Sukkur IBA University
Department of Computer Science

DATA STRUCTURES
Lab01 – Arrays, LinkesLists

Instructor: Saif Hassan

READ IT FIRST

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don't waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

KEEP IT WITH YOU!

1. Indent your code inside the classes and functions. It's a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don't jump to the formula directly.

I, __**Amjad Ali**__ with student ID _**191-21-0001**_ Section __**'A'**__ hereby declare that I do understand the instructions above and follow them. This is

my own work.

Exercises

Task1 Description

Task01 (NLP)

Create a file named NLArray.java and design following functions:

- **`String [] wordTokenize (String fileName)`** → Read any text file and return list of words from that file. (Ignore . , : and all these types operators)
- **`String[] extractEmail (String fileName)`** → Read any text file and return all emails from and file

Note: Read about Natural Language Processing (NLP), Word Tokenizing, Stop Words, Information Extraction/Retrieval for Knowledge.

Solution:

```
1. package com.company;
2.
3. import java.io.File;
4. import java.io.FileNotFoundException;
5. import java.util.ArrayList;
6. import java.util.Arrays;
7. import java.util.Scanner;
8.
9. public class NLArray {
10.     public static ArrayList extractEmail (String filename)
11.     {
12.         ArrayList<String> words=new ArrayList<String>();
13.         String Content="";
14.         String word="";
15.         int index=-1;
16.         try {
17.
18.             File myObj = new File(filename);
19.             Scanner myReader = new Scanner(myObj);
20.
21.             while (myReader.hasNextLine()) {
22.                 Content=Content+myReader.nextLine();
23.             }
24.             myReader.close();
25.         }catch (Exception e) {
26.             System.out.println("File Not Found");
27.             String[] arr3={"Wrong","Directory"};
28.             return (ArrayList) Arrays.asList(arr3);
29.         }
30.
31.         if(Content.charAt(Content.length()-1)!=' ')
```

```
32.             Content=Content+" ";
33.
34.         for(int i=0;i<Content.length();i++)
35.         {
36.
37.             boolean con=true;
38.             if(Content.charAt(i)==' ' && word!="")
39.             {
40.
41.                 int length=word.length();
42.                 int at=word.indexOf('@');
43.                 int dot=-1;
44.                 if(at!=-1) {
45.                     for (int z = at; z < length; z++) {
46.                         if (word.charAt(z) == '.')
47.                             dot = z;
48.
49.                     }
50.
51.                     if (dot == -1) {
52.                         con = false;
53.                     } else {
54.                         if (at < 3)
55.                             con = false;
56.                         if (dot + 3 > length)
57.                             con = false;
58.                         if (at + 4 > dot)
59.                             con = false;
60.                     }
61.                 }
62.                 else
63.                 {
64.                     con=false;
65.                 }
66.                 if(con)
67.                     words.add(word);
68.                 word="";
69.             }
70.             else{
71.
72.                 word=word+Content.charAt(i);
73.
74.
75.             }
76.
77.
78.         }
79.         return  words;
80.
81.     }
82.
83.     public static String[] wordTokenize (String filename)
84.     {
85.         String Content="";
86.         try {
```

```
87.
88.             File myObj = new File(filename);
89.             Scanner myReader = new Scanner(myObj);
90.
91.             while (myReader.hasNextLine()) {
92.                 Content=Content+myReader.nextLine();
93.             }
94.             myReader.close();
95.         }catch (Exception e) {
96.             System.out.println("File Not Found");
97.             String[] arr={"Wrong","Directory"};
98.             return arr;
99.         }
100.
101.          if(Content.charAt(Content.length()-1)!=' ')
102.              Content=Content+" ";
103.
104.          int size=0;
105.          for(int i=0;i<Content.length();i++)
106.          {
107.              if(Content.charAt(i)==' ')
108.                 size++;
109.          }
110.          String Words[]=new String[size];
111.          String word="";
112.          int index=-1;
113.
114.          for(int i=0;i<Content.length();i++)
115.          {
116.
117.
118.              if(Content.charAt(i)==' ')
119.              {
120.
121.                 Words[++index]=word;
122.                  word="";
123.              }
124.              else{
125.
126.                  word=word+Content.charAt(i);
127.
128.
129.              }
130.
131.
132.          }
133.          return  Words;
134.
135.      }
136.
137.      //"D:\\Java Notes\\filename.txt"
138.      public static void main(String[] args) {
139.          Scanner sc=new Scanner(System.in);
140.          System.out.print("Insert The Complete Link of File e.f(D:\\\\Java
    Notes\\\\filename.txt): ");
```

```
141.          String link =sc.nextLine();
142.          String[] Words=wordTokenize(link);
143.          ArrayList<String> Words2=extractEmail(link);
144.          System.out.print("Which Method you want to call:- \n
   1)WordTokenSize\n      2)extractEmail\n       3)Exit\n Your Input: ");
145.          int input =sc.nextInt();
146.          if(input==1)
147.          {
148.              for(String word:Words)
149.                  System.out.println(word);
150.          }
151.          else if(input==2)
152.          {
153.              for(int i=0;i<Words2.size();i++)
154.                  System.out.println(Words2.get(i));
155.          }
156.          else if(input==3)
157.          {
158.              System.out.println("Have A Good Day");
159.              System.exit(0);
160.          }
161.          else{
162.
163.              System.out.println("Wrong Input ");
164.              System.exit(0);
165.
166.          }
167.
168.
169.          }
170.
171.      }
```

Sample Input:

```
Insert The Complete Link of File e.f(D:\\Java Notes\\filename.txt): D:\\Java Notes\\filename.txt
Which Method you want to call:-
     1)WordTokenSize
      2)extractEmail
       3)Exit
 Your Input: 1
```

Sample Output

```
Child
labour
is
a
term
you
might
have
heard
about
in
news
or
movies.It
refers
to
a
crime
where
```

```
Insert The Complete Link of File e.f(D:\\Java Notes\\filename.txt): D:\\Java Notes\\filename.txt
Which Method you want to call:-
     1)WordTokenSize
      2)extractEmail
       3)Exit
 Your Input: 2
ahsan@iba-suk.ed.pk
amjadaligolo477@gmail.com
amjad.bsais@iba-suk.edu.pk
hyder-@gmail.com
```

Task2 Description

Task02 (Image Cropping)

Design following methods in above same class NLArray.java.

- **`void extractBoundaries (int arr[][])`** → This function should extract boundaries and print from arr (Boundaries include 1st row, 1st col, last row, last col).

- **`void cropCenterPart (int arr[][])`** → This function should extract center part and print from arr, center part includes everything except Boundaries (Boundaries include 1st row, 1st col, last row, last col).

Solution:

```
1. package com.company;
2.
3. import java.util.Random;
4. import java.util.Scanner;
5.
6.
7.
8.
9. public class NLArray2 {
10.      public static boolean NConRep (int arr[][])
11.      {
12.          boolean cond=true;
13.          for(int i=0;i< arr.length;i++)
14.          {
15.              for(int j=0;j<arr[0].length;j++)
16.              {
17.                  if(arr[i][0]!=arr[i][j])
18.                      cond=false;
19.
20.              }
21.              if(cond)
22.                  return true;
23.
24.              cond=true;
25.          }
26.
27.
28.          return false;
29.      }
30.      public static void extractBoundaries (int arr[][])
31.      {
```

```
32.          int frow[]=new int[arr[0].length];
33.          int lrow[]=new int [arr[0].length];
34.          int fcol[]=new int[arr.length-2];
35.          int lcol[]=new int[arr.length-2];
36.          if(arr.length<=2 || arr[0].length<=2)
37.          {
38.              System.out.println(" \n     Output will be:- \n");
39.              for(int i=0;i<arr.length;i++)
40.              {
41.                  for(int j=0;j<arr[0].length;j++)
42.                  {
43.                      System.out.print(arr[i][j]+" ");
44.                  }
45.                  System.out.println();
46.              }
47.
48.          }
49.          else
50.          {
51.              System.out.println(" \n     Output will be:- \n");
52.
53.              for(int i=0;i<arr[0].length;i++)
54.              {
55.                  frow[i]=arr[0][i];
56.                  lrow[i]=arr[arr.length-1][i];
57.
58.              }
59.              int index1=-1;
60.              int index2=-1;
61.              for(int i=1;i<arr.length-1;i++)
62.              {
63.                  fcol[++index1]=arr[i][0];
64.                  lcol[++index2]=arr[i][arr[0].length-1];
65.              }
66.
67.              for(int ele:frow)
68.                  System.out.print(ele+" ");
69.
70.              System.out.println();
71.              for(int i=0;i<fcol.length;i++)
72.              {
73.                  System.out.print(fcol[i]);
74.                  for(int s=0;s<arr[0].length-2;s++)
75.                      System.out.print("  ");
76.                  //if(arr[0].length!=3)
77.                      System.out.print(" ");
78.
79.                  System.out.print(lcol[i]);
80.                  System.out.println();
81.              }
```

```
82.
83.             for(int ele:lrow)
84.                 System.out.print(ele+" ");
85.
86.         }
87.
88.
89.     }
90.
91.     static void cropCenterPart(int arr[][])
92.     {
93.         if(arr.length<=2 || arr[0].length<=2)
94.         {
95.             System.out.println("The Given Array Can't be processed");
96.         }
97.         else
98.         {
99.             System.out.println(" \n     Output will be:- \n");
100.            for(int i=1;i<arr.length-1;i++)
101.            {
102.                System.out.print(" ");
103.                for(int j=1;j<arr[0].length-1;j++)
104.                {
105.                    System.out.print(arr[i][j]+" ");
106.                }
107.                System.out.println();
108.            }
109.        }
110.    }
111.
112.    public static void main(String[] args) {
113.        Scanner sc =new Scanner(System.in);
114.        Random rand=new Random();
115.        System.out.print("input num of Row: ");
116.        int row=sc.nextInt();
117.        System.out.print("input num of colomn: ");
118.        int colomn=sc.nextInt();
119.        int[][] arr=new int[row][colomn];
120.        for(int i=0;i<row;i++)
121.        {
122.
123.            for(int j=0;j<colomn;j++)
124.            {
125.                int a= rand.nextInt(10);
126.
127.                arr[i][j]=a;
128.                System.out.print(arr[i][j]+" ");
129.            }
130.            System.out.println();
131.        }
```

```
132.
133.        System.out.print("Which Method you want to call:- \n
   1)ExtractBoundaries\n      2)CropCenterPart\n
   3)CheckforConsective\n        4)Exit\n          Your Input: ");
134.        int input =sc.nextInt();
135.        if(input==1)
136.        {
137.            extractBoundaries(arr);
138.        }
139.        else if(input==2)
140.        {
141.            cropCenterPart(arr);
142.        }
143.        else if(input==3)
144.        {
145.            System.out.print("Result = ");
146.            System.out.println(NConRep(arr));
147.        }
148.        else if(input==4)
149.        {
150.            System.out.println("Have A Good Day");
151.            System.exit(0);
152.        }
153.        else {
154.
155.            System.out.println("Wrong Input ");
156.            System.exit(0);
157.
158.        }
159.
160.
161.
162.
163.
164.
165.
166.    }
167. }
```

Sample Input:

```
input num of Row: 8
input num of colomn: 8
7 2 9 7 0 4 3 0
8 7 1 8 5 0 3 1
8 9 2 8 6 7 9 4
9 7 7 4 4 8 4 9
9 9 3 1 9 9 2 2
5 5 3 6 8 8 2 6
5 5 4 0 2 3 3 8
2 4 0 9 3 7 2 1
Which Method you want to call:-
     1)ExtractBoundaries
      2)CropCenterPart
       3)CheckforConsective
        4)Exit
          Your Input: 1
```

Sample Output

```
Which Method you want to call:-
     1)ExtractBoundaries
      2)CropCenterPart
       3)CheckforConsective
        4)Exit
          Your Input: 2

     Output will be:-

 6 8 7 0 0 6
 7 4 6 3 2 2
 5 3 3 5 8 3
 5 1 7 5 5 0
 8 9 0 8 7 0
 0 4 5 9 1 9
```

```
Which Method you want to call:-
     1)ExtractBoundaries
      2)CropCenterPart
       3)CheckforConsective
        4)Exit
          Your Input: 1

     Output will be:-

1 7 7 2 5 3 3 8
3             6
9             0
5             2
4             3
2             4
1             4
3 0 6 3 9 2 7 6
```

Task3 Description

Task03 (Determining N consecutive same values)

Design following methods in above same class NLArray.java.

boolean NConRep (int arr[][]) → This function would return True if N consecutive values are same otherwise false. Check examples:

2	1	3	5
22	22	22	22
12	41	88	53
57	8	74	4

N is 4 so matrix is 4x4 and in 4 consecutive values are same so it should return True.

Solution:

```
1. public static boolean NConRep (int arr[][])
2.     {
3.         boolean cond=true;
4.         for(int i=0;i< arr.length;i++)
5.         {
6.             for(int j=0;j<arr[0].length;j++)
7.             {
8.                 if(arr[i][0]!=arr[i][j])
9.                     cond=false;
10.
11.              }
12.              if(cond)
13.                  return true;
14.
15.              cond=true;
16.          }
17.
18.
19.          return false;
20.     }
```

Sample Input:

```
input num of Row: 6
input num of colomn: 6
1 5 9 8 6 7
4 6 6 2 9 2
9 7 2 4 0 4
1 5 1 0 9 2
6 4 2 1 2 7
1 4 9 7 0 7
Which Method you want to call:-
     1)ExtractBoundaries
      2)CropCenterPart
       3)CheckforConsective
        4)Exit
          Your Input: 3
```

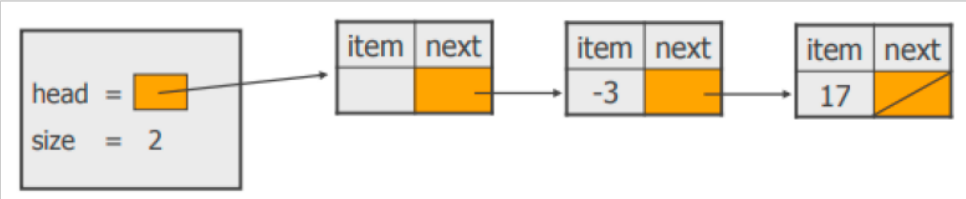
Sample Output

```
input num of Row: 6
input num of colomn: 6
1 5 9 8 6 7
4 6 6 2 9 2
9 7 2 4 0 4
1 5 1 0 9 2
6 4 2 1 2 7
1 4 9 7 0 7
Which Method you want to call:-
     1)ExtractBoundaries
      2)CropCenterPart
       3)CheckforConsective
        4)Exit
          Your Input: 3
Result = false
```

Task04 (Linked Lists)

In this task you will write a program that implements a variant of a linked list. This variant has a dummy node pointed to by the head link as shown in the following figure:

Linked list with a dummy first node:



This trick will allow your code to be a little simpler, not requiring a special case for add or remove operations. Your constructor method will be:

```
public LinkedList(){

      head = new Node(null);

      size = 0;

}
```

You need to write a class called `LinkedList` that implements the following `List` interface:

```
// a list interface

public interface List {

 public boolean isEmpty();
      // returns true if the list is empty, false otherwise

      public int size();
      // returns the number of items in the list
       public void add(Object item);
      // adds an item to the list
      // item is added at the end of the list
       public void add(int index, Object item); //
      adds an item to the list at the given index //
      item is added at the given index; // the
      indices start from 1.

      public void remove(int index);
      // removes the item from the list that has the given index
       public void remove(Object item);
      // removes an item from the list
      // removes the first item in the list whose equal method matches
      // that of the given item

      public List duplicate();
```

```
        // creates a duplicate of the list
        // returns a copy of the linked list

        public List duplicateReversed();
        // creates a duplicate of the list with the nodes in reverse order
        // returns a copy of the linked list with the nodes in reverse order
}
```

In addition to the interface, your `LinkedList class` needs to implement a `toString()` method that prints the list in the format

[size: the_size_of_the_list - item1, item2, ...]

Specifications, notes, and hints

Your program needs to meet the following specifications:

- **Submit the file `LinkedList`.java and additional files if applicable. Your `Node class` should be an inner class within the `LinkedList` class. Make sure your `class` implements the `interface` as specified, i.e. your `class` should begin with `public class LinkedList implements List`.**
- **When commenting your code use `Javadoc` style comments at the beginning of each method.**
- **Put comments at the top of the file (`Java` File) with your name, S_ID, S_Name, date and course, and a short (one or two line) description of what the program does. Make sure your code runs on machine.**

Submit your `source code` files via the classroom by the due date (remember the course syllabus for the late policy).

Solution:

1) List Interface

```
1. package com.company;
2. public interface List {
3.
4.     public boolean isEmpty();
5.     // returns true if the list is empty, false otherwise
6.
7.
8.     public int size();
9.     // returns the number of items in the list
10.
11.
12.    public void add(int item);
13.    // adds an item to the list
```

```
14.     // item is added at the end of the list
15.
16.
17.     public void add(int index, int item);
18.    // adds an item to the list at the given index
19.    // item is added at the given index;
20.    // the indices start from 1.
21.
22.     public void removeIndex(int index);
23.     // removes the item from the list that has the given index
24.
25.     public void remove(int item);
26.     // removes an item from the list
27.     // removes the first item in the list whose equal method matches that of
    the given item
28.
29.     public List duplicate();
30.     // creates a duplicate of the list
31.     // returns a copy of the linked list
32.
33.     public List duplicateReversed();
34. // creates a duplicate of the list with the nodes in reverse order
35. // returns a copy of the linked list with the nodes in reverse order
36.
37. }
```

2) Linked List Class

```
1. package com.company;
2.
3.
4.
5. public class Linked_List implements List{
6.
7.      private int size=0;
8.      private Node Head;
9.
10.
11.       public class Node{
12.           int data;
13.           Node next;
14.
15.           Node(int data)
16.           {
17.               this.data=data;
18.
19.           }
20.       }
21.       public void incSize(){
```

```
22.           size++;
23.       }
24.
25.
26.       public boolean isEmpty()
27.       {
28.           return Head==null;
29.       }
30.
31.       public void add(int data)
32.       {
33.           Node newNode=new Node(data);
34.           if(isEmpty())
35.           {
36.               Head=newNode;
37.           }
38.           else{
39.               newNode.next=Head;
40.               Head =newNode;
41.              /* Node Current=Head;
42.               while(Current.next!=null)
43.               {
44.                   Current=Current.next;
45.               }
46.               Current.next=newNode;*/
47.
48.           }
49.           size++;
50.
51.       }
52.
53.       @Override
54.       public int size() {
55.
56.           return size;
57.       }
58.
59.       @Override
60.       public void add(int index, int item) {
61.
62.           if(index>size)
63.               System.out.println("index Out of Bound");
64.           else{
65.               int i=1;
66.               Node current=Head;
67.               while(i<index)
68.               {
69.                   current=current.next;
70.                   i++;
71.               }
```

```
72.                 current.data=item;
73.                 System.out.println("Successfully Added");
74.             }
75.         }
76.
77.         @Override
78.         public void removeIndex(int index) {
79.             if(size<index)
80.                 System.out.println("Index out of bound");
81.             else if(index==1)
82.             {
83.                 Head=Head.next;
84.             }
85.             else{
86.                 int i=1;
87.                 Node current=Head;
88.                 while(i+1!=index)
89.                 {
90.                     current=current.next;
91.                     i++;
92.                 }
93.                 current.next=current.next.next;
94.                 System.out.println("Successfully Removed");
95.                 size--;
96.             }
97.
98.         }
99.
100.        @Override
101.        public void remove(int item) {
102.            if (size == 0) {
103.                System.out.println("List Is Empty.");
104.            } else {
105.                boolean cond = false;
106.                Node Current = Head;
107.                Node oneBackCurrent = Head;
108.                while (Current.next != null) {
109.
110.                    if (Current.data==item) {
111.                        cond = true;
112.                        break;
113.
114.                    }
115.                    oneBackCurrent = Current;
116.                    Current = Current.next;
117.
118.
119.
120.                }
121.                if (Current.data==item)
```

```
122.                 cond = true;
123.
124.             if(Head.data==item)
125.             {
126.                 Head=Head.next;
127.                 size--;
128.             }
129.             else if (cond) {
130.                 oneBackCurrent.next= oneBackCurrent.next.next;
131.                 System.out.println("Succesfully Removed");
132.                 size--;
133.             } else {
134.                 System.out.println("No Such Element In the List");
135.             }
136.         }
137.     }
138.         public List duplicateReversed()
139.         {
140.             Linked_List list=new Linked_List();
141.             Node Current=Head;
142.             for(int i=0;i<size;i++)
143.             {
144.
145.                 list.add(Current.data);
146.                 Current=Current.next;
147.             }
148.
149.
150.             return list;
151.
152.         }
153.
154.         public List duplicate()
155.         {
156.             Linked_List list=new Linked_List();
157.             Node Current=Head;
158.             for(int i=0;i<size;i++)
159.             {
160.                 Node newNode=new Node(Current.data);
161.
162.                 if(list.isEmpty()) {
163.                     list.Head = newNode;
164.                 }
165.                 else{
166.                     Node Check=list.Head;
167.                     while(Check.next!=null)
168.                     {
169.                         Check=Check.next;
170.                     }
171.                     Check.next=newNode;
```

```
172.
173.
174.                 }
175.                 list.incSize();
176.                 Current=Current.next;
177.
178.             }
179.
180.             return list;
181.         }
182.
183.         public void print()
184.         {
185.             if(isEmpty())
186.             {
187.                 System.out.println("List is Empty");
188.
189.             }
190.             else{
191.                Node Current=Head;
192.                while(Current.next!=null)
193.                {
194.                    System.out.print(Current.data+", ");
195.                    Current=Current.next;
196.                }
197.                 System.out.println(Current.data);
198.
199.             }
200.         }
201.
202.     public String toString()
203.     {
204.         String Str ="[ Size:("+size+")-->";
205.         Node Current=Head;
206.         while(Current.next!=null)
207.         {
208.             Str+=Current.data+", ";
209.             Current=Current.next;
210.         }
211.         Str+=Current.data+" ]";
212.
213.         return Str;
214.     }
215.
216.     }
```

2) Linked List Demo Class

```
1. package com.company;
2.
3. public class Linked_ListDemo {
4.     public static void main(String[] args) {
5.         Linked_List list=new Linked_List();
6.         list.add(56);
7.         list.add(78);
8.         list.add(35);
9.         list.add(85);
10.          list.add(29);
11.          list.add(23);
12.          list.add(22);
13.          list.add(34);
14.          System.out.println("Initially"+list);
15.          list.add(3,100);
16.          list.add(5,99);
17.          System.out.println("After Replacement"+list);
18.          List list1,list2;
19.          list.remove(34);
20.          System.out.println(list);
21.          list1=list.duplicate();
22.          list2=list.duplicateReversed();
23.          System.out.println("Duplicate"+list1);
24.          System.out.println("Reversed "+list2);
25.
26.
27.
28.     }
29. }
```

Sample Input:

```
Linked_List list=new Linked_List();
list.add(56);
list.add(78);
list.add(35);
list.add(85);
list.add(29);
list.add(23);
list.add(22);
list.add(34);
System.out.println("Initially"+list);
list.add( index: 3, item: 100);
list.add( index: 5, item: 99);
System.out.println("After Replacement"+list);
List list1,list2;
list.remove( item: 34);
System.out.println(list);
list1=list.duplicate();
list2=list.duplicateReversed();
System.out.println("Duplicate"+list1);
System.out.println("Reversed "+list2);
```

Sample Output

```
Initially[ Size:(8)-->34, 22, 23, 29, 85, 35, 78, 56 ]
Successfully Added
Successfully Added
After Replacement[ Size:(8)-->34, 22, 100, 29, 99, 35, 78, 56 ]
[ Size:(7)-->22, 100, 29, 99, 35, 78, 56 ]
Duplicate[ Size:(7)-->22, 100, 29, 99, 35, 78, 56 ]
Reversed [ Size:(7)-->56, 78, 35, 99, 29, 100, 22 ]
```