Java Program

Write a program to reverse number

```
public static void main(String[] args) {
int n=123456,  reverse = 0;
while (n != 0) {
reverse = reverse * 10;
reverse = reverse + n % 10;
n = n / 10;
}
System.out.println("Reverse of the number is " +
reverse);
}
```

OUTPUT: Reverse of the number is 654321

Prime Number Program in Java

```
public static void primeNumber() {
int count = 0,num=7;
for (int i = 2; i < num / 2; i++) {
if (num % 2 == 0) {
break;
} }
if (count == 0) {
System.out.println("Given number is prime");
} else {
System.out.println("Given numbewr is not prime");
} }
```

Write a program for palindrome number

```
public static void pallindrone() {
int n = 1331,  temp, reverse = 0;
temp = n;
while (n != 0) {
reverse = reverse * 10;
reverse = reverse + n % 10;
n = n / 10;
}
if (temp == reverse) {
System.out.println("palindrone number::" + reverse);
} else {
System.out.println("the given number is not
palindrone");  } }
```

Write a program for palindrome string

```
static void pallindroneString() {
String str = "abbbba", temp, rev = "";
temp = str;
for (int i = str.length() - 1; i >=0; i--) {
rev = rev + str.charAt(i);
}
if (temp.equals(rev)) {
System.out.println("String is Pallindrone");
} else {
System.out.println("String is not Pallindrone"); }      }
```

Write a program for reverse string

```
static void reverseString() {
String str = "I am Natha Rathod",  rev = "";
for (int i = str.length() - 1; i >=0; i--) {
rev = rev + str.charAt(i);
}
if (str.equals(rev)) {
System.out.println("String is reversed:"+rev);
} else {
System.out.println("String is not reversed:"+rev); } }
```

Swap of 2 no. without using third variable.

```
static void swappingOf2Number() {
int a=10,b=20;
a=a+b;
b=a-b;
a=a-b;
System.out.println("A="+a+" and "+"B="+b);
}
```

Write a program count the repeated characters in string using java

```
static void repeatedCharCountInString() {
String str = "Natha Rathod";
int count = 0;
for (int i = 0; i < str.length(); i++) {
for (int j = i + 1; j < str.length(); j++) {
if (str.charAt(i) == str.charAt(j)) {
count++;
break;
} } }
SOP("Reapeted charater without Array::" + count);}
```

OUTPUT: Reapeted charater without Array::4

```
static void repeatedCharCountInStringArray() {
String str = "Natha Rathod";
int count = 0;
char[] revStr = str.toCharArray();
for (int i = 0; i < str.length(); i++) {
for (int j = i + 1; j < str.length(); j++) {
if (revStr[i] == revStr[j]) {
count++;
break;
} } }
SOP("Reapeted charater Using Array::" + count);}
```

OUTPUT: Reapeted charater without Array::4

Find highest Array and sorted

```
public static void largestArray() {
        int arr[] = { 10, 20, 70, 80, 50, 30 };
        int max = arr[0];
        for (int i = 1; i < arr.length; i++) {
        if (arr[i] > max) {
        max = arr[i];
        }
        }
System.out.println("Maximum array::" + max);
}
```

```
public static void sortedArray() {
        int arr[] = { 60, 10, 40, 20, 70, 80, 50, 30 };
        int temp;
        for (int i = 0; i < arr.length; i++) {
        for (int j = i + 1; j < arr.length; j++) {
        if (arr[i] > arr[j]) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;} } }
        System.out.println("Ascending order");
        for (int i = 0; i < arr.length; i++) {
        System.out.println(arr[i]); } }
```

Duplicate Array

```
public static void DuplicateArrayList() {
        int[] arr = new int[] { 1, 2, 3, 4, 2, 7, 8, 8, 3 };
        for (int i = 0; i < arr.length; i++) {
        for (int j = i + 1; j < arr.length; j++) {
        if (arr[i] == arr[j])
    System.out.println("Duplicate elements in given
        array:"+ arr[j]);
        }}}
```

O/P :

```
Duplicate elements:2
Duplicate elements:3
Duplicate elements:8
```

```
public static void CountDuplicateArray(){
        int arr[] = { 10, 20, 5, 20, 10, 10, 20, 5, 20 };
Map<Integer, Integer> mp = new HashMap<Integer, Integer>();
        for (int i = 0; i < arr.length; i++){
        if (mp.containsKey(arr[i])) {
        mp.put(arr[i], mp.get(arr[i]) + 1);
        } else {
        mp.put(arr[i], 1);}}
        // Traverse through map and print frequencies
 for (Map.Entry<Integer, Integer> entry : mp.entrySet()) {
   System.out.println("Element:"+entry.getKey() + " " +
        "Count:"+entry.getValue());}}
```

O/P:

```
Element:20 Count:4
Element:5 Count:2
Element:10 Count:3
```

Reverse String but maintain original space of String

```
String str = "I am using Selenium WebDriver";
char[] inputArray = str.toCharArray();
char[] result = new char[inputArray.length];
// Mark spaces in result
for (int i = 0; i < inputArray.length; i++) {
if (inputArray[i] == ' ') {
result[i] = ' ';
}}
// Traverse input string from beginning and put characters in result from end
int j = result.length - 1;
for (int i = 0; i < inputArray.length; i++) {
// Ignore spaces in input string
if (inputArray[i] != ' ') {
// ignore spaces in result.
if (result[j] == ' ') {
j--;
}
result[j] = inputArray[i];
j--;
}}
System.out.println("Reverse String maintain Space:"+ String.valueOf(result));}
```

O/P : Reverse String maintain Space: r ev irDbe Wmuinele Sgnisumal

Java program to count the occurrence of each character in a string using Hash Map

```
String str = "Natorao";
Map<Character, Integer> map = new HashMap<Character, Integer>();

// checking each char of strArray

for (int i = str.length() - 1; i >= 0; i--) {

if (map.containsKey(str.charAt(i))) {

int count = map.get(str.charAt(i));

// If char is present in charCountMap,incrementing it's count by 1

map.put(str.charAt(i), ++count);

} else {// If char is not present in charCountMap putting this char to charCountMap with 1 as it's value

map.put(str.charAt(i), 1);}}

System.out.println(map);

// Printing the character Count Map

for (Map.Entry entry : map.entrySet()) {
```

System.*out*.println(entry.getKey() + " " + entry.getValue()); }

OUTPUT: {a=2, r=1, t=1, N=1, o=2}
a 2
r 1
t 1
N 1
o 2

Java Program to Find the Occurrence of Words in a String using HashMap

// Declaring the String

String str = "Alice is girl and Alice is boy";

// Declaring a HashMap of <String, Integer>

Map<String, Integer> hashMap = **new** HashMap<String, Integer>();

// Splitting the words of string and storing them in the array.

String[] words = str.split(" ");

for (String word : words) {

// Asking whether the HashMap contains the key or not. Will return null if not.
Integer integer = hashMap.get(word);

if (integer == **null**)

// Storing the word as key and its occurrence as value in the HashMap.

hashMap.put(word, 1);

else {

// Incrementing the value if the word is already present in the HashMap.

hashMap.put(word, integer + 1); } }

System.*out*.println(hashMap); }}

O/P: {Alice=2, and=1, is=2, girl=1, boy=1}

OR

```
String str = "Alice is girl and Alice is boy";
Map<String, Integer> hashMap = new HashMap<String, Integer>();
String[] words = str.split(" ");
for (String word : words) {
// containsKey(key) will return a boolean value i.e. true if it contains the key and false if it doesn't.
if (hashMap.containsKey(word))
hashMap.put(word, hashMap.get(word) + 1);
else
hashMap.put(word, 1);
}
System.out.println(hashMap);
}
```

O/P: {Alice=2, and=1, is=2, girl=1, boy=1}

Program to determine whether two strings are the anagram

Two Strings are called the anagram if they contain the same characters. However, the order or sequence of the characters can be different.

```
public static void bothStringAnagram() {
        String str1 = "Brag";
        String str2 = "Grab";
    // Converting both the string to lower case.
        str1 = str1.toLowerCase();
        str2 = str2.toLowerCase();
    // Checking for the length of strings
        if (str1.length() != str2.length()) {
System.out.println("Both the strings are not anagram");
        } else {
    // Converting both the arrays to character array
        char[] string1 = str1.toCharArray();
        char[] string2 = str2.toCharArray();
    // Sorting the arrays using in-built function sort ()
        Arrays.sort(string1);
        Arrays.sort(string2);
// Comparing both the arrays using in-built function equals ()
        if (Arrays.equals(string1, string2) == true) {
System.out.println("Both the strings are anagram");
        } else {
System.out.println("Both the strings are not anagram");}}}
```

Output: Both the strings are anagram.

public static void **printStar**(**int** n) {
for (**int** i = 0; i < n; i++) {
for (**int** j = i; j < n; j++) { System.**out**.print(" *"); }
System.**out**.println(" "); } }
	OUTPUT:
	* * * * *
	* * * *
	* * *
	* *
	*

public static void **printStarFrom1**(**int** n) {
int i, j;
//outer loop to handle number of rows n in this case
for (i = 0; i < n; i++) {
/ inner loop to handle number of columns, values changing acc. to outer loop
for (j = 0; j <= i; j++) {/ printing stars
System.***out***.print("* ");}
/ ending line after each row
System.***out***.println();}}
	OUTPUT:
	
	*
	* *
	* * *
	* * * *
	* * * * *

```
public static void printReverseStar(int n) {
int i, j;
/outer loop to handle number of rows n in this case
for (i = 0; i < n; i++) {
/inner loop to handle number spaces,/ values changing acc. to requirement
for (j = 2 * (n - i); j >= 0; j--) {/ printing spaces
System.out.print(" ");}
/inner loop to handle number of columns
/values changing acc. to outer loop
for (j = 0; j <= i; j++) {/ printing stars
System.out.print("* ");}
/ending line after each row
System.out.println();}}
```

OUTPUT

```
          *
        * *
      * * *
    * * * *
  * * * * *
* * * * * *
 * * * * * * *
  * * * * * * * *
   * * * * * * * * *
    * * * * * * * * * *
```

```
public static void printTriagle(int n){
/outer loop to handle number of rows n in this case
for (int i = 0; i < n; i++) {
/inner loop to handle number spaces,values changing acc. to requirement
for (int j = n - i; j > 1; j--){//for (int j = 0; j < n - i; j++)
/ printing spaces
  System.out.print(" ");}
/inner loop to handle number of columns values changing acc. to outer loop
for (int j = 0; j <= i; j++) {/ printing stars
    System.out.print("* "); }
// ending line after each row
System.out.println();}}
```

OUTPUT

```
          *
        * *
      * * *
    * * * *
  * * * * *
* * * * * *
 * * * * * * *
  * * * * * * * *
   * * * * * * * * *
    * * * * * * * * * *
```