==========String_Test==========

```
package com.String_Function;

public class String_Fucntion {
	// Ok
	// test function is used to test the function input in ok and xx form
	public final static void test(String get , String expected){

		try {
			// condition used for result
			if (get.equalsIgnoreCase(expected)){
				System.out.print("OK"+"  ");
			}// condition used for reject the result
			else{
				System.out.print("XX"+"  ");
			}
		} catch (Exception e) {
			// TODO Auto-generated catch block
			System.out.println("System not working ");;
		}

	}
	//Ok
    // donuts and send back to the main for test
	public final static String donuts(int count){

		try {
			// Condition for donuts
			if(count == 10 || count > 10){
				// if the count is equal the 10 and grater than the 10
				return "Number of donuts: many";
			}else{
				// if the count is less than the 10 than print
				return "Number of donuts: "+count;
			}
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			return null;
		}

	}
	//Ok
	// both_ends and send back to the main for test
	public final static String both_ends(String string){
		//
		try {
		   // if the length of the String is one
		   if(string.length() <= 1 || string.isEmpty()){
		       return "";
		    }else{
		      // this return back the both_end String
		      return string.substring(0,2)+""+ new StringBuffer(new
StringBuffer(string).reverse().substring(0, 2)).reverse();
		    }

		} catch (Exception e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        return null;
                }

        }
        //Ok
        // fix_start and send back to the main for test
        public final static String fix_start(String string){
                //
                //
                try {
                        //
                        if(string.isEmpty()){
                                return null;
                        }
                        // get the temp as String to get the string who value is less than one
the real string
                        String temp = string.substring(1);
                        // return the real fix_start String to the test
                        return string.charAt(0)+temp.replaceAll(string.charAt(0)+"", "*");
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        return null;
                }

        }
        // Ok
        // mix and send back to the main for test
        public final static String mixUp(String string, String string2){

                try {
                        // return the mix string and test it
                        return string.replace(string.subSequence(0, 2), string2.subSequence(0,
2))+" "+string2.replace(string2.subSequence(0, 2), string.subSequence(0, 2));
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        return null;
                }
        }
        //Ok
        // verbing and send back to the main for test
        public final static String verbing(String string){
                //

                try {
                        // if the length of the String is less than 3 than not change the
string and send same back
                        if(string.length() < 3){
                                // send the same string that i received for the parameter
                                return string;
                        }else{
                                // if the string is ing than return back the same
                                if(string.contains("ing") && string.length() == 3){
                                   // this one used when it containing the ing in the text
                                       return string;
```

```
                        }else if(string.contains("ing")){
                                // if true than concat the old string with ly
                                return string+"ly";
                        }else{
                                // if flase than concat the onl string with ing
                                return string+"ing";
                        }
                }

        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                return null;
        }
    }
    // Ok
    // not bad and send back to the main for test
    public final static String not_bad(String string){
        //
        try {
                // condition for not and bad contains in the string
                if(string.contains("not") && string.contains("bad")){
                        // if yes than watch not is first come and bad is lass
                        if(string.indexOf("not") < string.indexOf("bad")){
                                // return back to the main
                                return
string.replace(string.substring(string.indexOf("not")), "good");
                        // if the bad is first and not is lass than is will work
                        }else if(string.indexOf("not") > string.indexOf("bad")){
                                // than send the same string that get from the parameter
                                // send the same the string
                                return string;
                        }
                }else{
                        // send the same the sting
                        // if only not and only bad contains in the string
                        return string;
                }
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                return null;
        }
        return null;
    }
    //Ok
    // front back and send back to the main for test
    public final static String front_back(String string , String string2){
        // condition for both equal length
        if(string.length()%2 == 0 && string2.length()%2 == 0){
                // send back to the main
                return string.substring(0,
string.length()/2)+string2.substring(0,string2.length()/2)+string.substring(string.length(
)/2)+string2.substring(string2.length()/2);

        }else{
                // condition for first even and second odd
                if(string.length()%2 == 0 && string2.length()%2 == 1){
```

```
                        // send back to the mian
                        return string.substring(0,
string.length()/2)+string2.substring(0,string2.length()/2+1)+string.substring(string.lengt
h()/2)+string2.substring(string2.length()/2+1);
                  }else{
                        // this one work when both string are odd
                        // send back to the main
                        return string.substring(0,
string.length()/2+1)+string2.substring(0,string2.length()/2+1)+string.substring(string.len
gth()/2+1)+string2.substring(string2.length()/2+1);
                  }
            }

      }
      //
      //
      public static void main(String args[]){
            //
            System.out.println ("donuts");
            // Each line calls donuts, compares its result to the expected for that call.
            test(donuts(4), "Number of donuts: 4");
            test(donuts(9), "Number of donuts: 9");
            test(donuts(10), "Number of donuts: many");
            test(donuts(99), "Number of donuts: many");
            //----------------------------------------
            // Each line calls both_end, compares its result to the expected for that
call.
            System.out.println("\nboth_ends");
            test(both_ends("spring"), "spng");
            test(both_ends("Hello"), "Helo");
            test(both_ends("a"), "");
            test(both_ends("xyz"), "xyyz");
            //----------------------------------------
            //Each line calls fix_start, compares its result to the expected for that
call.
            System.out.println("\nfix_start");
            test(fix_start("babble"), "ba**le");
            test(fix_start("aardvark"), "a*rdv*rk");
            test(fix_start("google"), "goo*le");
            test(fix_start("donut"), "donut");
            //----------------------------------------
            //Each line calls mixUp, compares its result to the expected for that call.
            System.out.println("\nmix_up");
            test(mixUp("mix", "pod"), "pox mid");
            test(mixUp("dog", "dinner"), "dig donner");
            test(mixUp("gnash", "sport"), "spash gnort");
            test(mixUp("pezzy", "firm"), "fizzy perm");
            //----------------------------------------
            //Each line calls verbing, compares its result to the expected for that call.
            System.out.println("\nverbing");
            test(verbing("hail"), "hailing");
            test(verbing("swiming"), "swimingly");
            test(verbing("do"), "do");
            test(verbing("ing"),"ing");
            //----------------------------------
            //Each line calls not_bad, compares its result to the expected for that call.
            System.out.println("\nnot_bad");
            test(not_bad("This movie is not so bad"), "This movie is good");
```

```
		test(not_bad("This dinner is not that bad!"), "This dinner is good");
		test(not_bad("This tea is not hot"), "This tea is not hot");
		test(not_bad("It's bad yet not"), "It's bad yet not");
		//------------------------------------
		//Each line calls front_back, compares its result to the expected for that
call.
		System.out.println("\nfront_back");
		test(front_back("abcd", "xy"), "abxcdy");
		test(front_back("abcde", "xyz"), "abcxydez");
		test(front_back("Kitten", "Donut"), "KitDontenut");
	}


}
```

========== ListTest ==========

```
package com.String_Function;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;

//
public class List_Reddal {

	//Ok
	// test the both String and List
	public final static void test(Object get , Object expected){

	   try {
			// condition used for result
			if (get.toString().equalsIgnoreCase(expected.toString())){
				System.out.print("OK"+"  ");
			}// condition used for reject the result
			else{
				System.out.print("XX"+"  ");
			}
		} catch (Exception e) {
		   //
			System.out.println("System not working ");;
		}

	}
	// Ok
	// test the match ends and send back to the main
	public final static Object match_ends(Object match_e[]){


		try {
			// count the word
			int count_end = 0;
			// get the single word
```

```
                Object get_word;
                // break the word into the single char but the type is Object
                Object temp = null,temp2 = null;
                // used the loop to access the array index
                for(Object match_key : match_e){
                        // assign the word to the single object
                        get_word = match_key.toString();
                        // condition for null or empty String
                        if(!get_word.equals("")){
                                // get the first single letter but the type is Object
                                temp = get_word.toString().charAt(0);
                                // get the last single letter but the type is Object
                           temp2 = get_word.toString().charAt(get_word.toString().length()-1);
                           // condition for both first and end are equal if true it count else not
                           if(temp.toString().equalsIgnoreCase(temp2.toString())){
                                count_end += 1;
                           }
                         }
                }
              // return the Result of the Process
              return count_end;
             // catch used to handle the exception
        } catch (Exception e) {
                // if the exception is rise it show the message of the error
                return "Indext out of Range The Error are found in method match_End";

        }
}
// Ok
// test the front_x and send back to the main
public final static Object front_x(Object f_x[]){
        // get the two array
        // array with_x
        ArrayList<Object> array_with_xs = new ArrayList<Object>();
        // array with_out x
        ArrayList<Object> array_without_xs = new ArrayList<Object>();
        // loop for adding the item individual
        for (Object object : f_x) {
                // condition for front_X
                if((object.toString().charAt(0) == 'x') || (object.toString().charAt(0) == 'X')){
                        // add the front_x
                        array_with_xs.add(object);

                }else{
                        // add the non_Front_x
                        array_without_xs.add(object);
                }
        }
        // sort the both the array
```

```
                Collections.sort(array_with_xs,null);
                //
                Collections.sort(array_without_xs,null);
                //
                // know Combine the both array,s into the array_with_xs
                for (Object object : array_without_xs ) {
                        array_with_xs.add(object);
                }
                // send array for test
                return array_with_xs.toString();
        }
        // Ok
        // test the sort_last and send back to the main
        public final static Object sort_last(Object array_sort_last_1[][]){
                // Method work only Integer
                // get the temp array for single index change
                Object temp1[];
                // loop's work like a sort-ing
                // outer-loop
           for(int i = 0; i < array_sort_last_1.length-1; i++) {
                // inner-loop
                   for (int j = i+1; j < array_sort_last_1.length; j++) {
                           // condition watch
                        // (Integer) casting required to convert the object element into the Integer
                                if(((Integer) array_sort_last_1[i][array_sort_last_1[i].length-1] > (Integer)
array_sort_last_1[j][array_sort_last_1[j].length-1])){
                                        // sewap_ing apply_ing on the address of the array not on item
                                        temp1 = array_sort_last_1[i];
                                        array_sort_last_1[i] = array_sort_last_1[j];
                                        array_sort_last_1[j] = temp1;
                                }
                        }
                }
           // send array for test
                return Arrays.deepToString(array_sort_last_1);
        }
   // Ok
        // test the remove_adjacent and send back to the main
        public final static Object remove_adjacent(Object r_adjacent[]){
                // get the temp array and pass the all element
                // of the first array into the temp
      ArrayList<Object> temp_Array = new ArrayList<Object>();
      // loop used to get the single item at one time
                for (Object object : r_adjacent) {
                        // if it contains the object into the array it
                        //become false and not item more add to the temp
                        if(!(temp_Array.contains(object))){
                                // add teh single object to the temp_Array
                                temp_Array.add(object);
```

```
                }
            }
            // return the array as the type of string
            return temp_Array.toString();
        }
// Ok
        // test the linear_merge and send back to the main
        public final static Object linear_merge(Object list1[], Object list2[]){
            //
          // merge mean combine to list into single list
            ArrayList<Object> object = new ArrayList<Object>();
            // add the first list
            for (Object object1 : list1) {
                // condition for front_X
                object.add(object1);
            }
            // add the second list
            for (Object object2 : list2) {
                // condition for front_X
                object.add(object2);
            }
            // sort the list by use of the Collection
            Collections.sort(object,null);
            return object.toString();
        }
        //
// Main Method
        public static void main(String[] args) {

            //Ok
            // Match_end problem
          System.out.println("Match_End");
          // array 1 match_ends
          Object array[] = {"xxa",121, "aa", "x", "bbb"};
          test(match_ends(array), 4);
          // array 2 match_ends
          Object array1[] =  {"", "xyw", "xy", "xyx", "xx"};
     test(match_ends(array1), 2);
     // array 3 match_ends
      Object array2[] = {"aaA", "be", "abc", "hello"};
     test(match_ends(array2), 1);
     //-------------------------------------------------
     // Ok
          // Sort problem
          System.out.println("\nSort_last");
          // array,s 1 sort_last
          Object array_sort_last_1[][] = { {1, 3}, {3, 2}, {2, 1}};
          Object array_sort_last_1_1[][] = { {2, 1}, {3, 2}, {1, 3}};
          test(sort_last(array_sort_last_1), Arrays.deepToString(array_sort_last_1_1));
```

```
        // array,s 2 sort_last
        Object array_sort_last_2[][] = {{2, 3},{1, 2},{3, 1}};
        Object array_sort_last_2_2[][] = {{3, 1}, {1, 2}, {2, 3}};
        test(sort_last(array_sort_last_2), Arrays.deepToString(array_sort_last_2_2));
        // array,s 3 sort_last
        Object array_sort_last_3[][] = {{1, 7}, {1, 3}, {3, 4, 5}, {2, 2}};
        Object array_sort_last_3_3[][] = {{2, 2}, {1, 3}, {3, 4, 5}, {1, 7}};
        test(sort_last(array_sort_last_3), Arrays.deepToString(array_sort_last_3_3));
//-------------------------------------------------

        // Ok
        // Front_end problem
        System.out.println("\nFront_X");
        // array,s 1 front-x
        Object array_front_x1[] = {"bbb", "ccc", "axx", "xzz", "xaa"};
        Object array_front_x1_1[] = {"xaa", "xzz", "axx", "bbb", "ccc"};
        test(front_x(array_front_x1), Arrays.toString(array_front_x1_1));
        // array,s 2 front-x
        Object array_front_x2[] = {"ccc", "bbb", "aaa", "xcc", "xaa"};
        Object array_front_x2_2[] = {"xaa", "xcc", "aaa", "bbb", "ccc"};
        test(front_x(array_front_x2), Arrays.toString(array_front_x2_2));
        // array,s 3 front-3
        Object array_front_x3[] = {"mix", "xyz", "apple", "xanadu", "aardvark"};
        Object array_front_x3_3[] = {"xanadu", "xyz", "aardvark", "apple", "mix"};
        test(front_x(array_front_x3), Arrays.toString(array_front_x3_3));
        //-------------------------------------------------

        // Ok
        // Remove adjacent problem
        System.out.println( "\nremove_adjacent");
        // array,s 1 r_adjacent
        Object r_adjacent[] = {1, 2, 2, 3};
        Object r_adjacent1[] = {1, 2, 3};
 test(remove_adjacent(r_adjacent), Arrays.toString(r_adjacent1));
 // array,s 2 adjacent
 Object r_adjacent2[] = {2, 2, 3, 3, 3};
 Object r_adjacent2_2[] = {2, 3};
 test(remove_adjacent(r_adjacent2), Arrays.toString(r_adjacent2_2));
 // array,s 3 adjacent
 Object r_adjacent3[] = {};
 Object r_adjacent3_3[] = {};
 test(remove_adjacent(r_adjacent3), Arrays.toString(r_adjacent3_3));
 //-------------------------------------------------

 // Ok
 // linear merge Problem
 System.out.println("\nlinear_merge");
 // array,s 1 linear_merge
 Object temp_list1[] = {"aa", "xx", "zz"};
```

```
Object temp_list1_1[] = {"bb", "cc"};
Object temp_list1_1_1[] = {"aa", "bb", "cc", "xx", "zz"};
test(linear_merge(temp_list1, temp_list1_1), Arrays.toString(temp_list1_1_1));
// array,s 2 linear_merge
Object temp_list2[] = {"aa", "xx"};
Object temp_list2_1[] = {"bb", "cc", "zz"};
Object temp_list2_1_1[] = {"aa", "bb", "cc", "xx", "zz"};
test(linear_merge(temp_list2, temp_list2_1), Arrays.toString(temp_list2_1_1));
// array,s 3 linear_merge
Object temp_list3[] = {"aa", "aa"};
Object temp_list3_1[] = {"aa", "bb", "bb"};
Object temp_list3_1_1[] = {"aa", "aa", "aa", "bb", "bb"};
test(linear_merge(temp_list3, temp_list3_1), Arrays.toString(temp_list3_1_1));
//------------------------------------------------------------
//-----------------------Best of Luck-----------------------

}
   }
```