ArrayOps.java

```java
public class ArrayOps {
    public static void main(String[] args) {

    }

    public static int findMissingInt (int [] array) {
        // Write your code here:
        int missing = array.length;
        for (int i = 0; i < missing; i++) {
            boolean found = false;
            for (int j = 0; j < array.length; j++) {
                if (array[j] == i){
                    found = true;
                    break;
                }
            }
            if (!found) {
                missing = i;
                break;
            }
        }
        return missing;
    }

    public static int secondMaxValue(int [] array) {
        int max = array[0];
        int second_max = array[0];
        for (int i = 0; i < array.length; i++) {
            if (array[i] > max)
                max = array[i];
        }
        int maxCounter = 0;
        for(int i = 0; i < array.length; i++) {
            if (array[i] == max)
                maxCounter++;
            if (array[i] > second_max && (array[i] != max || maxCounter > 1))
                second_max = array[i];

        }
        return second_max;
    }

    public static boolean containsTheSameElements(int [] array1,int [] array2) {
        // Write your code here:
        boolean containsSame = true;
        for (int i = 0; i < array1.length; i++) {
            boolean found = false;
            for (int j = 0; j < array2.length; j++) {
                if (array1[i] == array2[j]) {
                    found = true;
                    break;
                }
            }
            if (!found) {
                containsSame = false;
                break;
            }
```

```
        }
        return containsSame;
    }

    public static boolean isSorted(int [] array) {
        // Write your code here:
        if (array.length < 3)
             return true;
        boolean sorted = true;
        for (int i = 1; i < array.length-1; i++) {
            if (array[i-1] < array[i] && array[i] > array[i+1]){
                sorted = false;
                break;
            }
            else if (array[i-1] > array[i] && array[i] < array[i+1]) {
                sorted = false;
                break;
            }
        }
        return sorted;
    }

}

```

StringOps.java

```java
public class StringOps {
    ////////////////////////////////////////////////////////
    //////                                            //////
    //////              Reminder:                     //////
    //////        allowed methods                     //////
    //////                                            //////
    //////        1.charAt(int index)                 //////
    //////        2.length()                          //////
    //////        3.substring(int start)              //////
    //////        4.substring(int start,int ends)     //////
    //////        5.indexOf(String str)               //////
    //////                                            //////
    //////        The rest are not allowed !          //////
    //////        if you want to use a different      //////
    //////        method, and you can implement       //////
    //////        it using material from the course   //////
    //////        you need to implement a version of  //////
    //////        the function by yourself.           //////
    //////                                            //////
    //////        see example for substring           //////
    //////        in Recitation 3 question 5          //////
    //////                                            //////
    ////////////////////////////////////////////////////////
    public static void main(String[] args) {

    }

    public static String capVowelsLowRest (String string) {
        // Write your code here:
        String vowels = "aeiouAEIOU";
        String newString = "";
        for (int i = 0; i < string.length(); i++) {
            boolean foundVowel = false;
            for (int j = 0; j < vowels.length(); j++) {
                if (vowels.charAt(j) == string.charAt(i)) {
                    foundVowel = true;
                    break;
                }
            }
            if (foundVowel && string.charAt(i) > 96)
                newString += (char) (string.charAt(i) - 32);

            else if (!foundVowel && string.charAt(i) < 91 && string.charAt(i) != ' ')
                    newString += (char) (string.charAt(i) + 32);

            else
                newString += (char) (string.charAt(i));
        }
        return newString;
    }

    /* Gets a string, returns the string with no spaces at its start*/
    public static String noSpacesAtStart(String string) {
        if (string.length() > 0) {
            int index = 0;
            while (string.charAt(index) == 32) {
                index++;
```

```
58                  if (index >= string.length())
59                      break;
60              }
61              if (index < string.length())
62                  return string.substring(index);
63          }
64          return "";
65      }
66
67      /* Gets a string, returns the string with a lower case version of the first word in
    the string*/
68      public static String doLowerCaseUnitilSpace(String string) {
69          String newString = "";
70          int i = 0;
71          while (i < string.length() && string.charAt(i) != ' ') {
72              if(string.charAt(i) < 91)
73                  newString += (char) (string.charAt(i) + 32);
74
75              else
76                  newString += string.charAt(i);
77              i++;
78          }
79          return newString;
80      }
81
82      /* Gets a string, returns the string without the first word in it */
83      public static String jumpToNewWord(String string) {
84          String newString = "";
85          int i = 0;
86          while (i < string.length() && string.charAt(i) != ' '){
87              i++;
88          }
89          if (i < string.length())
90              newString = string.substring(i);
91          newString = noSpacesAtStart(newString);
92          return newString;
93      }
94
95      public static String camelCase (String string) {
96          String newString = "";
97          String newSubString = noSpacesAtStart(string);
98          if(newSubString.charAt(0) < ('Z' + 1))
99              newString += (char) (newSubString.charAt(0) + 32);
100         else
101             newString += newSubString.charAt(0);
102         newString += doLowerCaseUnitilSpace(newSubString.substring(1));
103         newSubString = jumpToNewWord(newSubString);
104         while(newSubString.length() > 1) {
105             if(newSubString.charAt(0) > 96)
106                 newString += (char) (newSubString.charAt(0) - 32);
107
108             else
109                 newString += newSubString.charAt(0);
110             newString += doLowerCaseUnitilSpace(newSubString.substring(1));
111             newSubString = jumpToNewWord(newSubString);
112         }
113         return newString;
114     }
115
116     public static int[] allIndexOf (String string, char chr) {
```

```
117         int counter = 0;
118         for(int i = 0; i < string.length(); i++) {
119             if (string.charAt(i) == chr)
120                 counter++;
121         }
122         int[] allIndexOf = new int[counter];
123         int foundCounter = 0;
124         for (int i = 0; i < string.length(); i++) {
125             if (string.charAt(i) == chr) {
126                 allIndexOf[foundCounter] = i;
127                 foundCounter++;
128             }
129         }
130         return allIndexOf;
131     }
132 }
133
```