People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'hamed Bouguerra - Boumerdes
Institute of Electrical and Electronics Engineering
Electronics Department

Option: Computer engineering

LAB REPORT n°4

November 29, 2022

Title

EE423: Advanced Programming/ Working with the class String

Authored by :
Agli Wafa
Zidane Aymen

Instractor :
Dr.A Zitouni

Session : 2022/2023

Contents

Introduction Within this lab, we will learn about the String class, which is provided by Java. With this class we easily manipulate chains of characters. It has a set of predefined functions that we can use to solve many problems related to stings.

Tools and Software:

1. A PC with ECLIPSE IDE V8.
2. Online LaTeX Editor for writing the report.

1 Exercise 01: *Palindrome*

```
public static boolean onlyAlphabets(String s) { //method to check if onlyAlphabets
    for (int i = 0; i < s.length(); i++) {
      if(!((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') || (s.charAt(i) == ' ')))
        return false;
    }
    return true;
  }

  public static String findPalindrome(String s) {
    String new_s = s.toLowerCase(); //lowerCase all the string
    boolean check = onlyAlphabets(new_s); //calling onlyAlphabets(String s)
    if(!check) return "Only alphabetical characters are allowed"; //check==false
    if(new_s.length() < 3) return "The String is too short";
    int index_min = 0, index_max = 0, longest = 0; //vars for storing best palind
    String temp, f_temp; //temp for string, f_temp for string without spaces
    boolean isPalindrome;
    for(int i = 0; i <= new_s.length() - 3; i++) {
      if (new_s.charAt(i) == ' ') continue; // ignore space
      for(int j = i+3; j <= new_s.length(); j++) {
        isPalindrome = true; f_temp = "";
        temp = new_s.substring(i,j);
        for (int k = 0; k < temp.length(); k++) {
          if(temp.charAt(k) == ' ')
            continue;
          else
            f_temp += temp.charAt(k);
        } //ftemp == temp without space
        for(int start = 0,end = f_temp.length() - 1; start <= end; start ++, end --) {//
     if not read in opposite order isPalindrome = false
          if(f_temp.charAt(start) != f_temp.charAt(end)) {
            isPalindrome = false;
            break;
          }
        }
        //isPalindrome ==true and higher thn longest
        if(isPalindrome && (j-i) > longest) {
          index_min = i; //save the beggening
          index_max = j;//save the end
          longest = (j-i); //save the length
          System.out.print(longest);
        }
      }
```

```
42      }
43      if(longest == 0)
44        return "No palindrome found";
45      else
46        return s.substring(index_min, index_max);
47    }
48
49    public static void main(String[] args) {
50      // TODO Auto-generated method stub
51      System.out.println(findPalindrome("ok")); // The string is too short
52      System.out.println(findPalindrome("ok!"));// Only alphabetical characters are
       allowed
53      System.out.println(findPalindrome("coucou")); // No palindrome found
54      System.out.println(findPalindrome("good morning Madam")); // Madam
55      System.out.println(findPalindrome("nurses run")); // nurses run (notice the spaces)
56      System.out.println(findPalindrome("madam good morning Madam")); // madam
57      System.out.println(findPalindrome("room llevell morning refer")); // om llevell mo
58    }
```

Results and illustrative example:

i	j	IsPalindrome	Save longest

i=0	J=3 ==>nur
	J=4 ==>nurs
	J=5 ==>nurse
	J=6 ==>nurses
	J=7 ==>nurses
	J=8 ==>nursesr
	J=9 ==>nursesru
	J=10 ==>nursesrun
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = true
			i = 0;
			j = 10;
			j-i = 10
			longest = 10;

i=1	J=4 ==>urs
	J=5 ==>urse
	J=6 ==>urses
	J=7 ==>urses
	J=8 ==>ursesr
	J=9 ==>ursesru
	J=10 ==>ursesrun
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = true
		Palindrome = false
			i= 1;
			j = 9;
			j-i = 8;
			longest =10

i=2	J=5 ==>rse
	J=6 ==>rses
	J=7 ==>rses
	J=8 ==>rsesr
	J=9 ==>rsesru
	J=10 ==>rsesrun
		Palindrome = false
		Palindrome = false
		Palindrome = false
		Palindrome = true
		Palindrome = false
		Palindrome = false
			i= 2;
			j = 8;
			j-i = 6;
			longest =10

i=3	J=6 ==>ses
	J=7 ==>ses
	J=8 ==>sesr
	J=9 ==>sesru
	J=10 ==>sesrun
		Palindrome = true
		Palindrome = true
		Palindrome = false
		Palindrome = false
		Palindrome = false
			i= 3;
			j = 7;
			j-i = 4;
			longest =10

i=4	J=7 ==>es J=8 ==>esr J=9 ==>esru J=10 ==>esrun	Palindrome = false Palindrome = false Palindrome = false Palindrome = false	longest =10
i=5	J=8 ==>sr J=9 ==>sru J=10 ==>srun	Palindrome = false Palindrome = false Palindrome = false	longest =10
i=6	continue		
i=7	J=10 ==>run	Palindrome = false	longest =10

			Palindrome = nurses run
			index_min = 0;
			index_max = 10;
			longest = (j-i)=10

2 Exercise 02: *Sorting*

```java
public static String [] sortStrings(String[] ar){ //bubble sort
      String temp; String[] arr = ar.clone();//to save original array's version
      for (int j = 0; j < arr.length - 1; j++){
        for (int i = j + 1; i < arr.length; i++){
          if (arr[j].compareTo(arr[i]) > 0){
            //swap
            temp = arr[j];
            arr[j] = arr[i];
            arr[i] = temp;
          }
        }
      }
        return arr;
    }
    public static void main(String[] args)
    {
      String[] arr = { "bold", "a","Sort","string" };
      String[] arr2 = sortStrings(arr);
    System.out.println(Arrays.toString(arr2));//[Sort, a, bold, string]
    }
```

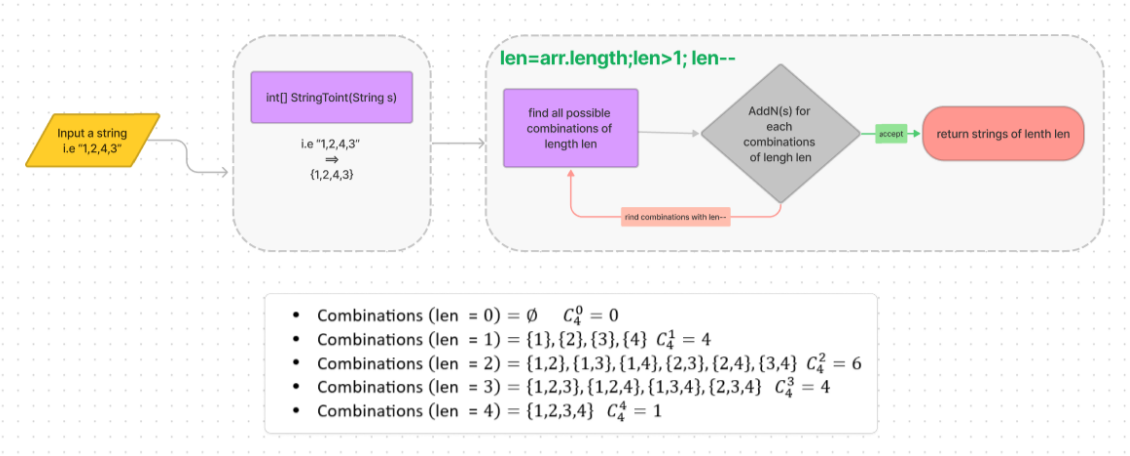
illustrative example:

bold	a	Sort	string
a	bold	Sort	string
Sort	bold	a	String
Sort	**a**	**bold**	**String**

Nb: In order to check our results we compared our output with the *Arrays.Sort(String[] s)* method

3 Exercise 3 : *Longest chain*

3.1 Code flowchart



3.2 Code

```
public static int[] StringToint(String s) { //change String to an int[]
    String[] numberStrs = s.split(",");
    int[] numbers = new int[numberStrs.length];
    for(int i = 0;i < numberStrs.length;i++){
      numbers[i] = Integer.parseInt(numberStrs[i]);
    }
    return numbers;
    }

static String addN(int[] array, int n) {
    int[] array2 = array.clone();
    String s="("+array2[0]+",";
    for(int i=1;i<array2.length;i++) {
    if(array2[i] >= array2[i-1]) { //exemple [1,2]
      if(i==array2.length-1) {
        s=s+array2[i]+")";
      }else {
        s=s+array2[i]+",";
      }
    }else if((n-(array2[i-1]-array2[i])>=0 && (array2[i]+(array2[i-1]-array2[i]) >=
    array[i-1]) ) ) {//i still have rest from n and i-1 > i exemple [2,1] n =3
        if(i==array2.length-1) {
          s=s+array2[i]+"+"+(array2[i-1]-array2[i])+")";
        }else {
          s=s+array2[i]+"+"+(array2[i-1]-array2[i])+",";
        }
        n = n-(array2[i-1]-array2[i]);
        array2[i] = array2[i]+(array2[i-1]-array2[i]);

      }
      if((n-(array2[i-1]-array2[i]))<0)return null; //if n is over before the end dont
    accept the combination
```

```
    }
    return s;
  }

 static void combination(int arr[], int data[], int start,int end, int index, int r, int
    n,List<String> list){//create combination of length r
    if (list.size()>0) return; //if i already accept once no need to continue
    if (index==r) //stop recurssion when r is reached
    {
       if (addN(data, n)!=null) {add only accepted Strings
         list.add(addN(data, n));
       }
       return ;
    }
    for (int i=start; i<=end && end-i+1 >= r-index; i++)
    {
      data[index] = arr[i];
      combination(arr, data, i+1, end, index+1, r,n,list);
    }
  }

static List<String>  longestIncrementalChain(int[] arr,int nn)
  {

    int n = arr.length; int r = arr.length;
    List<String> list =new ArrayList<String>();
        //start check from length of array (longest)
    for(r=arr.length;r>1; r--) {
    int data[]=new int[r];
    combination(arr, data, 0, n-1, 0, r,nn,list);
    }
    return list;
  }
```

```
public static void main (String[] args) public static void main (String[] args) {
    String arr1 = "5,0,1,7";
    System.out.println("longestIncrementalChain(\"5,0,1,7\", 0);               "+
    longestIncrementalChain(StringToint(arr1),0));
    String arr2 = "5,0,1,7";
    System.out.println("longestIncrementalChain(\"5,0,1,7\", 9);               "+
    longestIncrementalChain(StringToint(arr2),9));
    int arr3[] = "10,1,9,4,10,0,6,7";
    System.out.println("longestIncrementalChain(\"10,1,9,4,10,0,6,7\", 0);     "+
    longestIncrementalChain(StringToint(arr3),0));
    int arr4[] = "8,7,6,5,5";
    System.out.println("longestIncrementalChain(\"8,7,6,5,5\", 0);             "+
    longestIncrementalChain(StringToint(arr4),0));
    int arr5[] = "8,7,6,5,5";
    System.out.println("longestIncrementalChain(\"8,7,6,5,5\", 5);             "+
    longestIncrementalChain(StringToint(arr5),5));

  }
```

A Appendix

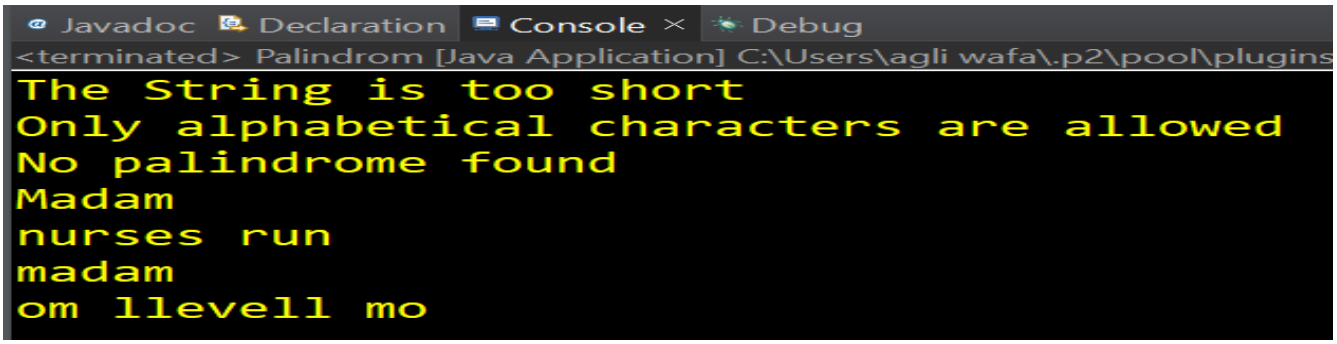


Figure 1: Exercise 01 execution : Palindrom.

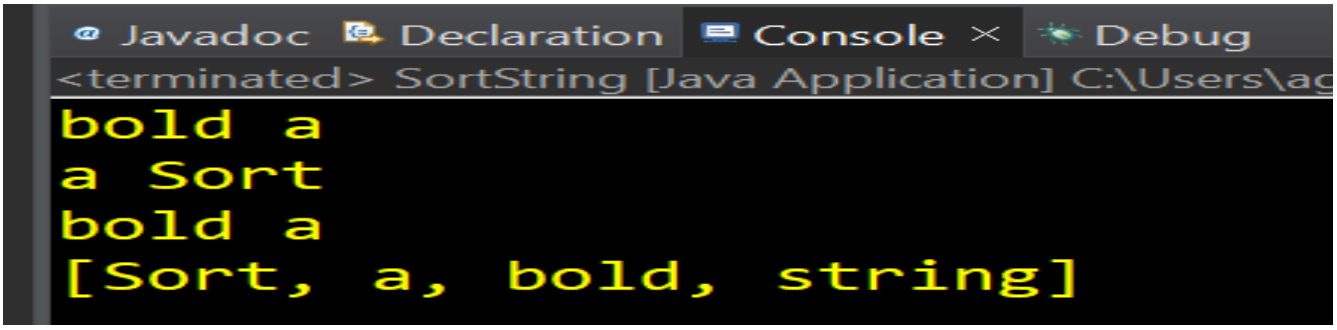


Figure 2: Exercise 02 execution : Sort.

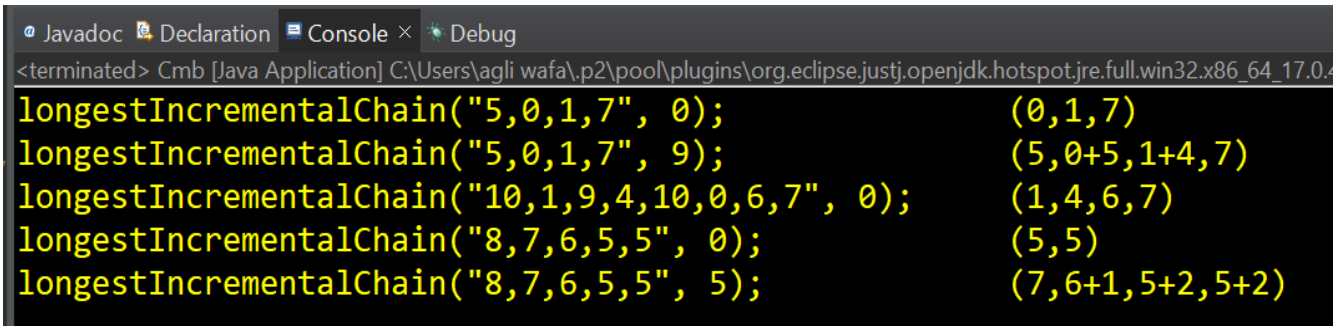


Figure 3: Exercise 03 execution : LongestIncrementalChain.