ArrayOps

findMissingInt

```
 public static int findMissingInt (int [] array) {
        int sum = 0;
        int arraySum=0;
        //sum the numbers that expected to be in the array
(include the missing)
        for (int i = 0; i <= array.length; i++)
        {
            sum = sum + i;
        }
        // sum the numbers that actually in the array.
        for(int j = 0; j < array.length; j++){
        arraySum = arraySum + array[j];
        }
        //find the missing number and return
        int missing = sum - arraySum;
        return missing;
    }
```

secondMaxValue

```java
public static int secondMaxValue(int [] array) {
        int max = array[0];
        int [] temp = new int[array.length - 1];
        int secondMaxValue;
        // Put in new array all numbers in the original except
the max.
        for (int i = 1; i < array.length; i++){
            if (array [i] > max)
            {
                temp [i-1] = max;
                max = array [i];
            }
            else
            temp [i-1] = array [i];
        }
        secondMaxValue = temp [0];
        //check the max number in the new array, which is the
secind max in the original array.
        for(int j = 0; j < temp.length; j++){
            if (temp [j] >secondMaxValue)
                secondMaxValue = temp [j];
        }

        return secondMaxValue;
    }
```

containsTheSameElements

```
public static boolean containsTheSameElements(int []
array1,int [] array2) {

    boolean sameSet = false;
    //check that all elements in array 1 appear in array 2, if
not rerurn false.
        for (int i = 0; i < array1.length; i++){
        sameSet = false;
            for(int j = 0; j < array2.length; j++){
                if (array1[i] == array2 [j])
                sameSet = true;

            }
            if (sameSet == false)
            return sameSet;
        }
        //check that all elements in array 2 appear in array
1, if not rerurn false.
        for (int k = 0; k < array2.length; k++){
        sameSet = false;
            for(int l = 0; l < array1.length; l++){
                if (array2[k] == array1 [l])
                sameSet = true;

            }
            if (sameSet == false)
            return sameSet;
        }
        //return true if 2 arrays contains The Same Elements
        return sameSet;
    }
```

isSorted

```java
 public static boolean isSorted(int [] array) {
        // dedine 2 boolean to check sorting , one for
increase and one for decrease
        boolean sortedMinMax = true;
        boolean sortedMaxMin = true;
        // check if all elements is sorted increase or
decrease. true if all elements sorted.
        for (int i = 0; i < array.length - 1; i++){
            if (array [i] > array [i+1])
                sortedMinMax = false;
            if (array [i] < array [i+1])
                sortedMaxMin = false;
        }
        // return true if it sorted (one of increase or
decrease is true).
        boolean sorted = sortedMinMax || sortedMaxMin;

        return sorted;
    }
```

StringOps

capVowelsLowRest

```
public static String capVowelsLowRest (String string) {
        // define vowels array, new empty string and boolean
to check  letter is vowel
        char[] vowels = {'a','e','i','o', 'u',
'A','E','I','O', 'U'};
        String result = "";
        boolean isVowel = false;
        for (int i = 0; i < string.length(); i++){
            char convertChar = string.charAt(i);
            isVowel = false;
            // check for each char if is vowel
            for (int j = 0; j < vowels.length; j++){

                 if (convertChar == vowels [j])
                {
                    isVowel = true;
                }
            }
            // low all letters that not vowel
            if (!isVowel)
            {
                if (convertChar >= 65 && convertChar <=90)
                {
                convertChar = (char)(convertChar + 32);
                result = result + convertChar;
                }
                else
                {
                result = result + convertChar;
                }
            }
            // all vowel letters change to uppercase
            if (isVowel)
            {
                if (convertChar>90)
                {
                convertChar = (char)(convertChar - 32);
                result = result + convertChar;
                }
                else
                result = result + convertChar;
            }
        }
      return result;
    }
```

camelCase

```
public static String camelCase(String string) {
    //define empty string to return, and space parameters for
seperate between words.
    String camelCase = "";
    int space = 0;
    int spaceindex = 0;
    for (int i = 0; i < string.length(); i++){
        char letter = string.charAt(i);
        //On the first word only.
        if (space == 0)
        {
            if (letter!=32)
        {
        // all letters in first word in low case
            if (letter >= 65 && letter <=90)
                {
                letter = (char)(letter + 32);
                }
                camelCase = camelCase + letter;

        }
        // check if the first word end.
            if (letter == 32)
            {
                if (camelCase.length() != 0)
                space++;
                spaceindex = i;
            }
        }
        // from the second word.
        else
        {
            if (letter ==32)
                spaceindex = i;
        // the first letter on each word (after space) with
uppercase.
            else if (letter!=32 && i == spaceindex + 1)
            {
                if (letter > 90){
                    letter = (char)(letter - 32);
                }
            camelCase = camelCase + letter;
            }
            // other letters in the word with low case.
            else if (letter!=32 && i != spaceindex + 1){
                if (letter >= 65 && letter <=90)
                {
```

```
                letter = (char)(letter + 32);
                }
            camelCase = camelCase + letter;

            }
        }
    }
    //return the new string
            return camelCase;
    }
```

allIndexOf

```java
 public static int[] allIndexOf (String string, char chr) {
        int CountIndex = 0;
        // count the number that the char appear in the string
        for (int i = 0; i < string.length(); i++){
            char temp = string.charAt(i);
            if (chr == temp)
            {
                CountIndex++;
            }
        }
        // define array with the size of the number index the
char appear.
        int [] allIndexOf = new int[CountIndex];
        // put in the array all indicies which the char appear
in the string.
        for (int j = 0; j < allIndexOf.length; j++){
            for (int k = 0; k < string.length(); k++){
            char temp = string.charAt(k);
            if (chr == temp)
            {
                allIndexOf [j] = k;
                j++;
            }

            }
        }
        // retrn the array
        return allIndexOf;
    }
```