Lab 2 :

Implement an ArrayList class using given design contract.

ArrayList (<T>) : Standard array based implementation of an ArrayList

private : T array ([]) data container
size is the # of elements in this ArrayList
public : constant capacity is the initial capacity of the ArrayList container

ArrayList() : Constructs an ArrayList object so that integer array is empty and size is 0
/*! alters: self and size
*! ensures: self =<> and size = 0
*/

ArrayList(T[] other) : Constructs an ArrayList using specified array *other*.
/*! alters: self and size
*! requires: other != <>
*! ensures: self =<#other> and size = other.length
*/

void clear() : resets this ArrayList back to empty
/*! alters: self and size
*! ensures: self =<> and size = 0
*/

boolean equals(ArrayList other) : equals returns true if elements in this ArrayList are equal to elements in *other* regardless of their order, and self.size == other.size, otherwise returns false.
/*! preserves: self and other
*! ensures: self == other
*/

void addFront(T x) : add x to the front of this ArrayList
/*! alters: self and size
*! consumes: x
*! ensures: self =<#x> + <#self> and size++
*/

void addBack(T x) : add x to the back of this ArrayList
/*! alters: self and size
*! consumes: x

```
  *! ensures: self = <#self>  + <#x> and size++
*/

void removeFront() :  remove the first element of this ArrayList
/*! alters: self and size
  *! requires: self != <>
  *! ensures: self =<#first> - <#self> and size--
*/

void removeBack( ) : remove the last element of this ArrayList
/*! alters: self and size
  *! requires: self != <>
  *! ensures: self = <#self>  - <#last> and size--
*/

String toString() : returns the content of this ArrayList as a String, ex: <A, B, C>
/*! preserves: self
  *! requires: self != <>
  *! ensures:  String = "<#self>"
*/

ArrayList subList(fromindex, toindex) : returns another ArrayList which is a portion of this
ArrayList between specified fromindex, inclusive and toindex exclusive.
/*! preserves: self
  *! requires: self != <> and (fromindex >= 0 || toindex < size) and fromindex < toindex
  *! ensures:  another =  <#self>[fromindex, toindex) {anotherList containing current values of
self from fromindex to toindex-1} and another.size = |another|
*/

boolean isEmpty(): returns true if this ArrayList contains no elements
/*! preserves: self
  *! requires: self != <>
  *! ensures:  true if self != <> otherwise false
*/
```

Explanation of terms:

<> - empty list
[] - an array
<#x> - current value of x
<#self> - current values of self
+ join
- disjoin
++ addition
- - subtraction

|| cardinality

Write a main method to test all the methods in your ArrayList class. You can score extra credit if you perform thorough tests. You must write pre/post conditions of all methods. Turn in your Java source programs, and outputs in a PDF document via Assignments/Lab2.

Here is a starter code!

```
public class ArrayList<T> { //Using Generics
    private T[] array;
    private int size;
    public static final int capacity = 10;

    public ArrayList(){
        size = 0;
        T[] array = (T[])(new Object[capacity]); //Due to use of
Generics
    }
    public ArrayList(T[] A){
        size = A.length;
        array = A;
    }

    public static void main(String[] args) {
        ArrayList<Integer> al = new ArrayList<>();
        Integer [] ia = { 3, 5, 9, 1};
        ArrayList<Integer> other = new ArrayList<>(ia);
        System.out.println(al); // requires implementation of
toString method
        System.out.println(other); // requires implementation of
toString method
    }
}
```