Contents

[1. Reverse a string without using String inbuilt function 1](#_Toc25638323)

[2. Reverse a string with using String inbuilt function 2](#_Toc25638324)

[3. Swap two numbers without using the third variable. 2](#_Toc25638325)

[4. Count the number of words in a string using HashMap 3](#_Toc25638326)

[5. Print all prime numbers between 1 and 100. 3](#_Toc25638327)

[6. Find whether a string is palindrome or not 4](#_Toc25638328)

[7. Find whether a number is palindrome or not 5](#_Toc25638329)

[10. Remove duplicates from an array without using the Collections 6](#_Toc25638330)

[11. Find the second highest number in an array. 7](#_Toc25638331)

[12. To check Armstrong number 7](#_Toc25638332)

[13. Remove all white spaces from a string with using replace() 8](#_Toc25638333)

[14. Remove all white spaces from a string with out using replace() 8](#_Toc25638334)

[15. Factorial by using Iterative 8](#_Toc25638335)

[16. Factorial by using recursive method 9](#_Toc25638336)

# ****Reverse a string without using String inbuilt function****

Method 1: using charAt(index) . After each iteration, the character will be concatenated to reverse the string variable.

|  |
| --- |
| **public** **static** **void** main(String args[]){  String input = "for help";  String output ="";  **for**(**int** i=input.length()-1; i>=0; i--){  output=output+input.charAt(i);  }  System.***out***.println(output);  } |

Method 2: split() method to split the string into its substrings. After each iteration, substrings added in reverse order

|  |
| --- |
| **public** **static** **void** main(String args[]){  String input;  Scanner sc = **new** Scanner(System.***in***);  input = sc.nextLine();  String[] charArray = input.split("");  String output="";  **for**(**int** i=charArray.length-1; i>=0; i--){  output=output+charArray[i];  }  System.***out***.println(output);  } |

Method 3: toCharArray() converts string to char array. After each iteration, character added in reverse order

|  |
| --- |
| **public** **static** **void** main(String args[]) **throws** IOException{  String input;  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));  input = br.readLine();  **char**[] charArray = input.toCharArray();  String output="";  **for**(**int** i=charArray.length-1; i>=0; i--){  output=output+charArray[i];  }  System.***out***.println(output);  } |

# ****Reverse a string with using String inbuilt function****

Method 1: StringBuffer class has reverse() method

|  |
| --- |
| **public** **static** **void** main(String args[]) **throws** IOException{  String input;  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));  input = br.readLine();  StringBuffer sb = **new** StringBuffer(input);  sb.reverse();  System.***out***.println(sb);  } |

Method 2: StringBuilder class has reverse() method

|  |
| --- |
| **public** **static** **void** main(String args[]) {  String input;  Scanner sc = **new** Scanner(System.***in***);  input = sc.nextLine();  StringBuilder sb = **new** StringBuilder(input);  sb.reverse();  System.***out***.println(sb);  } |

# ****Swap two numbers without using the third variable.****

Input: x = 5, y=2; Output : x=2, y=5

Logic:

1. Assign x with the value x + y which means x will have a sum of both x and y
2. Assign y with the value x – y which means we are subtracting the value of y from the sum of (x + y). Till here, x still has the sum of both x and y. But y has the value of x.
3. Assign x with the value x – y which means we are subtracting y (which has the value of x) from the total (x + y). This will assign x with the value of y and vice versa

|  |
| --- |
| **public** **static** **void** main(String args[]) {  **int** x=5;  **int** y=2;  x=x+y;  y=x-y;  x=x-y;  System.***out***.println(x+" "+y);  } |

# ****Count the number of words in a string using HashMap****

Input : Go gone better, gets going better.

Output: Go=1, gone=1, better=3, gets=1, going=1

Logic:use split() function to split words by space. declared HashMap and iterated using for loop. Inside for loop, we have an if else statement in which wherever at a particular position, the map contains a key, we set the counter at that position and add the object to the map.Each time, the counter is incremented by 1. Else, the counter is set to 1.

|  |
| --- |
| **public** **static** **void** main(String args[]) {  String input ;  Scanner sc = **new** Scanner(System.***in***);  input = sc.nextLine();  String[] words = input.split(" ");  HashMap<String, Integer> wordCount = **new** HashMap<String, Integer>();  **for**(**int** i=0; i<words.length; i++){  **int** value;  **if**(wordCount.containsKey(words[i])){  value = wordCount.get(words[i]);  value++;  }**else**{  value = 1;  }  wordCount.put(words[i],value);  }  System.***out***.println(wordCount);  } |

# ****Print all prime numbers between 1 and 100.****

Ouput: 2,3,5,7,11,13… Number >1 and not be formed by multiplication of any small numbers

Logic: Use for loop starting from 2, less than half of the number are entered and incremented by 1 for each iteration. Temp will have remainder for every iteration. If the remainder is 0, then isPrime will be set to False.

|  |
| --- |
| **public** **class** Practise {  **public** **static** **void** main(String args[]) {  **for**(**int** i=2; i<=100; i++){  **boolean** prime = **true**;  **for**(**int** j=2; j<=i/2; j++){  **if**(i%j ==0){  prime = **false**;  **break**;  }  }  **if**(prime == **true**){  System.***out***.print(" " +i);  }  }  }  } |

# ****Find whether a string is palindrome or not****

Input : MadaM Output : Pallindrome; Input: TEACHER output: Not Pallindrome

Method 1: check first and last char till middle char by char

|  |
| --- |
| **public** **class** Practise {  **public** **static** **void** main(String args[]) {  String input ="ASDOSA";  **int** stringSize = input.length();  **boolean** pallindrome = **true**;  **for**(**int** i=0; i<stringSize/2;i++){  System.***out***.println(input.charAt(i));  System.***out***.println(input.charAt(stringSize-1-i));  **if**(input.charAt(i)!=input.charAt(stringSize-1-i)){  pallindrome = **false**;  **break**;  }    }  **if**(pallindrome == **true**){  System.***out***.println("Pallindrome");  }**else**{  System.***out***.println("Not Pallindrome");  }  }  } |

Method 2: reverse string and compare with Original string

|  |
| --- |
| **public** **static** **void** main(String args[]) {  String input ="ASDSA";  String reverse;  StringBuffer sb = **new** StringBuffer(input);  reverse = sb.reverse().toString();    **if**(input.equals(reverse)){  System.***out***.println("Pallindrome");  }**else**{  System.***out***.println("Not Pallindrome");  }  } |

# ****Find whether a number is palindrome or not****

Logic: Number%10 gives remainder i.e last digit.

|  |
| --- |
| **public** **static** **void** main(String args[]){  **int** input = 123421;  **int** reverse =0;  **int** n=input;  **while**(n>0){  reverse=reverse\*10+(n%10);  n=n/10;  }  **if**(reverse == input){  System.***out***.println("Pallindrome");  }**else**{  System.***out***.println("Not Pallindrome");  }  } |

1. **Fibonacci series**

Fibonacci series is a series of numbers where after the initial two numbers, every occurring number is the sum of two preceding numbers.

Output: 0,1,1,2,3,5,8,13,21………

|  |
| --- |
| **public** **static** **void** main(String args[]) {  **int** current =0,next =0,secondNext =1;  **for**(**int** i=0; i<=10;i++){  current=next;  next=secondNext;  secondNext = current+next;  System.***out***.print(" "+current);  }  } |

1. **Find the duplicate characters in a string**

**Input : Apple, output:p**

**Logic: Convert the string to charArray. Use HaspMap to save characters as key and its count as value. Print all characters having value greater than 1**

|  |
| --- |
| **public** **static** **void** main(String args[]) {  String input = "AppleaPlp";  **char**[] charArray = input.toCharArray();  HashMap<Character, Integer> charCount = **new** HashMap<Character, Integer>();  **for**(**int** i=0; i<charArray.length;i++){  **if**(charCount.containsKey(charArray[i])){  **int** value = charCount.get(charArray[i]);  charCount.put(charArray[i], value+1);  }**else**{  charCount.put(charArray[i], 1);  }  }  Set test = charCount.entrySet();  Iterator itr = test.iterator();  **while**(itr.hasNext()){  Map.Entry<Character, Integer> pair = (Map.Entry<Character, Integer>)itr.next();  **if**(pair.getValue()>1){  System.***out***.println(pair.getKey());  }  }  } |

# ****Remove duplicates from an array without using the Collections****

Method1: If array elements are not sorted then to remove duplicate elements we need to iterate whole array for each current element. During each iteration, set 'null' for each matching element with current element.

|  |
| --- |
| **public** **static** **void** main(String args[]){  Integer[] array ={10,70,30,90,20,20,30,40,70,50};  **int** n = array.length;  Integer[] temp = **new** Integer[n];  **for**(**int** i=0; i<n; i++){  **for**(**int** j=i+1; j<n;j++){  **if**(array[i] == array[j]){  array[i] = **null**;  **break**;  }  }  }  **int** j=0;  **for**(**int** i=0; i<n; i++){  **if**(array[i] !=**null**){  temp[j]=array[i];  j++;  }  }  **for** (**int** i=0; i<j; i++){  array[i] = temp[i];  }  **for** (**int** i=0; i<j; i++){  System.***out***.print(array[i]+ " ");  }  } |

Method2: Sort the array and remove the duplicates

|  |
| --- |
| **public** **static** **int** removeDuplicateElements(**int** arr[], **int** n){  **if** (n==0 || n==1){  **return** n;  }  **int**[] temp = **new** **int**[n];  **int** j = 0;  **for** (**int** i=0; i<n-1; i++){  **if** (arr[i] != arr[i+1]){  temp[j++] = arr[i];  }  }  temp[j++] = arr[n-1];  // Changing original array  **for** (**int** i=0; i<j; i++){  arr[i] = temp[i];  }  **return** j;  }    **public** **static** **void** main (String[] args) {  **int** arr[] = {10,70,30,90,20,20,30,40,70,50};//unsorted array  Arrays.*sort*(arr);//sorting array  **int** length = arr.length;  length = *removeDuplicateElements*(arr, length);  //printing array elements  **for** (**int** i=0; i<length; i++)  System.***out***.print(arr[i]+" ");  } |

# ****Find the second highest number in an array.****

Logic: when the element at the index is greater than the largest, then assign value at index to largest and secondLargest to largest. Else if the element at the index is greater than the secondLargest and not equal to largest assign value at index to secondLargest

|  |
| --- |
| **public** **static** **void** main(String args[]) {  **int**[] input = {1,2,4,5,8,3,71,4,8};  **int** firstHigh =input[0], secondHigh=input[0];  **for**(**int** i=0; i<input.length; i++){  **if**(input[i]>firstHigh){  secondHigh= firstHigh;  firstHigh=input[i];  }**else** **if**(input[i]>secondHigh && input[i] != firstHigh){  secondHigh=input[i];  }  }  System.***out***.println(secondHigh);  } |

# ****To check Armstrong number****

Armstrong number is the number, which is the sum of the cubes of its entire unit, tens and hundred digits for three-digit number

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3 = 1 + 125 + 27 = 153

If you have a four digit number lets say

1634 = 1\*1\*1\*1 + 6\*6\*6\*6 + 3\*3\*3\*3 + 4\*4\*4\*4 = 1+1296+81+256 = 1634

Logic: Find number of digits by repetitive division by 10. Multiply each digit n times. We can get last digit by %10. Exclude last digit by division by 10

|  |
| --- |
| **public** **static** **void** main(String args[]) {  **int** c=0,a,temp;  **int** n=54748 ;//It is the number to check Armstrong  temp=n;  **int** digits =0;  **while**(n>0)  {  n=n/10;  digits=digits+1;  }  n=temp;  **while**(n>0){  a=n%10;  n=n/10;  **int** t=a;  **for**(**int** i=1; i<digits; i++){  a=t\*a;  }  c=c+a;  }  **if**(temp==c){  System.***out***.println("Armstrong");  }**else**{  System.***out***.println("Not Armstrong" +c);  }  } |

# ****Remove all white spaces from a string with using replace()****

|  |
| --- |
| **public** **static** **void** main(String args[]){  String input = "India is my Country";  input = input.replace(" ", "");  System.***out***.println(input);  } |

# ****Remove all white spaces from a string with out using replace()****

Logic: converted that string into a character array using toCharArray().

Using StringBuffer append method to append each character excluding spaces

|  |
| --- |
| **public** **static** **void** main(String args[]){  String input= "I Love my Country";  StringBuffer sb = **new** StringBuffer();  **char**[] charArray = input.toCharArray();  **for**(**int** i=0; i<charArray.length; i++){  **if**(charArray[i] != ' '){  sb.append(charArray[i]);  }  }  System.***out***.println(sb);  } |

# ****Factorial by using Iterative****

The factorial of a number is calculated by formula number\*(number -1) till zero and since the **value of factorial zero is 1**

|  |
| --- |
| **public** **static** **void** main(String args[]){  **int** n = 5;  **int** factorial = 1;  **while**(n!=0){  factorial = factorial \* n;  n = n-1;  }  System.***out***.println(factorial);  } |

# ****Factorial by using recursive method****

Logic: n\*fact(n-1)

|  |
| --- |
| **public** **static** **void** main(String args[]){  **int** n = 5;  **int** factorial ;  factorial = *fact*(n);  System.***out***.println(factorial);  }  **private** **static** **int** fact(**int** n) {  **if**(n==0){  **return** 1;  } **else**{  **return** n \* *fact*(n-1);  }  } |

# ****Print Pyramid****

|  |
| --- |
| **public** **static** **void** main(String args[]) {  **int** rows =5;    **for**(**int** i=1; i<=rows; i++) {  **for**(**int** spaces=0; spaces<rows-i; spaces++) {  System.***out***.print(" ");  }  **for**(**int** j=0; j<2\*i-1; j++) {  System.***out***.print("\*");  }  System.***out***.println();      }    } |