Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK HW3 REPORT

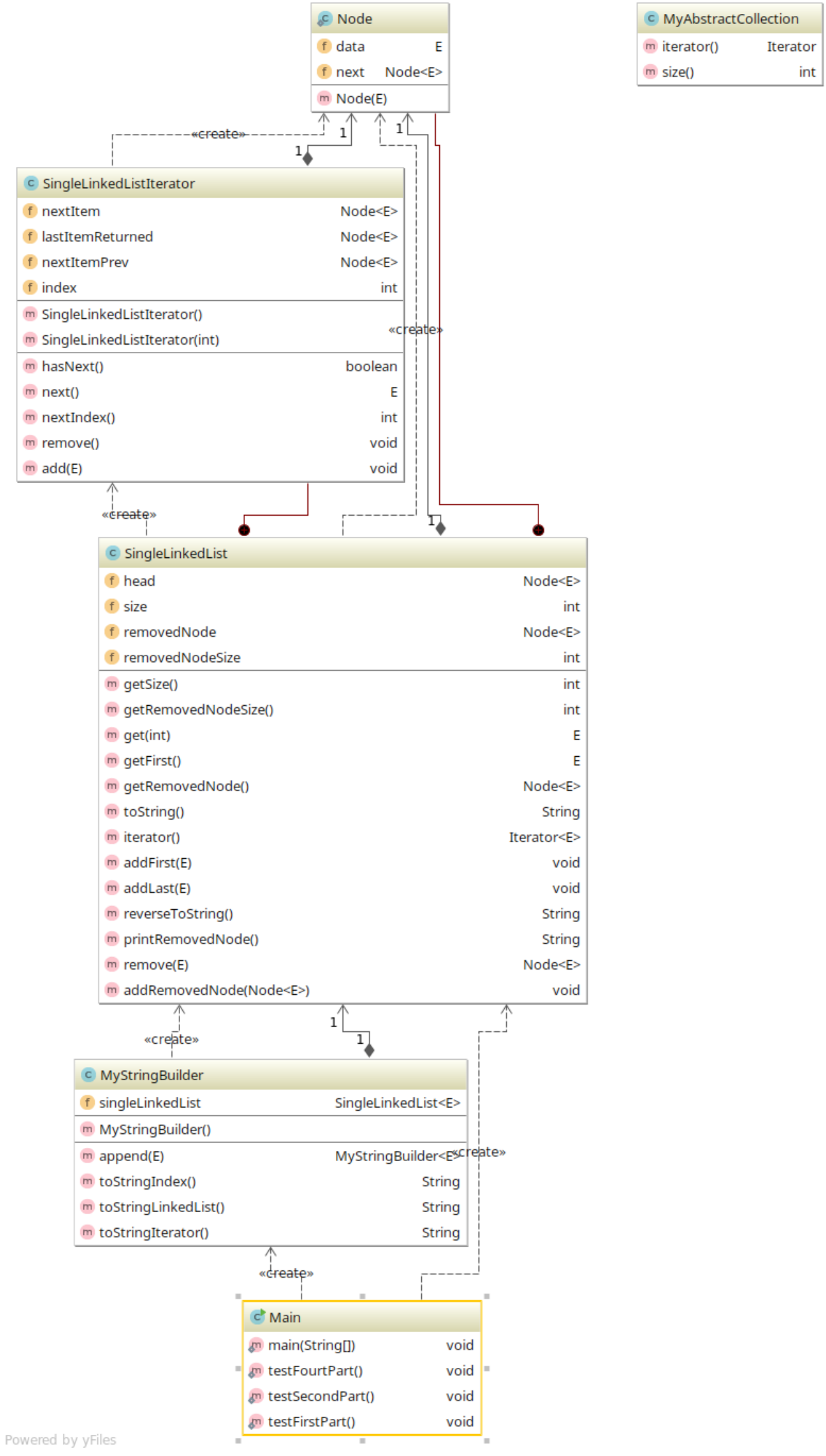
MEHMET GÜROL ÇAY
121044029

Course Assistant:

1. System Requirements

IntelliJ IDEA 2016.3.5
JRE: 1.8.0_112-release-408-b6 amd64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

2. Class Diagrams


Node
data E
next Node<E>
Node(E)
MyAbstractCollection
iterator() Iterator
size() int
«create»
1
1
1
SingleLinkedListIterator
nextItem Node<E>
lastItemReturned Node<E>
nextItemPrev Node<E>
index int
SingleLinkedListIterator()
SingleLinkedListIterator(int)
«create»
hasNext() boolean
next() E
nextIndex() int
remove() void
add(E) void
«create»
1
SingleLinkedList
head Node<E>
size int
removedNode Node<E>
removedNodeSize int
getSize() int
getRemovedNodeSize() int
get(int) E
getFirst() E
getRemovedNode() Node<E>
toString() String
iterator() Iterator<E>
addFirst(E) void
addLast(E) void
reverseToString() String
printRemovedNode() String
remove(E) Node<E>
addRemovedNode(Node<E>) void
«create»
1
1
«create»
MyStringBuilder
singleLinkedList SingleLinkedList<E>
MyStringBuilder()
append(E) MyStringBuilder<E>
toStringIndex() String
toStringLinkedList() String
toStringIterator() String
«create»
«create»
Main
main(String[]) void
testFourtPart() void
testSecondPart() void
testFirstPart() void
Powered by yFiles

3. Problem Solutions Approach

Q1) SingleLinkedList ders kitabından faydalanılarak implement edildi. Iterator metodları implement edildi. Iterator içerisinde nextItemPrev tutularak bir önceki eleman hakkında bilgi tutmuş olduk. Bunu tutmadan araya eklemenin yolu bulunamadı. Üç farklı *toString()* metodu yazıldı. Bunlar *toStringIndex()* , *toStringLinkedList()* ve *toStringIterator()* şeklinde. Sırasıyla *index*'e göre *LinkedList*'e göre ve *Iterator'*a göre çalışmaktadır.
100.000 farklı sayıyı dosyaya yazdıktan sonra okurken *MyStringBuilder* sınıfında yazmış olduğumuz *append* metodunu kullandım. a*ppend()* metodu için *addLast()* metodunu kullandım. Böylece verilen eleman iterator kullanılarak sona eklenmiş oldu.
Dosyadan okuduğum her bir sayıyı *append()* metodu ile bir string'e ekledikten sonra bunu *toString()* metodlarım ile yazmaya verdim. Aralarındaki performans farkını ölçmek için de javanın *System.nanoTime()* metodunu kullandım. Böylece işin başlama ve bitiş sürelerinde zaman tutmuş oldum. Farkları ile çalışma sürelerini hesapladım.

Q2) Elimizde zaten boyutu belli olan bir *SingleLinkedList* mevcut bu yüzden bu listenin *size'*ını kullanarak tersten yeni bir *string* oluşturduk. *Index* kullanarak yapılan bu işlem O(n) sürede çalışmaktadır.

Q3)

Q4) Silinen *node*'ları bir yerde tutmamız gerekmekte ki bu garbage collector daha az çalışsın. Bunun için *SingleLinkedList* içerisinde *removedNode* adlı bir *node* daha tuttuk. Silinen her eleman silinme aşamasında bu *node*'un *next'*i olarak eklenir. Böylece elimizde silinen elemanlardan oluşan bir liste olur.

6. Test Cases

Testler *main* class içinde yapıldı.

7. Running and Results

Q1) Aşağıdaki ekran çıktısına göre *index* ile çalışan *toStringIndex()* metodu yaklaşık 79 saniye çalışmaktadır. Çünkü O(n^2) de çalışmakta. *toStringIndex()* metodu kendi içinde *size'*a kadar dönerken aynı zamanda *get()* metodu da alınan *index* değerine *head*'den başlayarak ulaşmaya çalışmaktadır. Bu işlemi her seferinde yapmaktadır.
ToStringIterator() ise *iterator* ile her seferinde sadece bir sonraki elemanı kontrol etmektedir. Böylece listedeki eleman sayısı kadar dolaşacak yani O(n) sürede çalışacaktır.
ToStringLinkedList() ise listenin kendi *toString()* metodunu kullanmaktadır. Bu method ise *nextItem null* olana dek dolaştığı için O(n) sürede dolaşacaktır.
Bu yüzden en yüksek *toStringIndex()* metodu çalışacaktır.
Çalışma sürelerini aşağıdaki görselden daha net görebilirsiniz.
Diğer soruların çıktılarıda aşağıdaki görselde vardır.