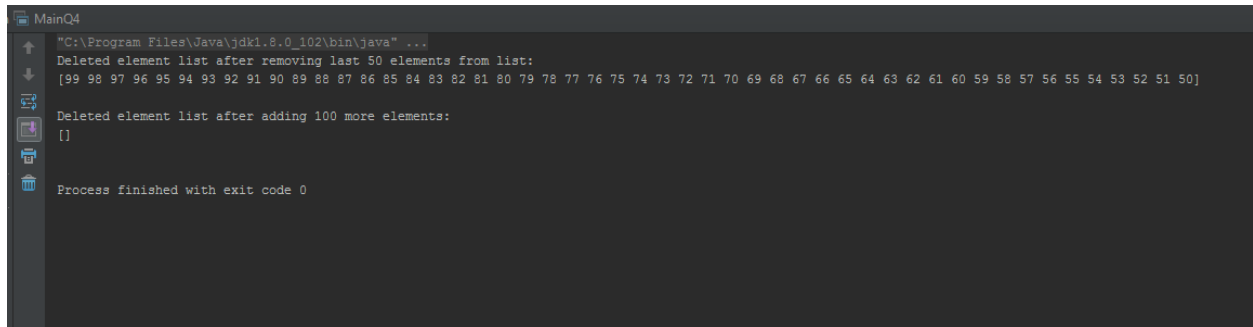
Silinen elemanların bulunduğu Node ları tutmak için tıpkı head node ile elemanları tuttuğum gibi bir remove head node u oluşturup bunun next ine ataya ataya sakladım. Yani yine node ları birbirine bağlayarak sakladım, farklı bir veri yapısı ile çalışmadım. AddRemoveList metodu ile recursive olarak silinen Nodeları removeList e ekledim.

Daha sonra kullanıcı yeni bir eleman eklemek istediğinde getFromRemoveList metodum ile removeList te eğer Node varsa oradan bir Node alıp buna kullanıcının verdiği datayı atayıp kullandım. Eğer yoksa new ile yeni Node oluşturdum.

### 4. Test Cases

Kodumu test etmek için öncelikle 100 tane sayıyı bir listeye ekledim. Daha sonra bu listeden 50 tane sayıyı sildim. Bu sildiğim 50 sayı SingleLinkedList objemin içinde removeList de saklandı. Daha sonra 100 tane daha sayı eklediğimde önce bu 50 tane silinmiş olan node kullanıldı. Bunu göstermek için 50 tane sildikten ve daha sonra 100 tane ekledikten sonra silinmişleri ekrana bastırdım.

### 5. Running and Result



```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...  
Deleted element list after removing last 50 elements from list:  
[99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50]  
Deleted element list after adding 100 more elements:  
[]  
Process finished with exit code 0
```