Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 03 REPORT

Yasin AÇIKGÖZ
131044070

1. Sistem Gereksinimleri

Programın çalışabilmesi için Java 1.80 gereklidir. Intellij IDEA ile Windows ve Linux işletim sistemlerinde çalışabilmektedir.

2. Class Diagramları

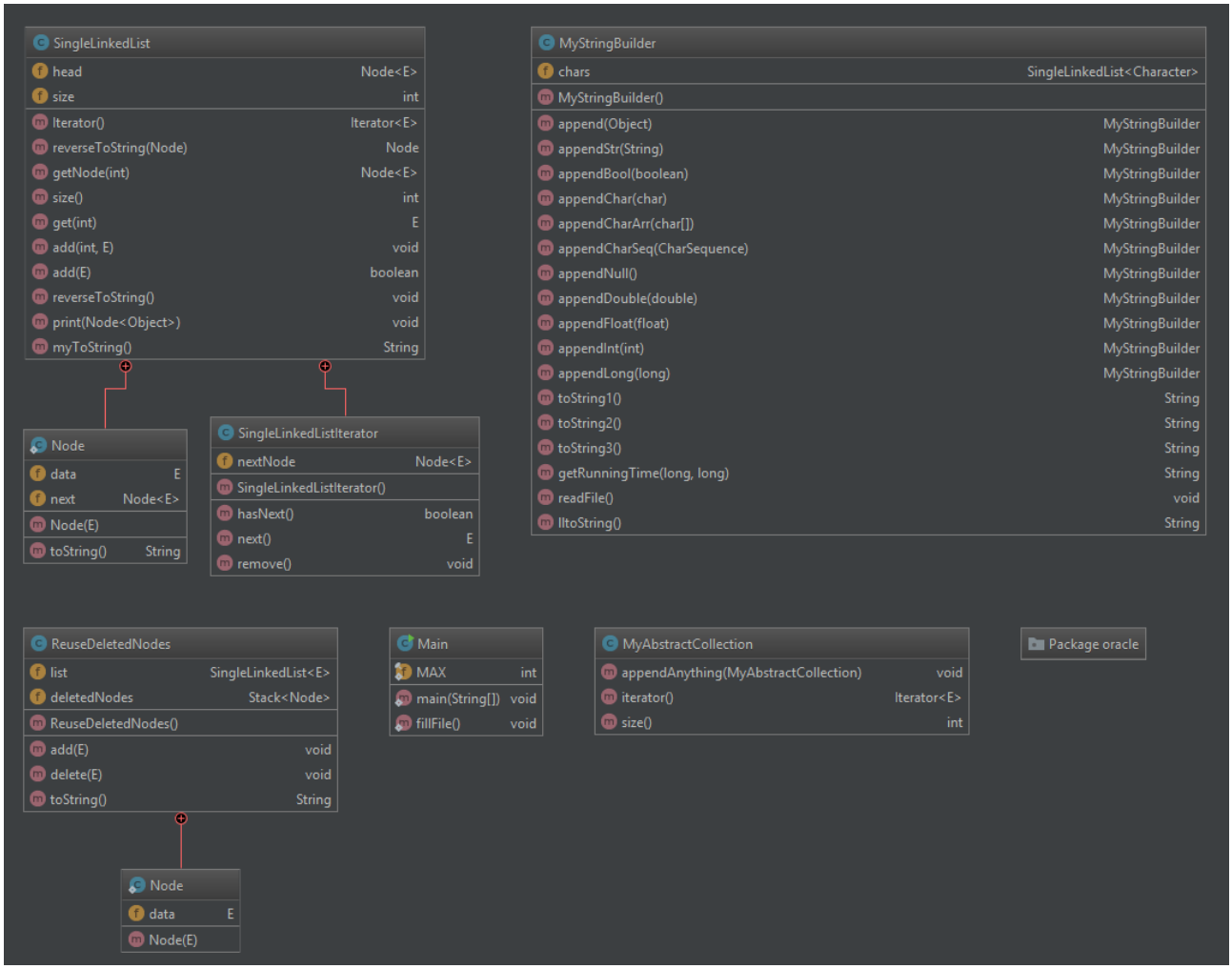

Figure 1 Use Case Diagram

3. Problem Çözüm Yöntemleri

PART1

Bu partta bizden istenilen bir StringBuilder sınıfı implement etmemizdi. Bu sınıfı implement etmek için öncelikle kitaptaki LinkedList implementasyonunu kullanarak SingleLinkedList sınıfı implement ettim. Bu sınıf içerisinde inner class şeklinde Node sınıfını ve Iterator'u implement ettim. Iterator'ü implement ederken standart Java kütüphanesindeki Iterator interface'ini implement ettim.

Ardından bu sınıfı kullanarak MyStringBuilder sınıfının içerisinde bir karakter SingleLinkedList'i tuttum. Bu liste, objeleri birleştirmek için kullanacağımız append fonksiyonu çağırıldıkça her bir objenin toString metodunu çağırarak stringin her karakterini linked list'in bir node'una eklemeye yaramaktadır.

```
public class MyStringBuilder <E>{
   SingleLinkedList<Character> chars;
…
}
```

Parametre olarak obje alan bir append metodu implement ettim ve bu metodun içerisinde instanceof keyword'ünü kullanarak gelen objenin tipini tespit ettim. Ardından gelen tipe göre gerekli fonksiyonu çağırarak ekleme işlemi yaptım.

Son olarak main'de içerisine 10 bin adet integer koyarak oluşturduğum input.txt dosyasından readFile metoduyla chars linkedlistine okudum ardından

toString1( ) metodunda;

SingleLinkedList sınıfının get metodunu kullanarak result1.txt dosyasına birleştirilen stringi yazdım. Bu işlemde her bir node için get metodu head'den o node'a kadar gidiliyor. Bu işlem n. sıradaki node için (n*(n+1))/2 kere yapılıyor. Bu nedenle bu işlemin karmaşıklığı $O(n^2)$ dir.

toString2( ) metodunda;

SingleLinkedList sınıfındaki iterator sınıfını kullanarak döngü içerisinde hasNext() false return edene kadar result2.txt dosyasına birleştirilen stringi yazdım. Iterator kullanıldığı için her bir node'a erişim O(1) zamanda olur bu işlem n kere tekrar edildiği için bu işlemin karmaşıklığı O(n) zaman alır.

toString3( ) metodunda;

Java'nın standart LinkedList sınıfındaki toString metodunu kullanarak result3.txt dosyasına birleştirilen stringi yazdım.

PART2

Bu partta her bir node'u obje içeren bir linked list oluşturmamız ve bu linked listi reverse edilmiş şekilde ekrana basmamız istendi. Bu metod bir node alır ve reverse edilmiş node'un head'ini return eder.

```
   private Node reverseToString(Node node){ }
```

Bu metodun base case'i node'un null olması veya node'un next'inin null olmasıdır. Recursion'dan dönerken ilerideki node ile gerideki node arasındaki bağlantıyı ters çevirir. Ardındaki gerideki node ile önündeki node arasındaki bağlantıyı yok eder ve kalan ilk node'u return eder.

Bu metodu test etmek için SingleLinkedList sınıfı içerisindeki parametre almayan bir reverseToString metodu implement ettim ve testi onun içerisinde yapıp ters çevrilmiş linked list'i ekrana bastım.

PART3

Bu kısımda abstract class olarak MyAbstractCollection sınıfı implement ettim. Bu sınıf içerisindeki appendAnything( ) metodu parametre olarak aldığı MyAbstractCollection objesinin iterator'ünü oluşturur ve bu iterator ile döngü üzerinden hasNext() ile kendi elemanlarını çağrıldığı objenin elemanlarına ekler.

```
public void appendAnything(MyAbstractCollection obj){
    Iterator<E> iter = obj.iterator();
    while(iter.hasNext())
        this.add(iter.next());
}
```

PART4

Bu partta kendi yazdığım SingleLinkedList sınıfını kullanarak oluşturduğum linkedList objesinden sildiğim elemanları node objeleri içeren bir Stack'te tuttum. Bu stack'te eleman yok iken linkedList'e normal olarak add metoduyla eleman ekledim.

Eleman silme işlemi yaparken öncelikle get metodunu kullanarak silinmek istenen elemanın index'ini buldum. Ardından iterator'ün next metodunu kullanarak linkedlist üzerinde gezdim ve bir counter tuttum. Eğerki bu counter verilen index'e eşitse next metodunu bir daha çağırmadım ve delete metodunu çağırdım. Bu metodun içerisinde gelen node objesinin data kısmına null atadım. Ardından yeni bir node objesi oluşturarak bunu silinen elemanları tuttuğum stack'e push ettim.

```
public void add(E obj){
    if(deletedNodes.isEmpty())
        list.add(list.size(),obj);
    else{
        Node node =  deletedNodes.pop();
        node.data=obj;
        list.add(list.size(), (E) node.data);
    }
}
```

Add metodunu çağırırken eğer ki silinen elemanları tuttuğum stack boş değilse bu stack'in ilk elemanının pop ettim ve bu elemanın data'sına linked list'e ekleyecek olduğum elemanın datasını atadım ve elemanı add metodunu kullanarak linkedList'ime ekledim.

Böylelikle garbage collector kullanmadan linkedList'ten eleman silmeyi gerçekleştirdim.

6. Test Cases

Test Case No	Girdiler	Beklenen Çıktı	Gerçekleşen Çıktı	fDurum
1	1’den 10 bine kadar tüm sayılar	result1.txt’de append edilmiş string	result1.txt	Başarılı Geçen süre 1,61 saniye
2	1’den 10 bine kadar tüm sayılar	result2.txt’de append edilmiş string	result2.txt	Başarılı Geçen süre 0,07 saniye
3	1’den 10 bine kadar tüm sayılar	result3.txt’de append edilmiş string	result3.txt	Başarılı Geçen süre 0,06 saniye
4	Elemanları sırasıyla y,a,s,i,n,a,c,i,k,g,o,z harfleri olan SingleLinkedList	Reverse edilmiş SingleLinkedList	Elemanları sırasıyla z,o,g,k,i,c,a,n,i,s,a,y olan karakter LinkedList	Başarılı
5	Elemanları sırasıyla “yasin” ve “acikgoz” stringleri olan SingleLinkedList	Reverse edilmiş SingleLinkedList	Elemanları sırasıyla “acikgoz” ve “yasin” olan string LinkedList	Başarılı
6	Elemanları sırasıyla 1, 9, 0, 5 integerları olan SingleLinkedList	Reverse edilmiş SingleLinkedList	Elemanları sırasıyla 5, 0, 9, 1 olan integer LinkedList	Başarılı
7	ReuseDeletedNodes listesine 0’den 100’e kadarki sayilari ekle	1’den 100’e kadar ardışık sayılar	1’den 100’e kadar ardışık sayıların listesinin ekranda görünmesi	Başarılı
8	ReuseDeletedNodes listesinden 50’den 100’e kadarki sayilari çıkar	0’dan 50’ye kadarki ardışık sayılar	Deleted Node List’te 0’dan 50’ye kadarki ardışık listesinin ekranda görünmesi	Başarılı
9	ReuseDeletedNodes listesine 100’den 150’ye kadar sayilari ekle	1’den 50’ye ve 100’den 150’ye kadarki ardışık sayılar	1’den 50’ye ve 100’den 150’ye listesinin ekranda görünmesi Deleted Node List’’in boş olması	Başarılı

7. Running and Results

```
/*****************************PART 1*************************************/
Execution time is 1,60900 seconds for toString1() method.

Execution time is 0,06100 seconds for toString2() method.

Execution time is 0,06300 seconds for toString3() method.

/*************************PART 2*************************************/

Character SingleLinkedList before reverseToString() call
yasinacikgoz
Character SingleLinkedList after reverseToString() call
zogkicanisay

String SingleLinkedList before reverseToString() call
yasinacikgoz
String SingleLinkedList after reverseToString() call
acikgozyasin

Integer SingleLinkedList before reverseToString() call
1905
Integer SingleLinkedList after reverseToString() call
5091
/*************************PART 4*************************************/
List (1-100)
[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, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 4
List After Deleting (50-99)
[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, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 4
Deleted Node List (50-99)
[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, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
List After Inserting (1-50) - (100-150) (
[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, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 4
Deleted Node List
```