Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 3 REPORT

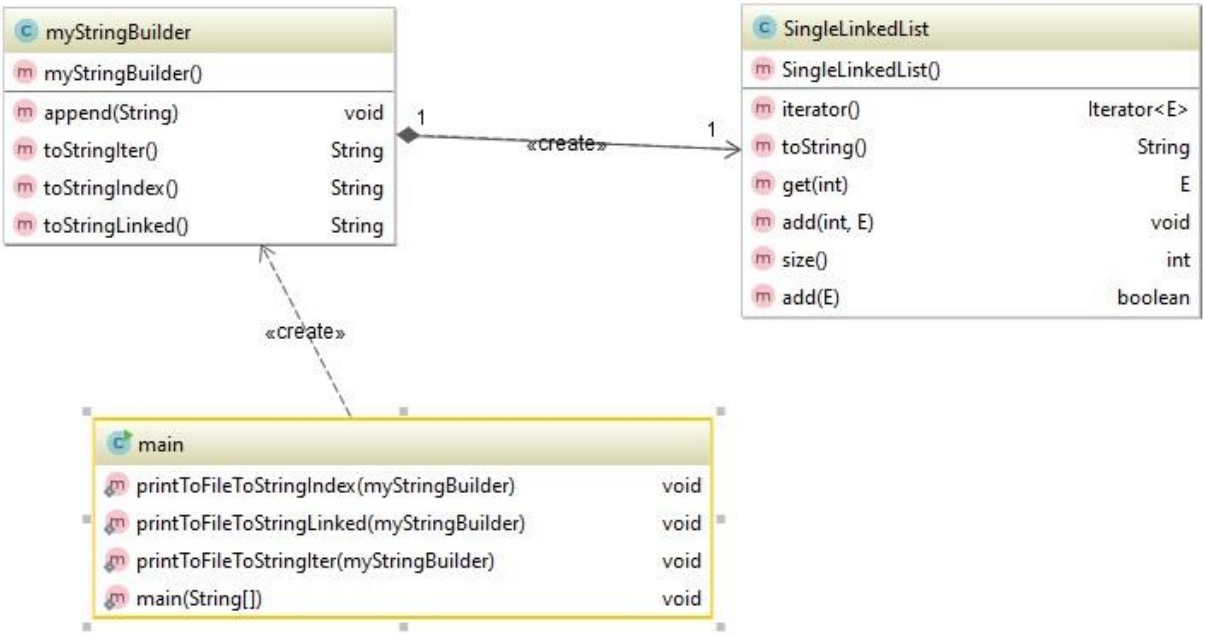
SEYİT AHMET KARACA
141044084

Course Assistant:Nur Banu ALBAYRAK

(Q1) 1- System Requirements

StringBuilder ile aynı işlevi olan myStringBuilder yazıldı.Append methodu ile gelen verileri tuttuğu verilerin sonuna eklemesi sağlandı.Veriler single linked list yapısında tutuldu. MyStringBuilder sınıfı içerisinde farklı yapılarla implement edilmiş üç adet toString methodu oluşturuldu ve bunlara 100000 tane tam sayı eklenerek dosyaya yazıldı.Dosyaya yazma işlemi, karmaşıklık hesabı yapılıp saniye cinsinden süreleri tutuldu ve bunlar karşılaştırıldı.

2. Class Diagrams



3. Problem Solutions Approach

Öncelikle myStringBuilderın verilerinin nerede tutulacağı ile iligili çözüm olarak single link list implement edildi. Single linked list için iteratör sınıfı yazıldı ve düğümlerin üzerinden geçilerek işlem yapılması sağlandı.Sonrasında myStringBuilder sınıfı implement edilerek yazılmış olan single linked list kullanılarak myStringBuildera eleman eklenmesi sağlandı. Geriye sadece toString methodlarını implement etmek kaldı ve onlarda yapmak için linked list sınıfı method ve iteratörü ile yazıldı.

4. Test Cases

```java
/**
 * toStringIter ile dosyaya yazar
 * @param msBuilder
 * @throws IOException
 */
public static void printToFileToStringIter(myStringBuilder msBuilder) throws IOException {
    long starttime,stopTime,result;
    FileWriter fw = new FileWriter(new File( pathname: "result3.txt"));
    PrintWriter pw = new PrintWriter(fw);

    starttime = System.currentTimeMillis();
    pw.println(msBuilder.toStringIter());

    stopTime = System.currentTimeMillis();
    result = stopTime-starttime;
    result /= 1000;
    System.out.println("toStringIter : "+result);

    pw.close();
    fw.close();

}
```

```java
public class main {
    /**
     * toStringIdex ile dosyaya yazar
     * @param msBuilder
     * @throws IOException
     */
    public static void printToFileToStringIndex(myStringBuilder msBuilder) throws IOException {
        long starttime,stopTime,result;
        FileWriter fw = new FileWriter(new File( pathname: "result1.txt"));
        PrintWriter pw = new PrintWriter(fw);

        starttime = System.currentTimeMillis();
        pw.println(msBuilder.toStringIndex());

        stopTime = System.currentTimeMillis();
        result = stopTime-starttime;
        result /= 1000;
        System.out.println("toStringIndex : "+result);
        pw.close();
        fw.close();
    }
```

```java
/**
 * toStringLinked ile dosyaya yazar
 * @param msBuilder
 * @throws IOException
 */
public static void printToFileToStringLinked(myStringBuilder msBuilder) throws IOException {
    long starttime,stopTime,result;
    FileWriter fw = new FileWriter(new File( pathname: "result2.txt"));
    PrintWriter pw = new PrintWriter(fw);

    starttime = System.currentTimeMillis();
    pw.println(msBuilder.toStringLinked());

    stopTime = System.currentTimeMillis();
    result = stopTime-starttime;
    result /= 1000;

    System.out.println("toStringLinked : "+result);
    pw.close();
    fw.close();
}
```

printToFileToStringIndex(myStringBuilder)
printToFileToStringIter(myStringBuilder)
printToFileToStringLinked(myStringBuilder)

Program yuakrıdaki 3 method ile test edilebilir.

5. Running and Results

```java
public String toStringLinked(){
    return list.toString();    O(n)
}
```

```java
public String toString(){
    Node<E> nodeRef = (Node<E>)head;     Ω(1)
    String result="";                    Ω(1)
    while(nodeRef != null){                        |
        result += ""+nodeRef.data;       Ω(1)      |
                                                   |
        if(nodeRef.next != null){        Ω(1)      | O(n)
            result +=" ";                Ω(1)      |
        }                                          |
        nodeRef = nodeRef.next;          Ω(1)      |
    }                                              |
    return result;                       Ω(1)
}
```

toStringLinked methodunun karmaşıklığı O(n)

```
public String toStringIter(){
    String str="";   Ω(1)
    while(iter.hasNext()){                  |
        str += iter.next() +" "; Ω(1)       | O(n)
    }                                       |
    return str; Ω(1)
}
```

```
    public boolean hasNext() {
        if(iterNode == null){       Ω(1)
            return false;           Ω(1)
        }else{
            return true;            Ω(1)
        }
    }
```

```
    public E next() {
        if(hasNext()){      Ω(1)
            E _data = (E)iterNode.data; Ω(1)
            iterNode = iterNode.next;   Ω(1)
            return _data;               Ω(1)
        }
        return null;        Ω(1)
    }
```

toStringIter methodunun karmaşıklığı O(n)

```
public String toStringIndex(){
    int i=0;                    Ω(1)
    String str= "";             Ω(1)
    while(i < list.size()){             |
        str += list.get(i)+" "; O(n)  | O(n)
        i++;                    Ω(1)    |
    }
    return str;
}
```

```
        public int size(){
            return size;        Ω(1)
        }
```

```
        public E get(int index){
            if(index < 0 || index >= size){
                throw new IndexOutOfBoundsException(Integer.toString(index));
        Ω(1)
            }
            Node<E> node = getNode(index);      O(n)
            return node.data;                   Ω(1)
        }
```

```
                private Node<E> getNode(int index){
                    Node<E> node = head;
                    for(int i=0; i < index && node != null ; i++)      |O(n)
                        node = node.next;       Ω(1)                   |
                    return node;
                }
```

toStringIndex methodunun karmaşıklığı $O(n^2)$

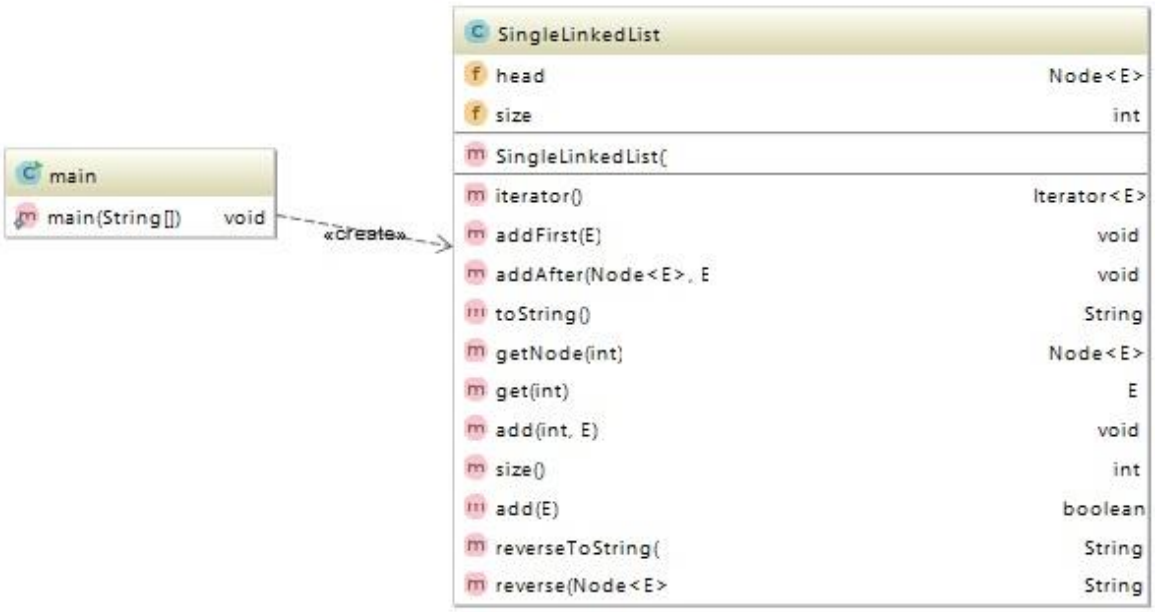
```
toStringIndex : 58
toStringIter : 23
toStringLinked : 47
```

Index ile yapılanin karmaşıklığı $O(n^2)$, iterator ile yapılanın O(n) ve linked list ile yapılanın karmaşıklığı O(n) çıktığı halde aralarında çalıştığındaki süre farkı sadece iteratörle olanda gözle görülür bir fark var.Index ile yapılanın $O(n^2)$ olduğu için süresi doğal fakat linked listle implement edilmiş halinde fazla statment olduğu ve string değişkene eklenerek yapıldığı için süresi arttı.

(Q2) 1- System Requirements

SingleLinkedList'in toString methodu ile oluşturulan stringin tam tersi sıralama ile recursive olarak reverseToString yazıldı.

2. Class Diagrams



3. Problem Solutions Approach

SingleLinkedList in elemanlarına erişmek için sınıfın içerisinde yazılması gerekti methodun.Dışarıdan aynı toString gibi çalışabilmesi için reverseToString isimli private bir method oluşturulması gerekti. ReverseToString methodunda head nodunun bir kopyası oluşturularak onu listin üzerinde gezebilmesi için yardımcı methoda göderildi.Node'un nexti ile en son elemana gidilip recursive olarak son elemana ulaşıldığında son elemanı alarak geriye gelirken üzerinden geçtiği elemanları bir string değişkende tutulması sağlandı.Yardımcı methodun çıktısı reverseToString ile döndürüldü.

4. Test Cases

```
SingleLinkedList<String> sl =new SingleLinkedList<String>();
sl.add("kalem");
sl.add("silgi");
sl.add("2");
sl.add("fare");
sl.add("telefonu");
sl.add("yedi");

SingleLinkedList<Integer> s2 =new SingleLinkedList<Integer>();

s2.add(1);
s2.add(2);
s2.add(3);
s2.add(4);
s2.add(5);

System.out.println(sl.toString());
System.out.println(sl.reverseToString());
```

5. Running and Results

```
SingleLinkedList<String> sl =new SingleLinkedList<~>();
sl.add("kalem");
sl.add("silgi");
sl.add("2");
sl.add("fare");
sl.add("telefonu");
sl.add("yedi");

SingleLinkedList<Integer> s2 =new SingleLinkedList<~>();

s2.add(1);
s2.add(2);
s2.add(3);
s2.add(4);
s2.add(5);

System.out.println(sl.toString());
System.out.println(sl.reverseToString());
System.out.println();
System.out.println(s2.toString());
System.out.println(s2.reverseToString());
```

in

```
kalem silgi 2 fare telefonu yedi
yedi telefonu fare 2 silgi kalem

1 2 3 4 5
5 4 3 2 1
```

İki farklı tipteki list ile toString ve reverseToString methodları çağrıldı ve çıktıları listelendi.

Q3 – System Requirements

appendAnything methodunun calisabilmesi icin myAbstractCollection sınıfından extend edilmis siniflarda sinif üzerinde iterator ve add methodları override edilmis olmasi gerekir.appendAnything methodu için gerekli bu iki method implement edilmiş ise kullanılabilir ve bu sayede bu sınıftan extend olmuş birden farklı sınıf atası aynı olduğu için appendAnything sayesinde birbirlerini tanıyıp sonlarına eleman olarak eklenebilicektir.

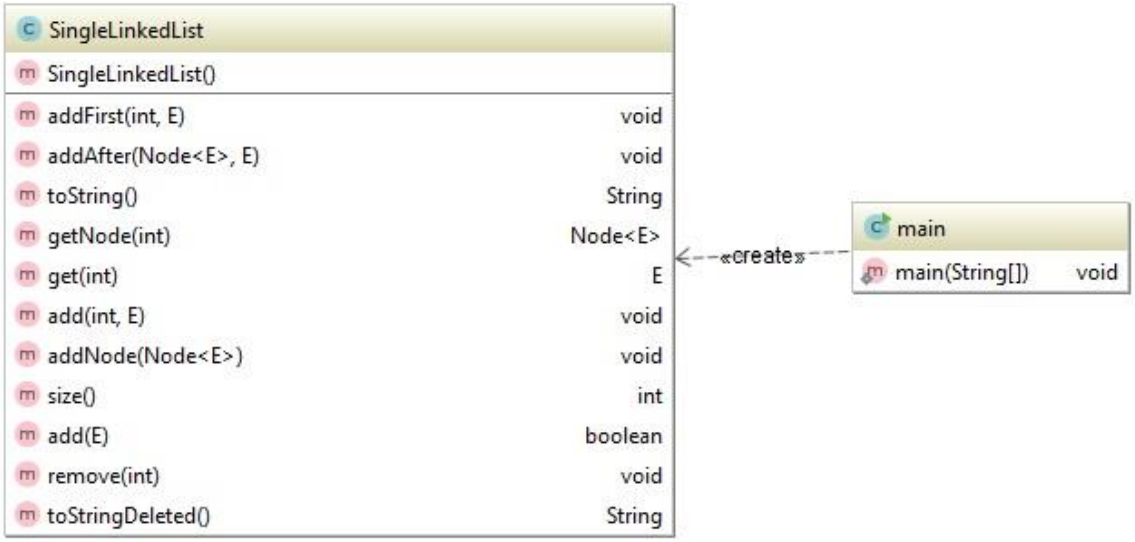
```
public abstract class myAbstractCollection<E> extends AbstractCollection{
    /**
     * myAbstractCollection ve ondan tureyen herhangi bir sinif
     * parametresi alip bu sinifi cagiran sinifa ekler
     * @param other
     */
    public void appendAnything(myAbstractCollection<E> other){
    Iterator<E> otherIter = other.iterator();

        while(otherIter.hasNext()){
            this.add(otherIter.next());
        }
    }
}
```

Q4- 1.System Requirements

SingleLinkedList implement edildi ve bu linked listin diğerlerinden farklı olarak tuttuğu elemanları garbage collection'ı daha az kullanmak adına silindiğinde elemanları ayrı bir yapıda tutuldu.ArrayList'te silinmiş nodelar tutuldu.Eleman eklendiğinde linked liste , silindiğinde silinen eleman arrayliste eklendi.Tekrar aynı eleman eklenmek istendiğinde önce arraylistte ilgili node arandı ve bulundu ise node linked listin sonuna eklendi ve arraylistten çıkarıldı.Linked listten silme işlemleri indexe göre yapıldığı için for veya while gibi döngülerde kullanımı epey işlevsiz oluyor.

2. Class Diagrams



3. Problem Solutions Approach

Problem önce silinecek nodlar ayrı bir deletedHead denerek tutacaktım fakat 9 saat boyunca o yöntemi denesemde başarılı olamadığım için alternatif yollara başvurdum.Linked listten silinen nodeları baştan ve sondan silmek benim için sorundu.Bu sorunumu iki ayrı durum içinde düşündüm.Elemanları silerken bir önceki node'a ulaşıp nextini arrayliste onunda nextini ilk bulduğum elemana atayarak bu işlemi hallediyorum.Bu durum sonda ve başta farklı şekilde düşünüp hallettim.İndex size kadar gelirse size-1. Node üzerinden bu işlemi yaptım.Eğer ilk node gelirse head i arrayliste attım ve head e head.next i atayıp bu durumuda çözdüm.

4. Test Cases

```
        SingleLinkedList<Integer> sl = new SingleLinkedList<Integer>();
        for(int i= 0; i < 100 ; i++)
            sl.add(i);

        for(int i= 50 ; i < 100 ; i++)
            sl.remove(i);

//        for(int i= 0; i < 100 ; i++)
  //          sl.add(i);

        System.out.println(sl.toStringDeleted());
        System.out.println(sl.toString());
    }
}
```

```
in
"C:\Program Files\Java\jdk1.8.0_112\bin\java" ...
50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98
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 37 38 39 40 41 42 43 44 45 46 47 48 49 51
```

```java
public class main {
    public static void main(String[] args) {
        SingleLinkedList<Integer> sl = new SingleLinkedList<Integer>();
        for(int i= 0; i < 100 ; i++)
            sl.add(i);

        for(int i= 50 ; i < 100 ; i++)
            sl.remove(i);

        for(int i= 50; i < 75 ; i++)
            sl.add(i);

        System.out.println(sl.toStringDeleted());
        System.out.println(sl.toString());
    }
}
```

```
ain
  "C:\Program Files\Java\jdk1.8.0_112\bin\java" ...
  76 78 80 82 84 86 88 90 92 94 96 98
  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 37 38
```

5. Running and Results

Sonuçlar ve test kodlarım bir önceki başlıkta bulunuyor.Silme işlemim indexe göre olduğu için ilk silinenlerde çift sayı şeklinde silinenler gözükmekte.
50 ve 75 arasını tekrar eklediğimde ise eklenen elemanlar toStringDeleted'tan silindi ve tekrar linked liste geri eklenmekte.

Github : https://github.com/SeyitAhmetKARACA/141044084_HW03