

Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK III REPORT

STUDENT NAME
LÜTFULLAH TÜRKER

STUDENT NUMBER
141044050

Course Assistant:
Nur Banu Albayrak

1. Detailed system requirements

2. Class diagrams

3. Problem Solutions Approach

4. Test Cases

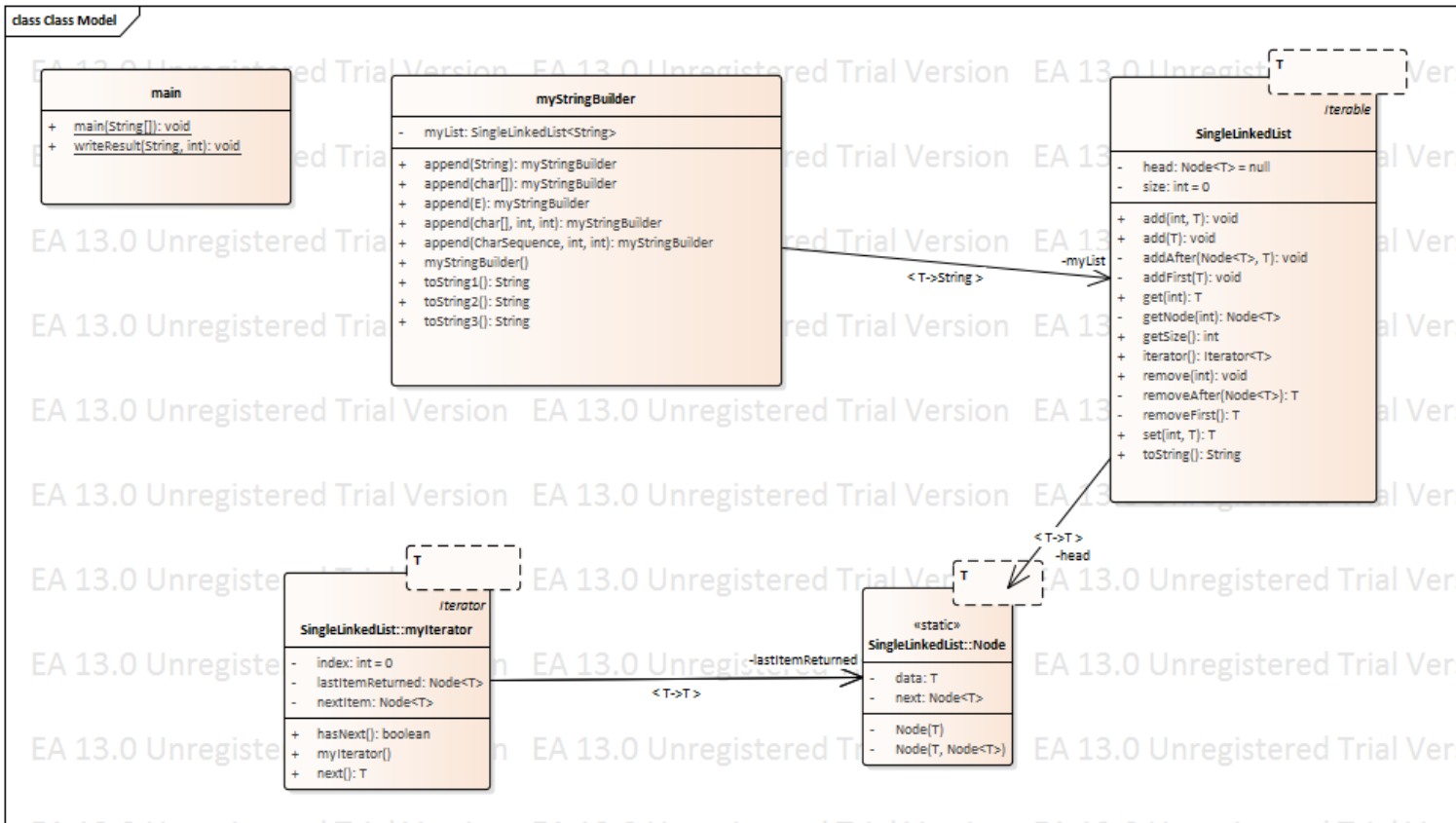
5. Running and Results

1. Detailed system requirements

Sistemde Java yüklü olmalı

NetBeans proje klasörleri IntelliJ projesi olarak da IntelliJde açılabilir.

2. Class diagrams



Diyagram Enterprise Architect ile çizildi.

3. Problem Solutions Approach

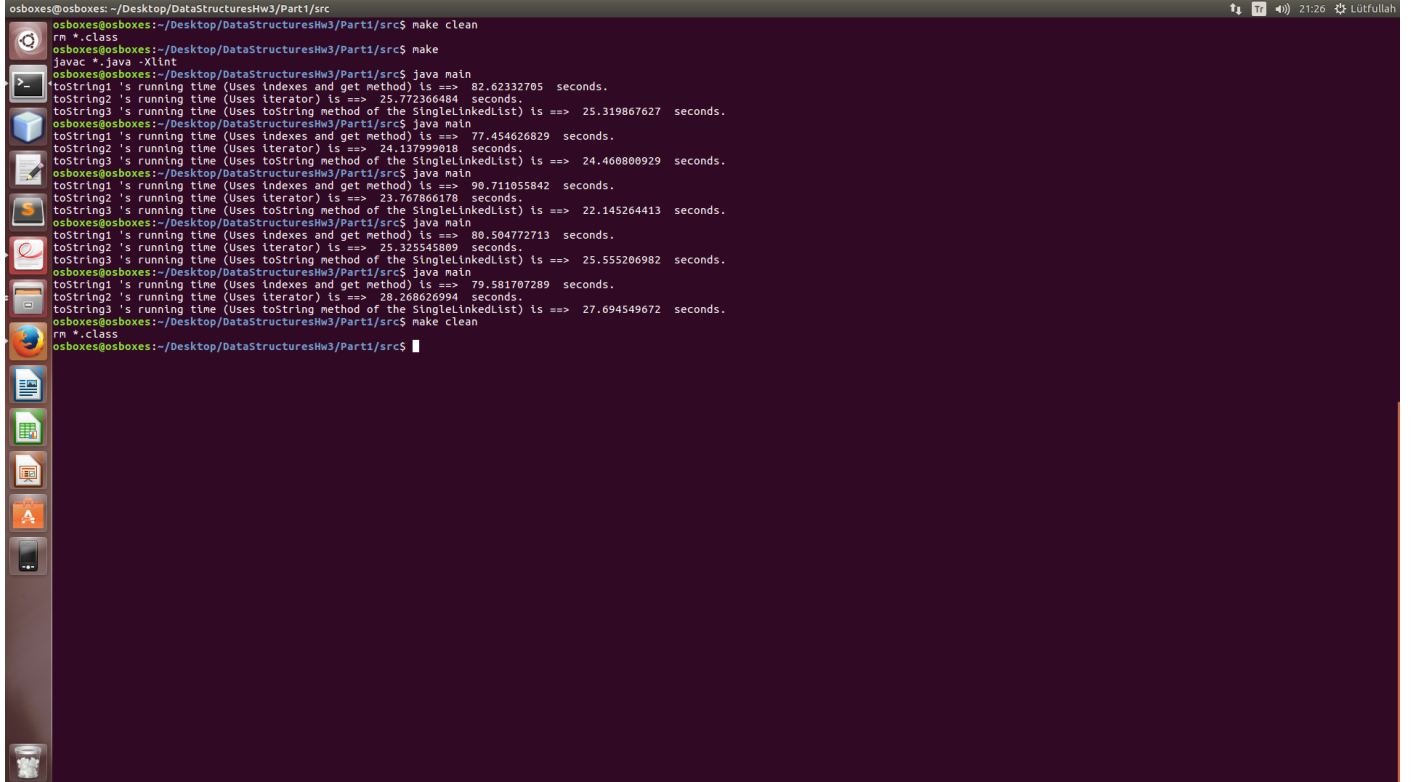
1. Soruya yaklaşımım, orijinal class'ın benzerini yazacağımız için Java'nın orijinal class larının içeriğini görebildiğim bir siteden orijinal StringBuilder class ına ve gerekli fonksiyonlara baktım. Bu fonksiyonlarla aynı parametreleri alan fonksiyonları orijinal hallerinden de yardım alarak yazdım. myStringBuilder içindeki SingleLinkedList sınıfını da kitabımızdaki örnekten yardım alarak yazdım. İteratörlü toString yazabilmek için SingleLinkedList class ının içinde kendi Iterator sınıfımı yazdım.
2. Soruya yaklaşımım, ilk başta açık bir şey belirtilmediği için toStringden gelen Stringi tamamen recursive ile ters çevirmiştim. Daha sonra elemanları ter sırayla basılacak şeklinde belirtilince değiştirerek ters sıralı şekilde tekrar yazdım. Test için mainde Stringler ekleyerek onların ters ve düz hallerini sırayla bastım ve birbirinin ters sıralı olduğunu doğruladım.
3. Soruda ise AbstractCollection sınıfının kendi addAll isimli fonksiyonu olduğunu gördüm ve bunun ödevde bizden istenen görevi yaptığını anladım fakat kendimizin yazmamızı istediğinizi düşündüm. Orijinal addAll fonksiyonundan da yararlanarak addAll fonksiyonunu yazdım.
4. Soruya yaklaşımım, Silinenleri de tutmamız istendiği için silinenleri depolamak için de bir SingleLinkedList oluşturdum. İç içe aynı classdan oluşturduğum için sonsuz döngüde kendini oluşturmasını önlemek için farklı constructorlarda oluşturdum deleted LinkedList ini. Bu sayede bir linkedList oluşturulduğunda kendisi ve içindeki deleted LinkedList i oluyor. add ve remove fonksiyonlarının her birinde deleted isimli LinkedList im için if'ler koymak zorunda kaldım bunu yapabilmek için. Eğer önceden silinmiş eleman varsa deleted 'dan bir elemanı kullan yeni eleman oluşturmak için dedim. Ve eğer silinen eleman listemin Size 'ı 0 sa yeni eleman oluştur şeklinde if'ler ile yönetimi sağladım.

4. Test Cases

Senaryo	Test	Beklenen Sonuç	Gerçek Sonuç
Program İteratör ile ayrı index ler ile ayrı çalışan ama aynı çıktıyı veren 2 farklı toString fonksiyonuna sahiptir.	Aynı sonucu veren iki toString fonksiyonunu da çağırdığımızda hangisi daha hızlı çalışır.	İterator kullanan daha hızlı çalışması beklenir.	İterator kullanan toString metodu olması gerektiği gibi çok daha hızlı çalışır.Çünkü indexle çalışan $O(n^2)$ ile çalışır.iteratörle ise $O(n)$.

5. Running and Results

1.Soru için :



```
osboxes@osboxes: ~/Desktop/DataStructuresHw3/Part1/src$ rm *.class
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ javac *.java -Xlint
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ java main
toString1 's running time (Uses indexes and get method) is ==> 82.62332705 seconds.
toString2 's running time (Uses iterator) is ==> 25.772366484 seconds.
toString3 's running time (Uses toString method of the SingleLinkedList) is ==> 25.319867627 seconds.
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ java main
toString1 's running time (Uses indexes and get method) is ==> 77.454626829 seconds.
toString2 's running time (Uses iterator) is ==> 24.137999018 seconds.
toString3 's running time (Uses toString method of the SingleLinkedList) is ==> 24.468800929 seconds.
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ java main
toString1 's running time (Uses indexes and get method) is ==> 90.711055842 seconds.
toString2 's running time (Uses iterator) is ==> 23.767866178 seconds.
toString3 's running time (Uses toString method of the SingleLinkedList) is ==> 22.145264413 seconds.
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ java main
toString1 's running time (Uses indexes and get method) is ==> 80.504772713 seconds.
toString2 's running time (Uses iterator) is ==> 25.325545889 seconds.
toString3 's running time (Uses toString method of the SingleLinkedList) is ==> 25.555206982 seconds.
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ java main
toString1 's running time (Uses indexes and get method) is ==> 79.581707289 seconds.
toString2 's running time (Uses iterator) is ==> 28.268626994 seconds.
toString3 's running time (Uses toString method of the SingleLinkedList) is ==> 27.694549672 seconds.
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$ rm *.class
osboxes@osboxes:~/Desktop/DataStructuresHw3/Part1/src$
```

Ekran fotoğrafında da görüldüğü gibi İterator ile çalışan toString fonksiyonunun çalışma süresi index ve get fonksiyonları ile çalışan hallerinden açık ara bir fark ile önde.Bunun sebebi iterator ile olan fonksiyonun tek döngüde while (hasNext()) ile n kadar dönmesi yani çalışma süresinin $\theta(n)$ kadar olması ve index ile get fonksiyonunu kullanan hallerinin de while (i<size) derken while 'ın içinde de get fonksiyonu kullanılıyor ve bu get fonksiyonuna bakmadan karar verirse buna da $\theta(n)$ diyebilirdik fakat içerdeki get fonksiyonu da n kadar dönüp elemanı aradığı için indexli toString fonksiyonumuzun çalışma süresi $\theta(n^2)$ olur.Bu yüzden iterator kullanan hali büyük farkla daha hızlı çalışır.linkedList kullanan toString fonksiyonu da iterator gibi while (head.next!=null) şeklinde olduğu için o da iterator kadar hızlı çalışmaktadır.

Yukarıdaki çalışma süreleri 100.000 integer ile çalışmaktadır.

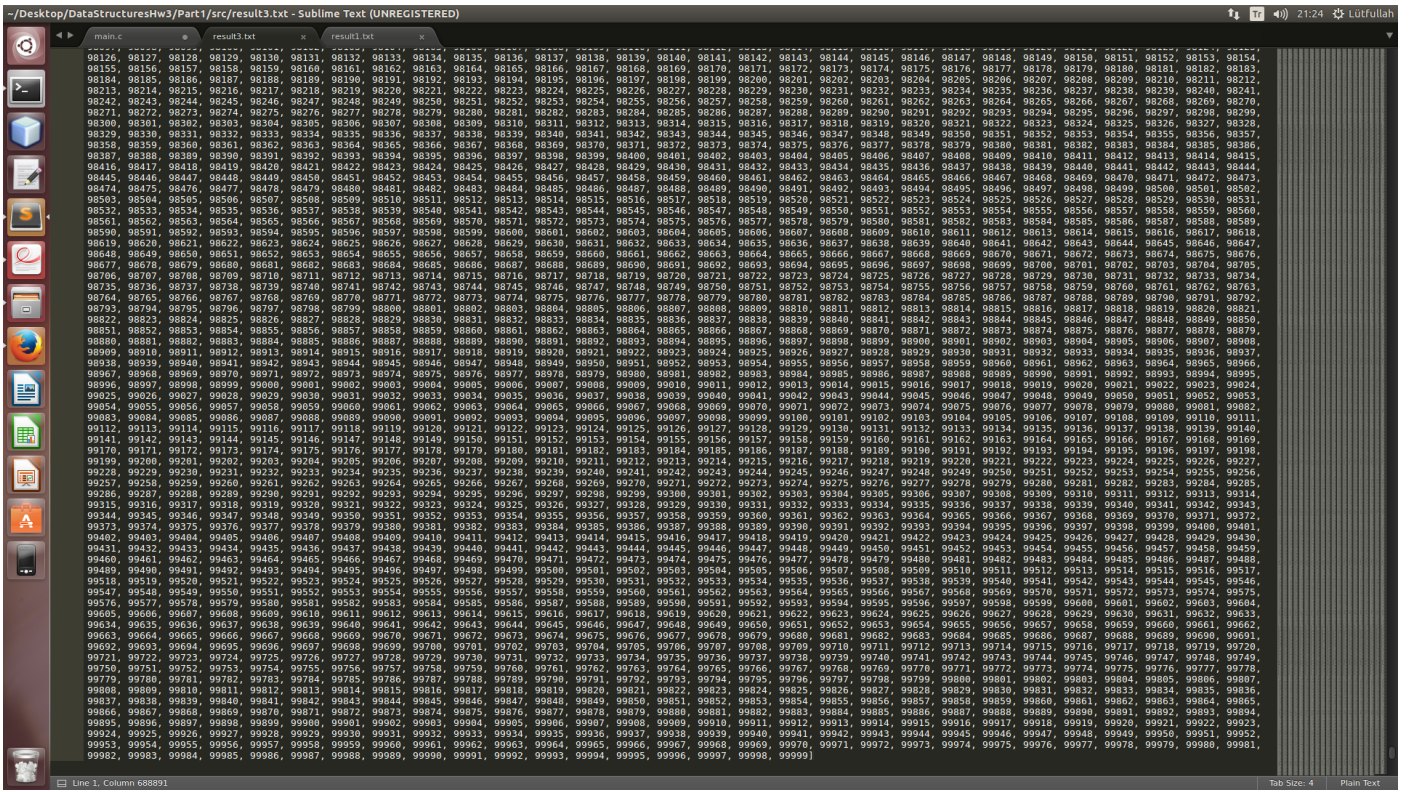
Çalışmanın inputu ve outputları aşağıdaki gibidir.



Yukarıdaki testi yaptığımız numbers.txt dosyası

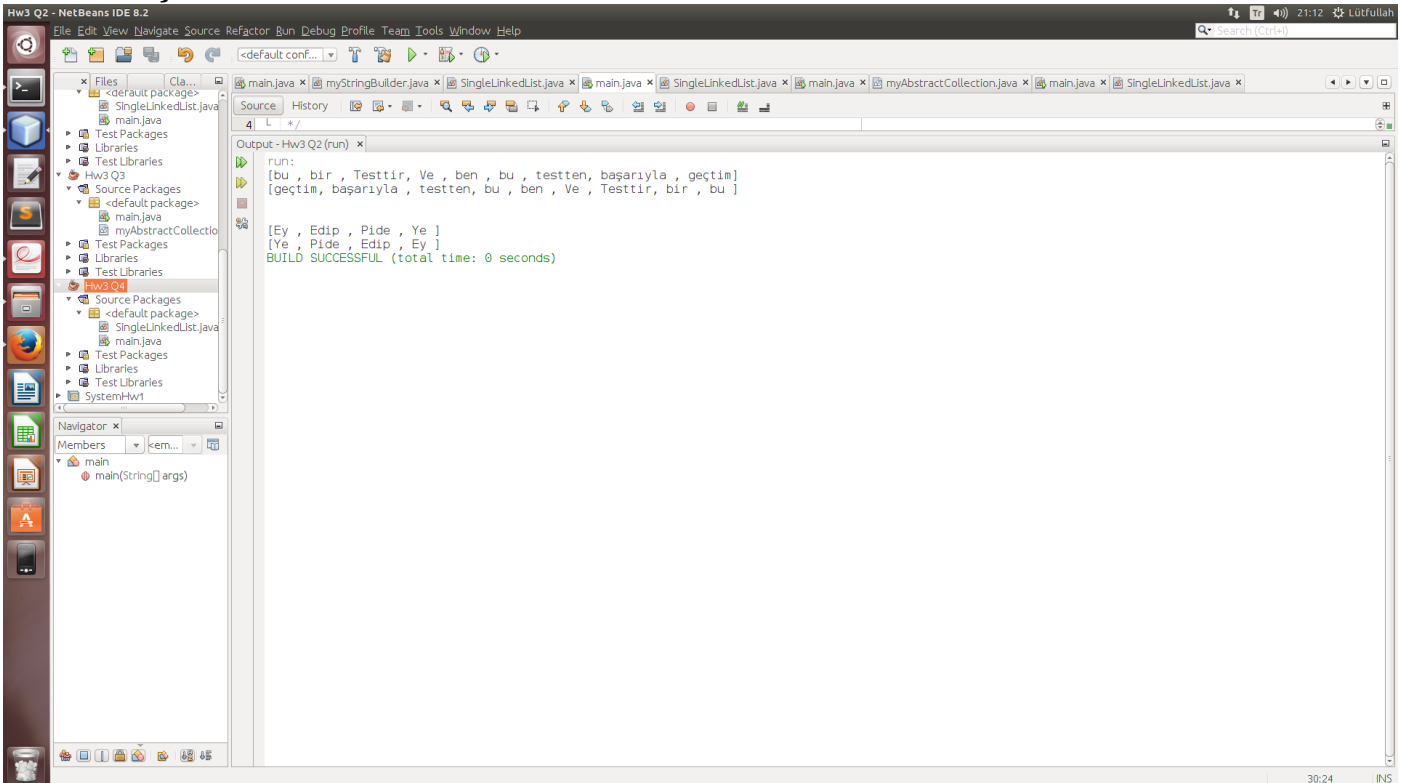


result1.txt dosyası



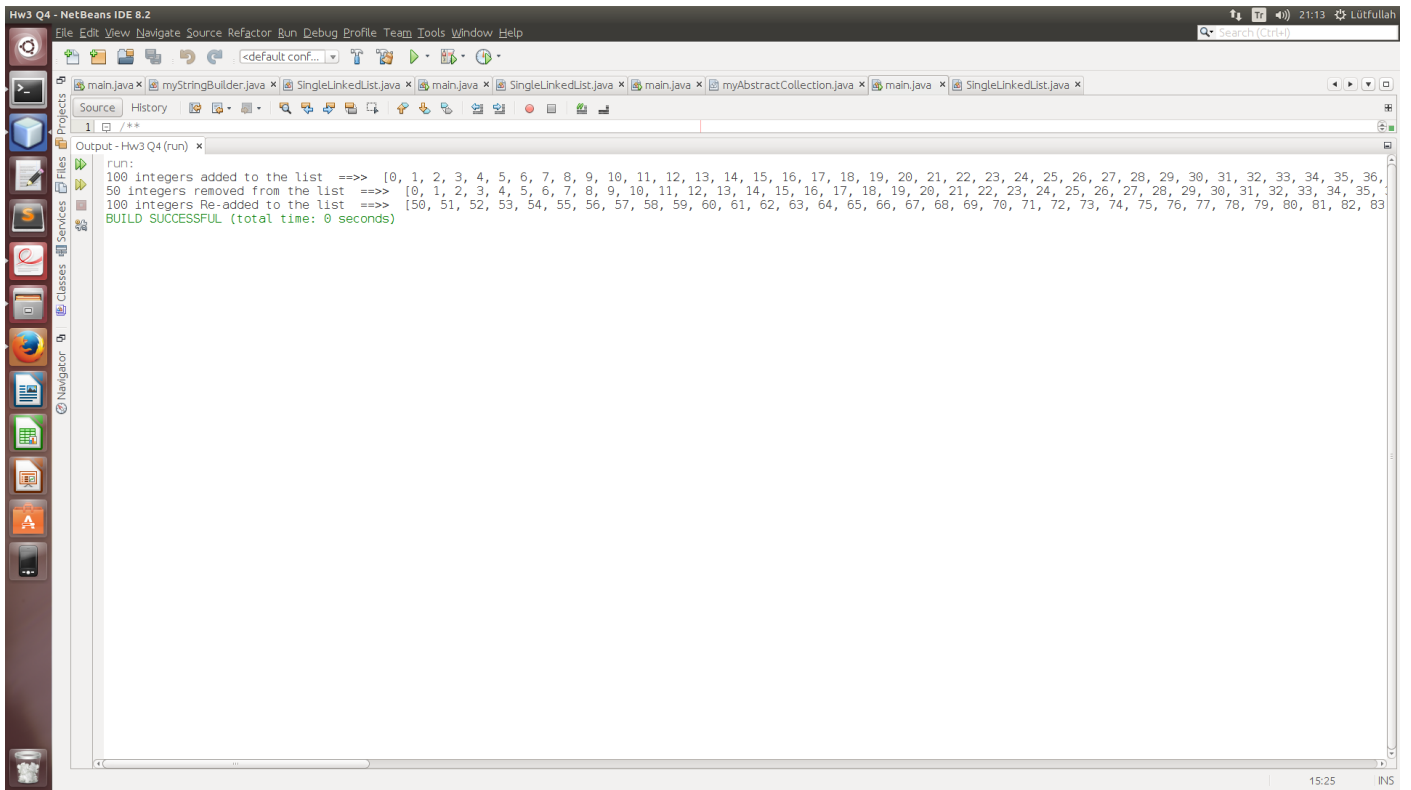
result3.txt dosyası

2.Sorui için :

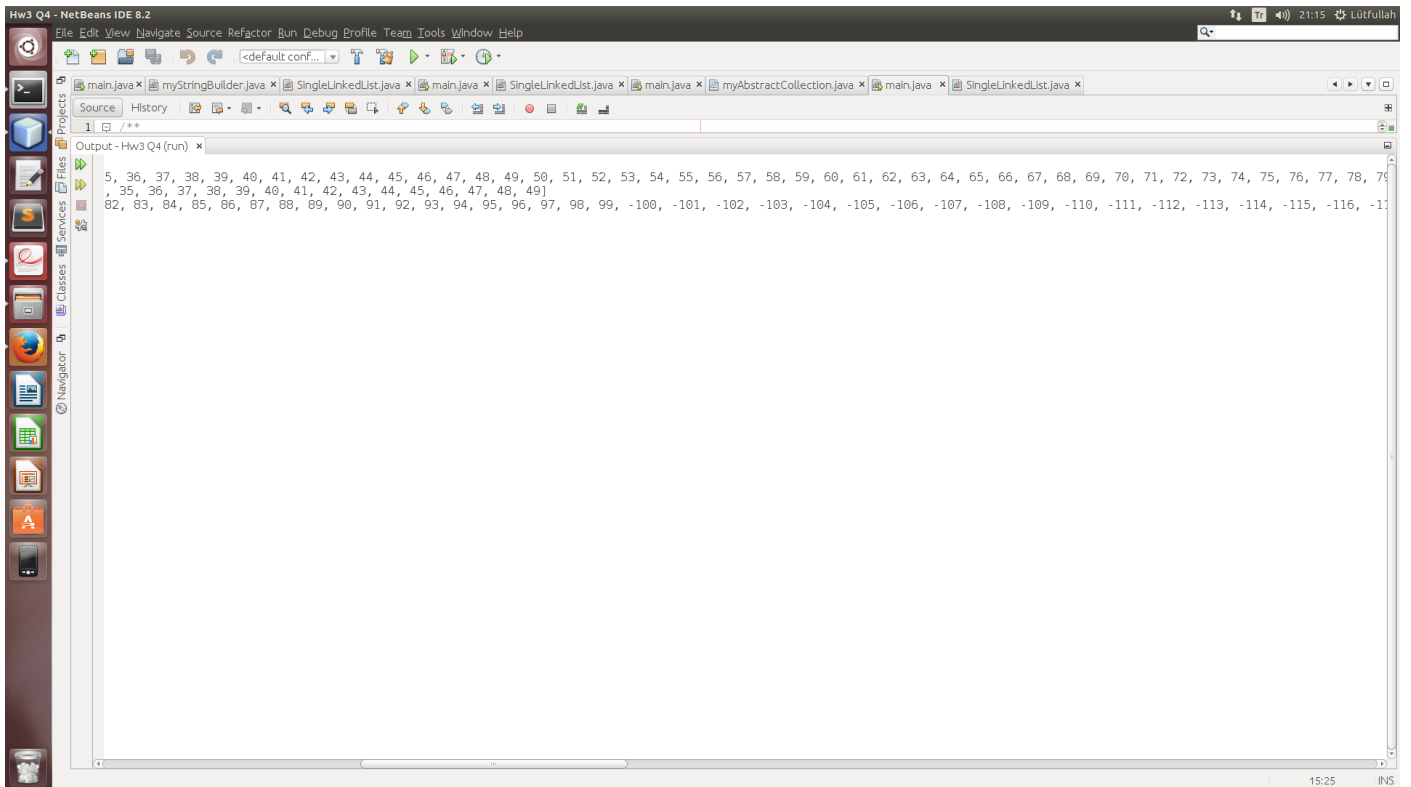


Test sonucu yukarda görüldüğü gibidir.İlk satırlar düz sıralı listeyi ikinci satırlar ters çevrilmiş listeyi göstermektedir.

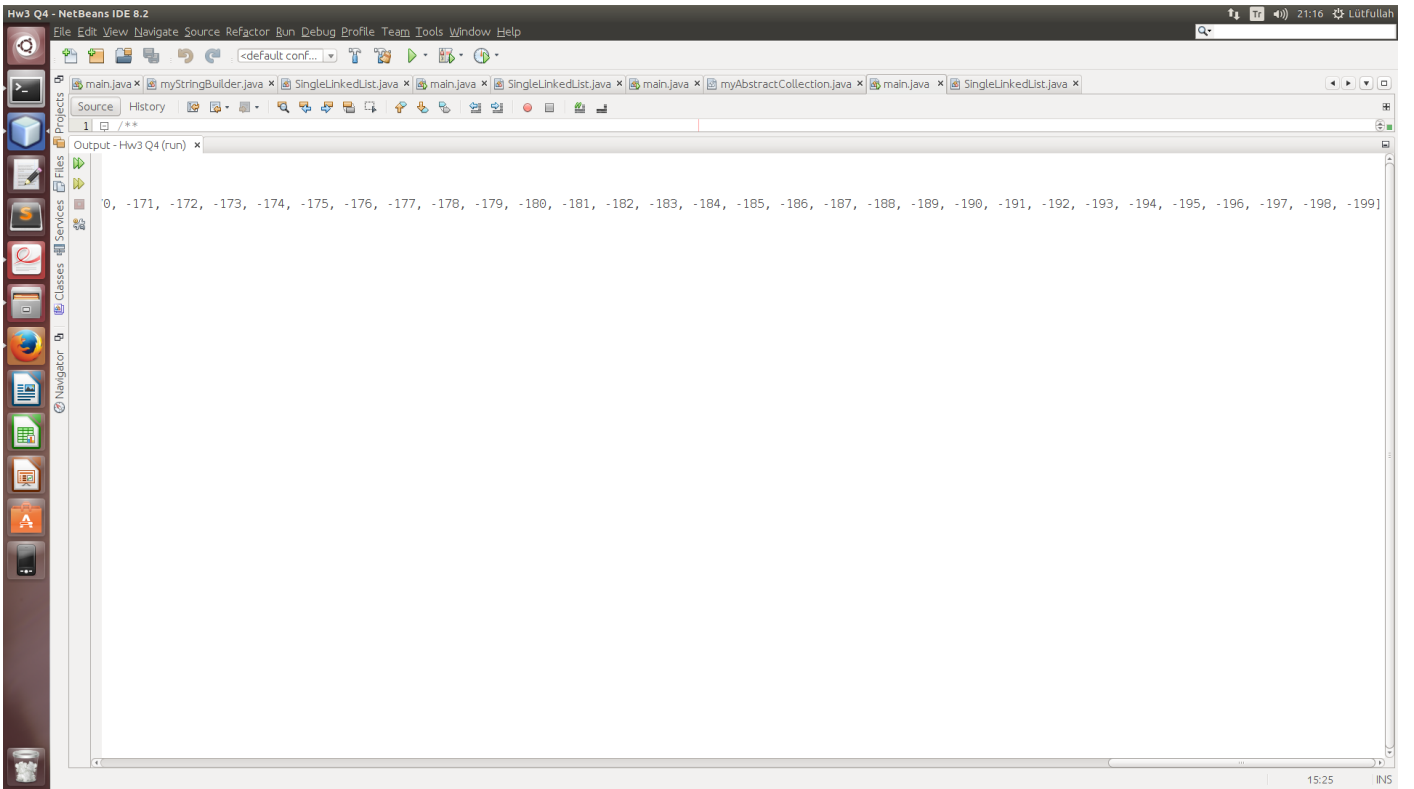
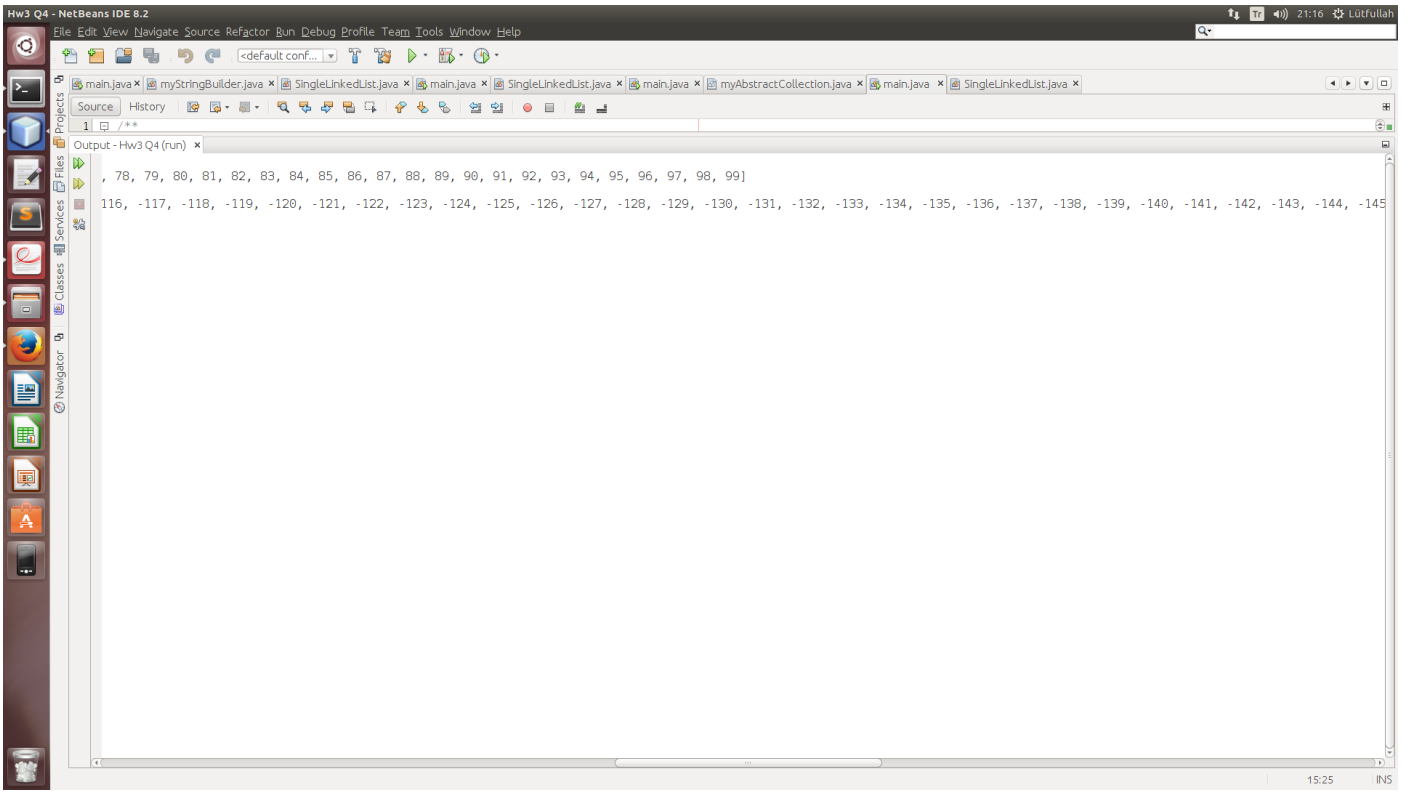
4.Soru için :



```
run:
100 integers added to the list ==> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
50 integers removed from the list ==> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
100 integers Re-added to the list ==> [50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
5, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, -100, -101, -102, -103, -104, -105, -106, -107, -108, -109, -110, -111, -112, -113, -114, -115, -116, -117
```

Ekran görüntüleri sıralı şekilde aynı ekran görüntüsünün devamıdır.
Testi başarıyla geçmiş silinen 50 elemanı tekrar kullanmıştır. Test sırasında düzen karışmasın diye farklı sayılar kullandım. Sonradan eklenen 100 sayıyı -100 den başlattım .ilk 100 de 0 dan itibaren 99 a kadar olan sayılardır.