**Java Programme Notes**

**Table Of Contents:**

1. **WAP in java which will take input from user and print the Sum.**
2. **Find greatest number among 3 numbers.**
3. **Write a program in Java to print the sum of all alternative number from 1 to n and print the result in 1+3+5+7=sum.**
4. **Enter number and find factorial and print in following format 1\*2\*3\*4\*5 = 120.**
5. **Write a program to check the given number is prime or not.**
6. **Number Reverse**
7. **Pint 1 To 50 Without Loop**
8. **Remove duplicate from List without Set**
9. **Sort Map with value**
10. **Swap Integer without extra Variable**
11. **String - First non-repeated character in String**
12. **Array – Array second Largest**
13. **Array – Array Sort**
14. **Post Fix Program (Asked in interview)**
15. **Array – Array find first duplicate element**
16. **Array - Find duplicate elements in Array**
17. **Array - Two repeating elements in a given array**
18. **Print Odd Even by 2 Threads**
19. **WAP in java which will take input from user and print the Sum.**

import java.util.Scanner;  
public class SumOf2Numbers {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter first number = ");  
 int a = scanner.nextInt();  
 System.*out*.println("Enter second number = ");  
 int b = scanner.nextInt();  
 System.*out*.println("Sum of 2 Digits = " + (a+b));  
 }  
}

1. **Find greatest number among 3 numbers.**

**Method - 1**

public class GreatestAmong3Method1 {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.println("Enter first number = ");  
 int a = input.nextInt();  
  
 System.*out*.println("Enter second number = ");  
 int b = input.nextInt();  
  
 System.*out*.println("Enter third number = ");  
 int c = input.nextInt();  
  
 if(a > b && a > c){  
 System.*out*.println("First number is greater = "+a);  
 } else if(b > a && b > c){  
 System.*out*.println("Second number is greater = "+b);  
 }else if(c > a && c > b){  
 System.*out*.println("Third number is greater = "+c);  
 } else{  
 System.*out*.println("Not valid comparison.");  
 }  
 }  
}

**Method –2**

import java.util.Scanner;  
public class GreatestAmong3Method2 {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.println("Enter first number = ");  
 int a = input.nextInt();  
  
 System.*out*.println("Enter second number = ");  
 int b = input.nextInt();  
  
 System.*out*.println("Enter third number = ");  
 int c = input.nextInt();  
  
 if(a>b){  
 if(a > c){  
 System.*out*.println("First number is greater = "+a);  
 }else{  
 System.*out*.println("Third number is greater = "+c);  
 }  
 }else if(b > c){  
 System.*out*.println("Second number is greater = "+b);  
 }else {  
 System.*out*.println("Third number is greater = "+c);  
 }  
 }  
}

1. **Write a program in Java to print the sum of all alternative number from 1 to n and print the result in 1+3+5+7=sum.**

import java.util.Scanner;  
public class AlterNumberSum {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Enter number of term = ");  
 int n = in.nextInt();  
 System.*out*.println();  
 int sum = 0;  
 for(int i= 0; i<=n; i++){  
 if(i%2 != 0){  
 sum = sum + i;  
 System.*out*.print(i+"+");  
 }  
 }  
 System.*out*.print("\b = "+sum);  
 System.*out*.println();  
 }  
}

1. **Enter number and find factorial and print in following format 1\*2\*3\*4\*5 = 120.**

**Method – 1**

import java.util.Scanner;  
public class FactorialMethod1 {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Enter number = ");  
 int n = input.nextInt();  
 int fact = 1;  
 System.*out*.println();  
 for(int i=1; i<=n; i++){  
 fact = fact\*i;  
 System.*out*.print(i +"\*");  
 }  
 System.*out*.print("\b = "+fact);  
 System.*out*.println();  
 }  
}

**Method –2 Using recursion:**

import java.util.Scanner;  
public class FactorialMethod2 {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Enter number = ");  
 int num = input.nextInt();  
 System.*out*.println();  
  
 int factorial = *factorial*(num);  
 System.*out*.println("\b = "+factorial);  
  
 }  
 private static int factorial(int num){  
 if(num == 1){  
 System.*out*.print(num+"\*");  
 return 1;  
 }  
 else{  
 System.*out*.print(num+"\*");  
 return num\* *factorial*(num-1);  
 }  
 }  
}

1. **Write a program to check the given number is prime or not.**

**Method – 1**

import java.util.Scanner;  
public class PrimeNumberCheckMethod1 {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Enter number = ");  
  
 int number = input.nextInt();  
 int count = 0;  
  
 for(int i = 1; i <= number/2; i++){  
 if(number % i == 0){  
 count ++;  
 }  
 }  
  
 if(count > 2){  
 System.*out*.println("Not a Prime number.");  
 }else {  
 System.*out*.println("Prime number.");  
 }  
 }  
}

**Method –2**

import java.util.Scanner;  
public class PrimeNumberCheckMethod2 {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Enter number = ");  
  
 int number = input.nextInt();  
 int i = 2;  
 boolean isPrime = true;  
  
 while(number > i){  
 if(number % i == 0){  
 isPrime = false;  
 break;  
 }  
 i++;  
 }  
 if(isPrime){  
 System.*out*.println("Prime");  
 }else {  
 System.*out*.println("Not Prime");  
 }  
  
 }  
}

1. **Number Reverse**

public class NumberReverse {  
  
 public static void main(String[] args) {  
 int num = 567;  
  
 int d;  
 int reverse = 0;  
  
 while(num > 0){  
 d = num % 10;  
 reverse = reverse \* 10 +d;  
 num = num/10;  
 }  
  
 System.*out*.println("Reverse Number = "+reverse);  
 }  
}

1. **Pint 1 To 50 Without Loop**

public class Pint1To50WithoutLoop {  
  
 public static void main(String[] args) {  
 *print*(1);  
 }  
  
 public static void print(int num){  
 System.*out*.print(num + " ");  
 if(num < 50) {  
 *print*(num+1);  
 }  
 }  
}

1. **Remove duplicate from List without Set**

import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class RemoveDuplicateFromListWithoutSet {  
  
 public static void main(String[] args) {  
 List<Integer> integerList = Arrays.*asList*(1, 2, 3, 4, 5, 5, 6, 3, 1, 7);  
  
 System.*out*.println("Before - " +integerList);  
  
  
 System.*out*.println("After - "+ integerList.stream().distinct().collect(Collectors.*toList*()));  
 }  
}

1. **Sort Map with value**

import java.util.HashMap;  
import java.util.LinkedHashMap;  
import java.util.Map;  
import java.util.stream.Collectors;  
  
public class SortMapWithValue {  
  
 public static void main(String[] args) {  
  
 Map<String, Integer> unSortedMap = new HashMap<>();  
  
 unSortedMap.put("A", 40);  
 unSortedMap.put("B", 50);  
 unSortedMap.put("C", 30);  
 unSortedMap.put("D", 10);  
 unSortedMap.put("E", 20);  
  
 unSortedMap = unSortedMap.entrySet().stream().sorted(Map.Entry.*comparingByValue*()).  
 collect(Collectors.*toMap*(Map.Entry::getKey, Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::new));  
  
  
 for(Map.Entry<String, Integer> ttt : unSortedMap.entrySet()){  
 System.*out*.println(ttt.getKey() +" -- "+ttt.getValue());  
 }  
  
  
 }  
}

1. **Swap Integer without extra Variable**

public class SwapIntegerWithoutExtraVariable {  
  
 public static void main(String[] args) {  
 int a = 10;  
 int b = 20;  
  
 a = (a+b);  
 b = a - b;  
 a = a - b;  
  
 System.*out*.println("a = "+a);  
 System.*out*.println("b = "+b);  
 }  
}

1. **String - First non-repeated character in String**

public class FirstNonRepeatedChar {  
  
 public static void main(String[] args) {  
 String str = "My Name is Vikash Singh MyNm";  
  
 int count;  
  
 int strLength = str.length();  
 for(int i = 0; i< strLength; i++){  
 count = 0;  
 for (int j = 0; j< strLength; j++){  
 if(str.charAt(i) == str.charAt(j)){  
 count++;  
 }  
 }  
 if(count < 2 ){  
 System.*out*.println("First non-repeated char = "+str.charAt(i));  
 break;  
 }  
 }  
 }  
}

1. **Array – Array second Largest**

public class ArraySecondLargest {  
  
 public static void main(String[] args) {  
 int[] a = {80, 70, 90, 60, 10, 20, 90};  
  
 int firstLargest = 0;  
 int secondLargest = 0;  
  
 for(int number : a){  
 if(number > firstLargest){  
 secondLargest = firstLargest;  
 firstLargest = number;  
 }  
 *// Checking this first condition because just assume there is only 2 number in list then if we will  
 // not test this else first part condition then secondLargest will remain 0  
 // Checking && number != firstLargest because in case of duplicate firstLargest number condition number > secondLargest  
 // will be true in secondLargest also we will get fistLargest only* else if (number > secondLargest && number != firstLargest) {  
 secondLargest = number;  
 }  
 }  
  
 System.*out*.println("Second largest = "+secondLargest);  
 System.*out*.println("First largest = "+firstLargest);  
 }  
}

1. **Array – Array Sort**

public class ArraySort {  
  
 public static void main(String[] args) {  
  
 Integer[] intArray = {80, 40, 50, 20, 30, 10, 70, 60};  
  
 for(int i=0; i<intArray.length; i++){  
 for(int j = i+ 1; j< intArray.length; j++){  
 if(intArray[i] > intArray[j]){  
 int temp = intArray[i];  
 intArray[i] = intArray[j];  
 intArray[j] = temp;  
 }  
 }  
 }  
  
 for(int i=0; i<intArray.length; i++){  
 System.*out*.print(intArray[i]+" ");  
 }  
 }  
}

1. **Post Fix Program (Asked in interview)**

import java.util.Scanner;  
import java.util.Stack;  
  
public class PostFixProgram {  
  
 public static void main(String[] args) {  
 *//In a postfix expression,  
 // • an operator is written after its operands.  
 // • the infix expression 2+3 is 23+ in postfix notation.  
 // • For postfix expressions, operations are performed in the order in which they are  
 // written (left to right).  
 // • No parentheses are necessary. ‘  
 // • the infix expression 2+3\*4 is 234\*+ in postfix notation  
 // • the infix expression 3\*4+2\*5 translates to 34\*25\*+ in postfix notation.  
 // • the infix expression 3\*(4+2)\*5 translates to 342+\*5\*  
   
 //Scanner scanner = new Scanner(System.in);* String expression = "234\*+";  
  
 *//String exp = "823\*+7/1-";* System.*out*.print("The PostFix Evaluation for the Given Expression " + expression + " is: ");  
 *evaluatePostfix*(expression);  
 }  
  
 static void evaluatePostfix(String exp) {  
 Stack<Integer> postFix = new Stack<>(); *// Create postfix stack* int n = exp.length();  
  
 for (int i = 0; i < n; i++) {  
 if (*isOperator*(exp.charAt(i))) {  
 *// pop top 2 operands.* int op1 = postFix.pop();  
 int op2 = postFix.pop();  
  
 *// evaluate in reverse order i.e. op2 operator op1.* switch (exp.charAt(i)) {  
 case '+':  
 postFix.push(op2 + op1);  
 break;  
  
 case '-':  
 postFix.push(op2 - op1);  
 break;  
  
 case '\*':  
 postFix.push(op2 \* op1);  
 break;  
  
 case '/':  
 postFix.push(op2 / op1);  
 break;  
  
 }  
  
 }  
 *// Current Char is Operand simple push into stack* else {  
 *// convert to integer* int operand = exp.charAt(i) - '0';  
 postFix.push(operand);  
 }  
 }  
  
 *// Stack at End will contain result.* System.*out*.println(postFix.pop());  
 }  
  
 static boolean isOperator(char ch) {  
 if (ch == '+' || ch == '-' || ch == '\*' || ch == '/')  
 return true;  
  
 return false;  
 }  
}

1. **Array – Array find first duplicate element**

public class ArrayFindFirstDuplicateElement {  
  
 public static void main(String[] args) {  
 int[] a = {2, 4, 3, 8 , 9, 4, 7};  
 int count = 1;  
 for(int i= 0; i< a.length; i++){  
 for(int j = i+1; j< a.length; j++){  
 if(a[i] == a[j]){  
 count ++;  
 }  
 }  
  
 if(count >= 2){  
 System.*out*.println("Duplicate found "+a[i]);  
 break;  
 }  
 }  
 }  
}

1. **Array - Find duplicate elements in Array**

public class ArrayFindDuplicateInArray {  
  
 public static void main(String args[])  
 {  
 int numArray[] = { 0, 4, 3, 2, 7, 8, 2, 3, 1 };  
  
 int arrayLength = numArray.length;  
  
 for (int i = 0; i < numArray.length; i++) {  
 numArray[numArray[i] % arrayLength] = numArray[numArray[i] % arrayLength] + arrayLength;  
 }

System.*out*.println("The repeating elements are : ");

for (int i = 0; i < arrayLength; i++) {  
 if (numArray[i] >= arrayLength \* 2) {  
 System.*out*.println(i + " ");  
 }  
 }  
 }  
}

1. **Array - Two repeating elements in a given array**

public class ArrayTwoRepeatingElements {  
  
 public static void main(String[] args) {  
 int arr[] = {4, 2, 4, 5, 2, 3, 1};  
 int arrSize = arr.length;  
 *printRepeating*(arr, arrSize);  
 }

static void printRepeating(int array[], int size) {  
 int i, j;  
 System.*out*.print("Repeating Elements are ");  
 for (i = 0; i < size - 1; i++) {  
 for (j = i + 1; j < size; j++) {  
 if (array[i] == array[j])  
 System.*out*.print(array[i] + " ");  
 }  
 }  
 }

}

1. **Print Odd Even by 2 Threads**

**Method – 1**

public class OddEvenBy2ThreadCompletableFuture {  
  
 private static Object *object* = new Object();  
  
 private static IntPredicate *evenCondition* = e -> e % 2 == 0;  
 private static IntPredicate *oddCondition* = e -> e % 2 != 0;  
  
 public static void main(String[] args) throws InterruptedException {  
  
 *// Odd number printer* CompletableFuture.*runAsync*(() -> OddEvenBy2ThreadCompletableFuture.*printNumber*(*oddCondition*));  
  
 *// Even number printer* CompletableFuture.*runAsync*(() -> OddEvenBy2ThreadCompletableFuture.*printNumber*(*evenCondition*));  
  
 Thread.*sleep*(1000);  
 }  
  
 public static void printNumber(IntPredicate condition){  
 IntStream.*rangeClosed*(1, 10).filter(condition).forEach(OddEvenBy2ThreadCompletableFuture::*execute*);  
 }  
  
 public static void execute(int num){  
 synchronized (*object*){  
 try{  
 System.*out*.println(Thread.*currentThread*().getName()+" : "+num);  
 *object*.notify();  
 *object*.wait();  
 }catch (InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
  
 }  
}

**Method – 2**

import java.util.concurrent.atomic.AtomicInteger;  
  
public class OddEvenBy2ThreadSimpleApproach {  
  
 static AtomicInteger *atomicNumber* = new AtomicInteger(1);  
  
 public static void main(String[] args) {  
 Runnable print = () -> {  
 while (*atomicNumber*.get() < 10) {  
 synchronized (*atomicNumber*) {  
 if ((*atomicNumber*.get() % 2 == 0) && "Even".equals(Thread.*currentThread*().getName())) {  
 System.*out*.println("Even" + ":" + *atomicNumber*.getAndIncrement());  
 }  
 else {System.*out*.println("Odd" + ":" + *atomicNumber*.getAndIncrement());  
 }  
 }  
 }  
 };  
  
 Thread t1 = new Thread(print);  
 t1.setName("Even");  
 t1.start();  
 Thread t2 = new Thread(print);  
 t2.setName("Odd");  
 t2.start();  
  
 }  
}

**Method – 3**

public class OddEvenBy2ThreadOldApproach {  
  
 public static void main(String[] args){  
 Printer print = new Printer();  
 Thread t1 = new Thread(new TaskEvenOdd(print, 10, false));  
 Thread t2 = new Thread(new TaskEvenOdd(print, 10, true));  
 t1.setName("Odd-Thread");  
 t2.setName("Even-Thread");  
 t1.start();  
 t2.start();  
 }  
}  
  
class TaskEvenOdd implements Runnable {  
  
 private int max;  
 Printer printer;  
 private boolean isEven;  
  
 TaskEvenOdd(Printer printer, int max, boolean isEven){  
 this.printer = printer;  
 this.max = max;  
 this.isEven = isEven;  
 }  
  
 @Override  
 public void run() {  
  
 int number = (isEven == true) ? 2 : 1;  
  
 while(number < max){  
  
 if(isEven){  
 printer.printEven(number);  
 }  
 else {  
 printer.printOdd(number);  
 }  
 number+=2;  
 }  
 }  
}  
  
class Printer {  
  
 boolean isOdd = true;  
  
 synchronized void printEven(int number) {  
  
 if(isOdd){  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 System.*out*.println("Even:"+number);  
 isOdd = true;  
 notifyAll();  
 }  
  
 synchronized void printOdd(int number) {  
 if(!isOdd){  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 System.*out*.println("Odd:"+number);  
 isOdd = false;  
 notifyAll();  
 }  
  
}