

Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 03 REPORT

EMRULLAH GENCOGLU
121044006

Course Assistant:

PART-1

1.System Requirement


Bu part ta kendi stringBuilder sınıfımızı oluşturmaktayız. Veri yapısı olarak SingleLinkedList kullanılmıştır. SingleLinkedList sınıfının implementasyonunda bazı methodları kitabın kodundan faydalanılmıştır. MyStringBuilder sınıfımızda üç farklı toString() ve append() methodu implement edilmiştir.

2.Class Diagram

3.Problem Solution Approach

public void append(E obj) methodu singlelinkedlist te yeni bir Node ekler.


```
public void append(E obj) {  
    sll.addFirst(obj);  
}
```



$O(1)$ (addFirst method head e ekleme yaptığı için constant time olur)

public String toString() methodu singlelinkedlist te Node lara indexlerle ulaşarak String oluşturmaktadır.


```
public String toString() {  
    String result="";  
    for(int i=0;i<sll.getSize();++i){  
        result+=sll.get(i);  
        if(i != sll.getSize()-1)  
            result += " ->";  
    }  
    //System.out.println("result 1 "+result);  
    return result;  
}
```



$O(n)$

public String toStringWithIterator() methodu Node lara iteratorle oluşturarak String oluşturur.

```
public String toStringWithIterator() {  
    Iterator it=sll.iterator();  
    String result = "";  
    while(it.hasNext())  
    {  
        result +=it.next();  
        result += " ->";  
    }  
    result +=sll.get(0);  
    // System.out.println("result 2 "+result);  
    return result;  
}
```



$O(\log n)$

public String toStringWithLinkedList() singleLinkedList te implement edilen toString methodunu kullanarak String oluşturmaktadır.

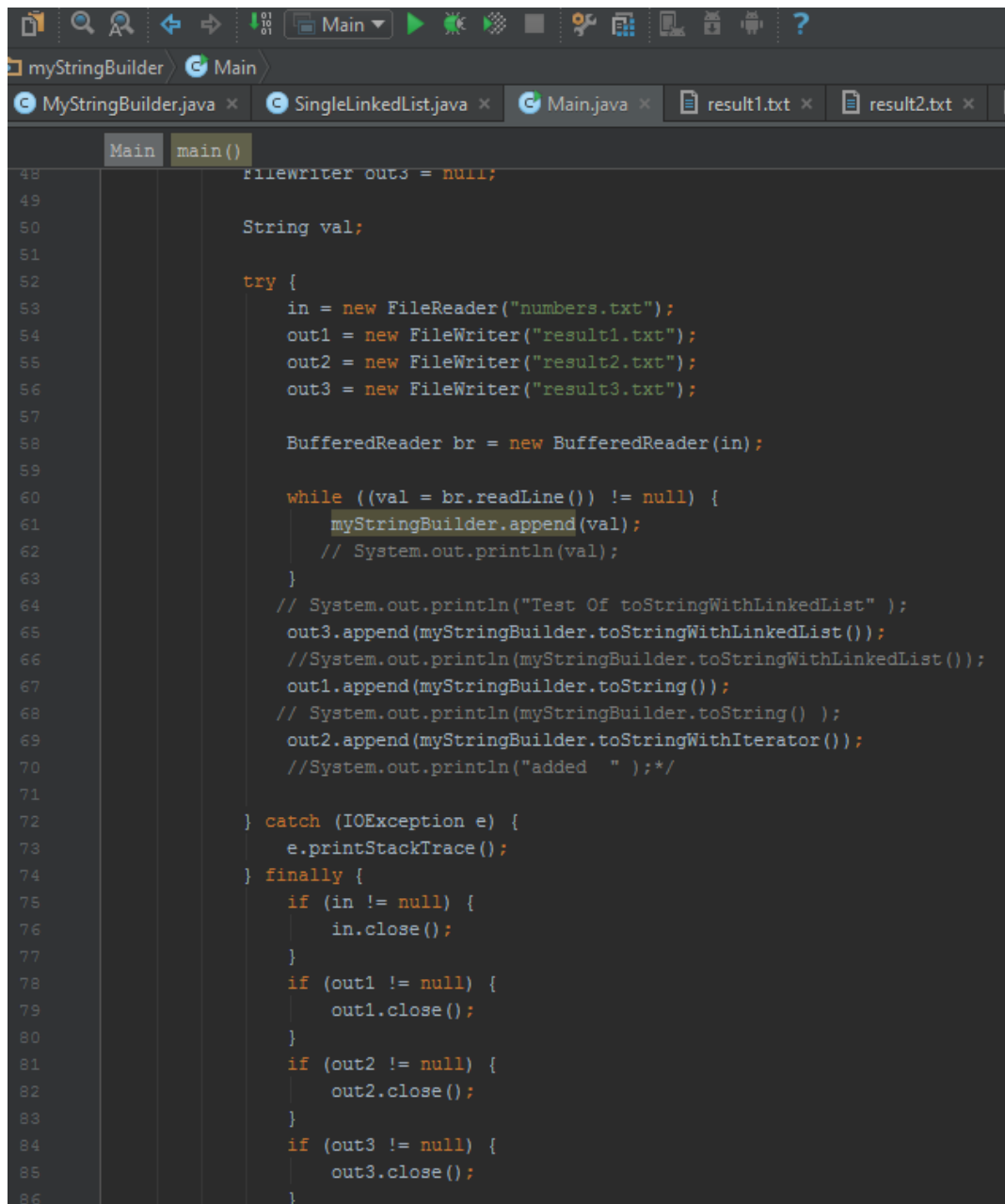
6. Test Cases

append() methodunun testi => dosyadan okunan sayıların eklenmesi

toString() methodunun testi => tutulan Node leri String olarak “result1.txt” dosyasına yazar.

toStringWithIterator() test => tutulan Node leri String olarak “result2.txt” dosyasına yazar.

toStringWithLinkedList() test => tutulan Node leri String olarak “result3.txt” dosyasına yazar.



```
48      FileWriter out3 = null;
49
50      String val;
51
52      try {
53          in = new FileReader("numbers.txt");
54          out1 = new FileWriter("result1.txt");
55          out2 = new FileWriter("result2.txt");
56          out3 = new FileWriter("result3.txt");
57
58          BufferedReader br = new BufferedReader(in);
59
60          while ((val = br.readLine()) != null) {
61              myStringBuilder.append(val);
62              // System.out.println(val);
63          }
64          // System.out.println("Test Of toStringWithLinkedList" );
65          out3.append(myStringBuilder.toStringWithLinkedList());
66          //System.out.println(myStringBuilder.toStringWithLinkedList());
67          out1.append(myStringBuilder.toString());
68          // System.out.println(myStringBuilder.toString() );
69          out2.append(myStringBuilder.toStringWithIterator());
70          //System.out.println("added " );*/
71
72      } catch (IOException e) {
73          e.printStackTrace();
74      } finally {
75          if (in != null) {
76              in.close();
77          }
78          if (out1 != null) {
79              out1.close();
80          }
81          if (out2 != null) {
82              out2.close();
83          }
84          if (out3 != null) {
85              out3.close();
86          }
87      }
```


Part-2

1.System Requirement

Bu partta recursion olarak Node larin sırasının tersten yazdırılmasıdır. **reverseToString()** methodu implement edilmiştir. Parametre olarak node almasından dolayı private olarak yazılmıştır. Ekstra olarak tester method yazılmıştır.

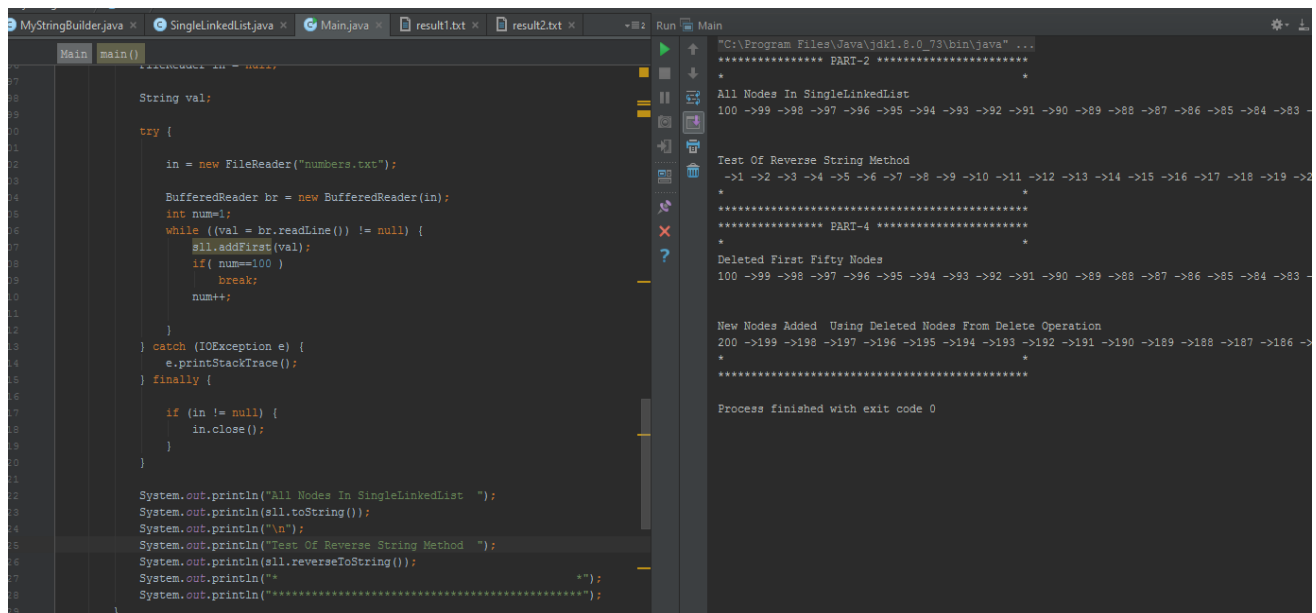
```
/**
 * Tester class
 * @return String
 */
public String reverseToString(){
    return reverseToString(head);
}

/**
 * Bu method String i recursive olarak tersten yazdırır.
 * @param node her seferde bi sonraki node verilir
 * @return String
 */
private String reverseToString(Node<E> node){
    if(node ==null){
        return "";
    }
    return reverseToString(node.next)+" ->"+node.getData();
}
```



6. Test Cases

reverseToString() method testi => 100 elemanlı singleLinklist in elemanlarının tersten sıralamasını göstermiştir.



```
MyStringBuilder.java x SingleLinkedList.java x Main.java x result1.txt x result2.txt x Run Main
Main main()
String val;
try {
    in = new FileReader("numbers.txt");
    BufferedReader br = new BufferedReader(in);
    int num=1;
    while ((val = br.readLine()) != null) {
        sll.addFirst(val);
        if( num==100 )
            break;
        num++;
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (in != null) {
        in.close();
    }
}

System.out.println("All Nodes In SingleLinkedList ");
System.out.println(sll.toString());
System.out.println("\n");
System.out.println("Test Of Reverse String Method ");
System.out.println(sll.reverseToString());
System.out.println("");
System.out.println("*****");
```

```
***** PART-2 *****
All Nodes In SingleLinkedList
100 ->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 ->49 ->48 ->47 ->46 ->45 ->44 ->43 ->42 ->41 ->40 ->39 ->38 ->37 ->36 ->35 ->34 ->33 ->32 ->31 ->30 ->29 ->28 ->27 ->26 ->25 ->24 ->23 ->22 ->21 ->20 ->19 ->18 ->17 ->16 ->15 ->14 ->13 ->12 ->11 ->10 ->9 ->8 ->7 ->6 ->5 ->4 ->3 ->2 ->1

Test Of Reverse String Method
-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 ->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 ->94 ->95 ->96 ->97 ->98 ->99 ->100

Deleted First Fifty Nodes
100 ->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 ->49 ->48 ->47 ->46 ->45 ->44 ->43 ->42 ->41 ->40 ->39 ->38 ->37 ->36 ->35 ->34 ->33 ->32 ->31 ->30 ->29 ->28 ->27 ->26 ->25 ->24 ->23 ->22 ->21 ->20 ->19 ->18 ->17 ->16 ->15 ->14 ->13 ->12 ->11 ->10 ->9 ->8 ->7 ->6 ->5 ->4 ->3 ->2 ->1

New Nodes Added Using Deleted Nodes From Delete Operation
200 ->199 ->198 ->197 ->196 ->195 ->194 ->193 ->192 ->191 ->190 ->189 ->188 ->187 ->186 ->185 ->184 ->183 ->182 ->181 ->180 ->179 ->178 ->177 ->176 ->175 ->174 ->173 ->172 ->171 ->170 ->169 ->168 ->167 ->166 ->165 ->164 ->163 ->162 ->161 ->160 ->159 ->158 ->157 ->156 ->155 ->154 ->153 ->152 ->151 ->150 ->149 ->148 ->147 ->146 ->145 ->144 ->143 ->142 ->141 ->140 ->139 ->138 ->137 ->136 ->135 ->134 ->133 ->132 ->131 ->130 ->129 ->128 ->127 ->126 ->125 ->124 ->123 ->122 ->121 ->120 ->119 ->118 ->117 ->116 ->115 ->114 ->113 ->112 ->111 ->110 ->109 ->108 ->107 ->106 ->105 ->104 ->103 ->102 ->101 ->100

Process finished with exit code 0
```

Part-3

1.System Requirement

Kendi AbstractCollection sınıfımızı implement ettik. **appendAnything()** methoduyla iki MyAbstractCollection objemizi concatenate etmek için implement edilmiştir.

appendAnything() methodunda AbstractCollection sınıfından override ettiğimiz **addAll()** methodu kullanılmaktadır.

addAll() methodunda kullanılan **add()** methodu implement edilmemistir.

```
public boolean appendAnything(MyAbstractCollection obj1, MyAbstractCollection obj2) {  
    obj1.addAll(obj2);  
    return false;  
}  
  
@Override  
public boolean addAll(Collection<? extends E> colle) {  
    boolean statu = false;  
    for (E e : colle)  
        if (add(e))  
            statu = true;  
    return statu;  
}
```

PART-4

1.System Requirement

Bu partta silinen Node ları bir ArrayList te tutup daha sonra yeni Node eklenirken bu silinen Node lar kullanılır. **deletedToString()** methoduyla silinen Node ları String sekilde gosterir.

3.Problem Solution Approach

Node lar silinecegi zaman once silinen node larin tutuldugu ArrayList e eklenir.

```
public E removeFirst() {  
    Node<E> temp = head;  
    if (head != null) {  
        head = head.next;  
    }  
    if (temp != null) {  
        deletedNodes.add(temp); //eklendi  
        size--;  
        return temp.data;  
    } else {  
        return null;  
    }  
}
```

Yeni Node eklenirken addFirst() methodunda önce daha önceden silinmiş node varsa o Node u kullanarak ekleme işlemi yapar.

```
public void addFirst(E item) {  
  
    if(deletedNodes.size()==0)  
        head=new Node(item,head);  
    else{  
        int lastIndex =deletedNodes.size()-1;  
  
        deletedNodes.get(lastIndex).data=item; //set nodes data fields (silinen node)  
        deletedNodes.get(lastIndex).next=head; (kullanilir)  
        head=deletedNodes.get(lastIndex);  
        deletedNodes.remove(lastIndex);  
    }  
  
    size++;  
}
```

6. Test Cases

Önce singlelinkedlist ten 50 Node silinmiştir.Daha sonra **addFirst()** methoduyla 100 eleman eklenmiştir.100 eleman ekleme işleminde 50 tane elemanın eklenmesi daha önce silinen 50 Node kullanılmıştır .

```
Test Part Four  
System.out.println("***** PART-4 *****");  
System.out.println("**  
SingleLinkedList sll =new SingleLinkedList();  
FileReader in = null;  
String val;  
  
try {  
  
    in = new FileReader("numbers.txt");  
  
    BufferedReader br = new BufferedReader(in);  
    int num=1;  
    while ((val = br.readLine()) != null) {  
        sll.addFirst(val);  
        if ( num==100 )  
            break;  
        num++;  
    }  
  
    for(int i=0;i<50;++i)  
        sll.removeFirst();  
    System.out.println("Deleted First Fifty Nodes " );  
  
    System.out.println(sll.deletedToString());  
    System.out.println("\n");  
    for(int i=100;i<200;++i)  
        sll.addFirst(String.valueOf(i));  
  
    System.out.println("New Nodes Added Using Deleted Nodes From Delete Operation");  
    //System.out.println("\n");  
    System.out.println(sll.toString());  
    System.out.println("**  
    System.out.println("*****");  
} catch (IOException e) {  
    e.printStackTrace();  
} finally {  
  
All Nodes In SingleLinkedList  
100 ->99 ->98 ->97 ->96 ->95 ->94 ->93 ->92 ->91 ->90 ->89 ->88 ->87 ->86 ->85 ->  
*  
Test Of Reverse String Method  
->1 ->2 ->3 ->4 ->5 ->6 ->7 ->8 ->9 ->10 ->11 ->12 ->13 ->14 ->15 ->16 ->17 ->18  
*  
***** PART-4 *****  
*  
Deleted First Fifty Nodes  
100 ->99 ->98 ->97 ->96 ->95 ->94 ->93 ->92 ->91 ->90 ->89 ->88 ->87 ->86 ->85 ->  
*  
New Nodes Added Using Deleted Nodes From Delete Operation  
200 ->199 ->198 ->197 ->196 ->195 ->194 ->193 ->192 ->191 ->190 ->189 ->188 ->187  
*  
*****  
Process finished with exit code 0
```


1. System Requirements

2. Use Case Diagrams

Add use case diagrams if required.

3. Class Diagrams

4. Other Diagrams

Add other diagrams.

5. Problem Solutions Approach

6. Test Cases