Q.1)

Armstrong Number –

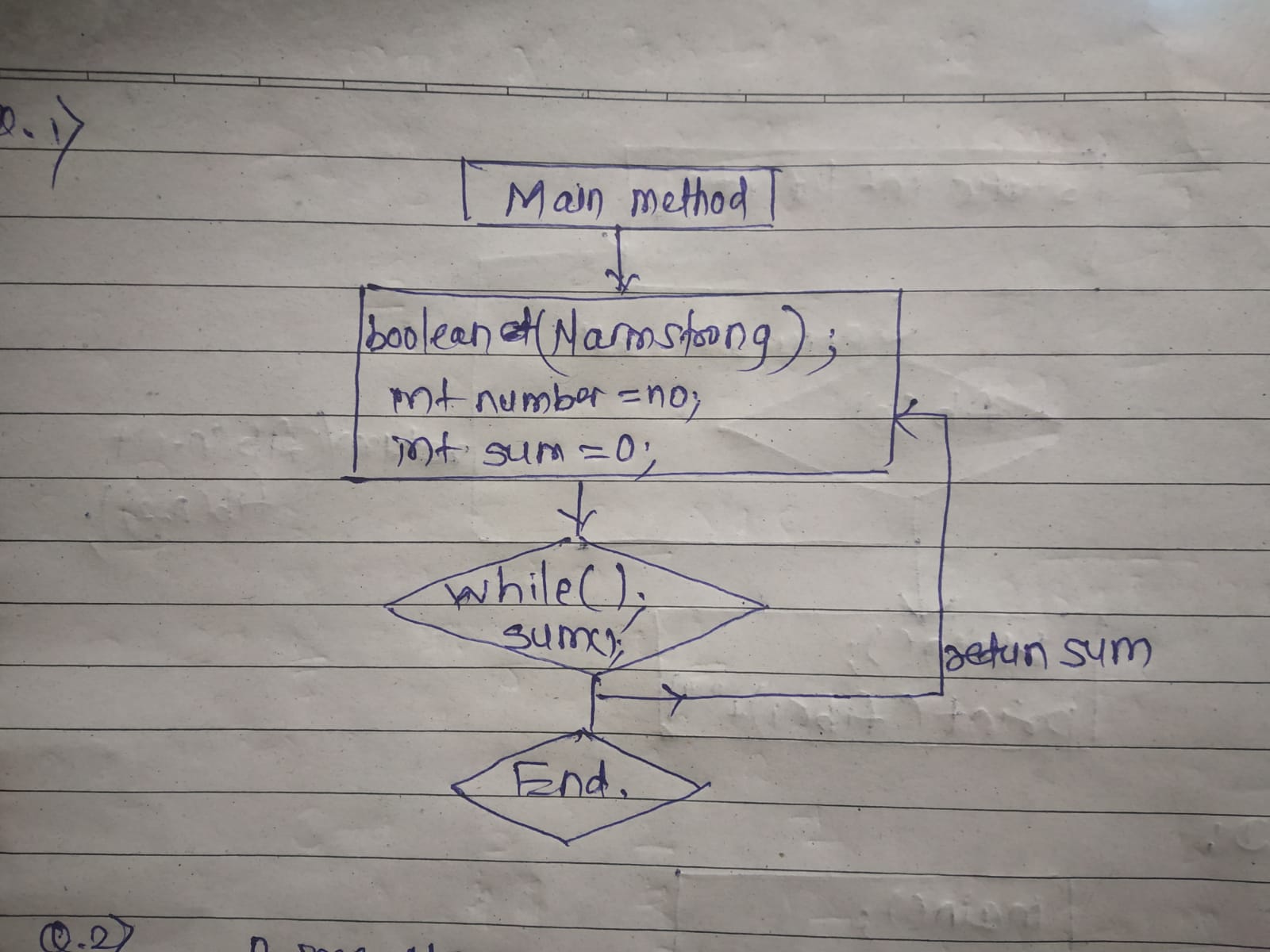
1.Take first input by using as Scanner input.

2. Boolean checks if number is Armstrong or not

3.while() block checks if number is Armstrong or not

Digit = number %10; gives last digit of number

Armstrong no is sum of (digit)^3 = number;



// armstrong Number//

import java.util.Scanner;

public class Numarmstrong{

public static boolean Narmstrong(int number){

int Number = number;

int sum =0;

while(number > 0){

int digit = number% 10;

sum += digit\*digit\*digit;

number /=10;

}

return sum == Number;

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number : ");

int number = sc.nextInt();

if (Narmstrong(number)){

System.out.println("Number is armstrong (true) :"+number);

}

else{

System.out.println(" Number is not armstrong (false) :"+number);

}

sc.close();

}

}

Q2.)Prime number

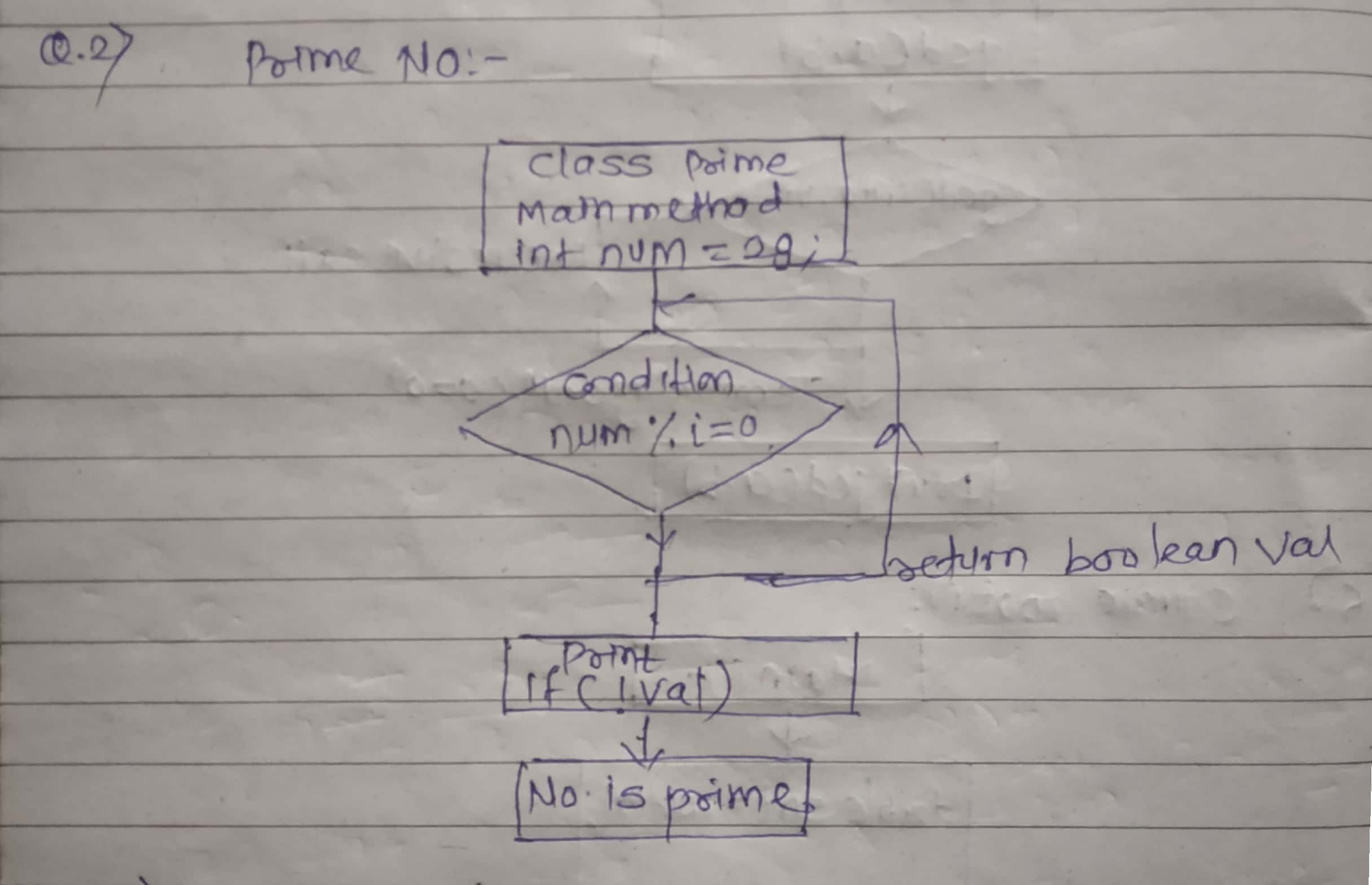
Take input as prime number as 29;

For a prime number it must be divisible by 1 and itself.

Check condition by using for loop

For( int I =2; i<= num /2 ; ++i);

We check val true / false by Boolean method.



public class prime {

public static void main(String[] args) {

int num = 29;

boolean val = false; // 0 and 1 are not prime numbers

if (num == 0 || num == 1) {

val = true;

}

for (int i = 2; i <= num / 2; ++i) {

// condition for nonprime number

if (num % i == 0) {

val = true;

break;

}

}

if (!val)

System.out.println(num + " is a prime number.");

else

System.out.println(num + " is not a prime number.");

}

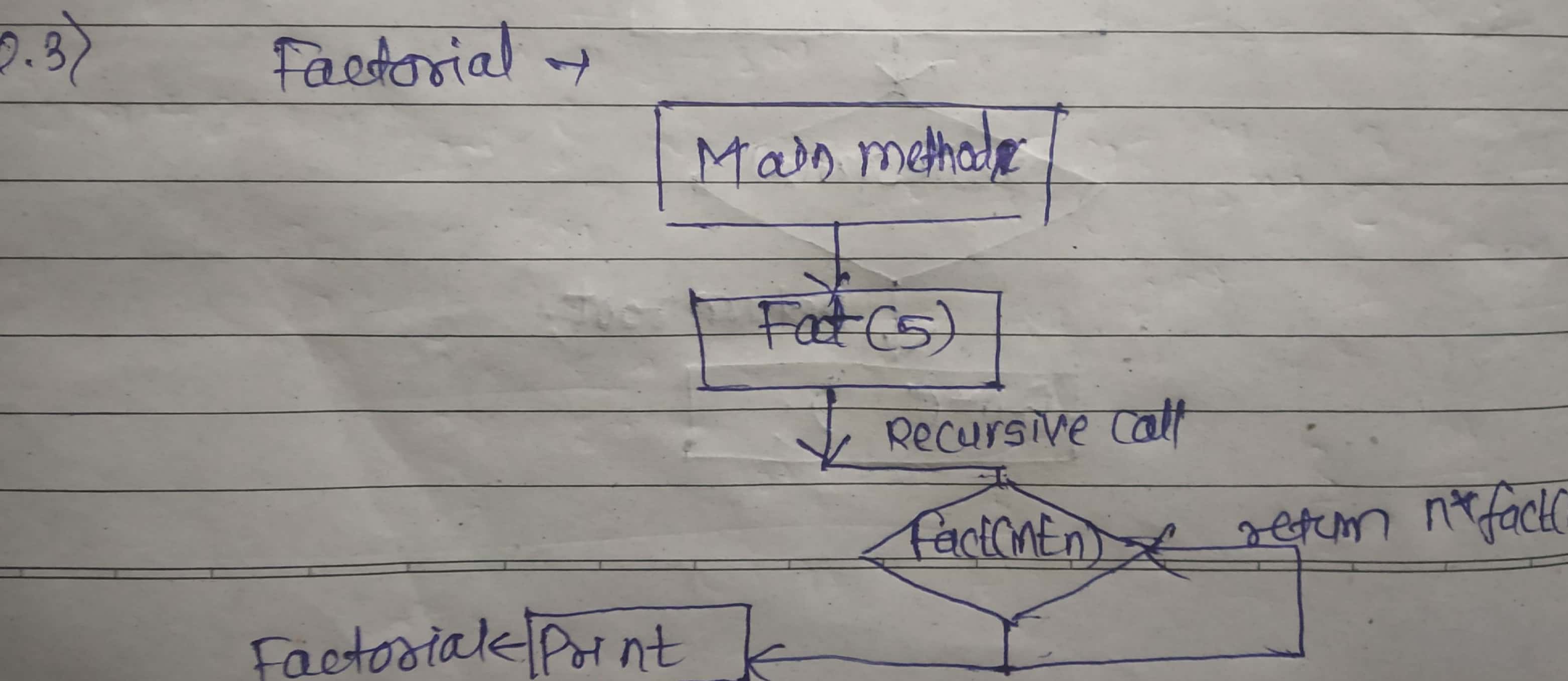
}

Q 3) Factorial

Here we use recursion method by static int fact (int n) method:

Fact n = n\* fact(n-1);

If condition is n is less than equal to 1 return false.



class factorial{

static int fact(int n){

if (n<=1)

return 1;

else

return n\*fact(n-1); }

public static void main(String [] args){

System.out.println(fact(5));

}

}

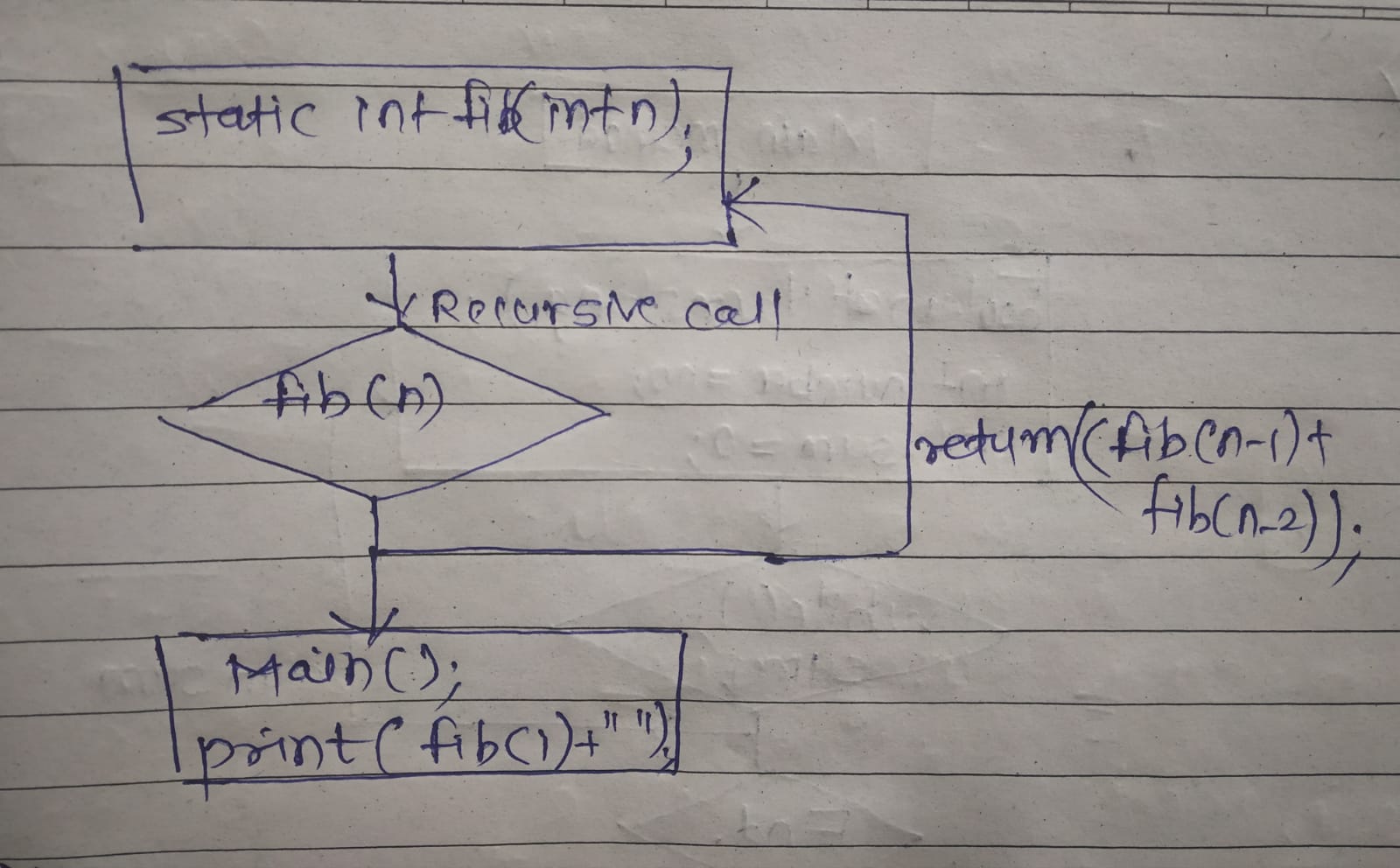
Q4) Fibonacci Series

Fibonacci series goes by 0 1 1 2 3 5 8 13 21 34

In Fibonacci series sum of last two digit is equal to last third digit.

Fib = fib(n-1) + fib(n-2);

Here we call fib function recursively inside itself to iterate till its terminating condition.

class fibonacci{

static int fib(int n){

if(n<= 1){

return n;

}

return fib(n-1)+fib(n-2);

}

public static void main(String[] args){

int num = 5;

System.out.print("[");

for(int i =0;i<=num;i++){

System.out.print(fib(i)+" ");

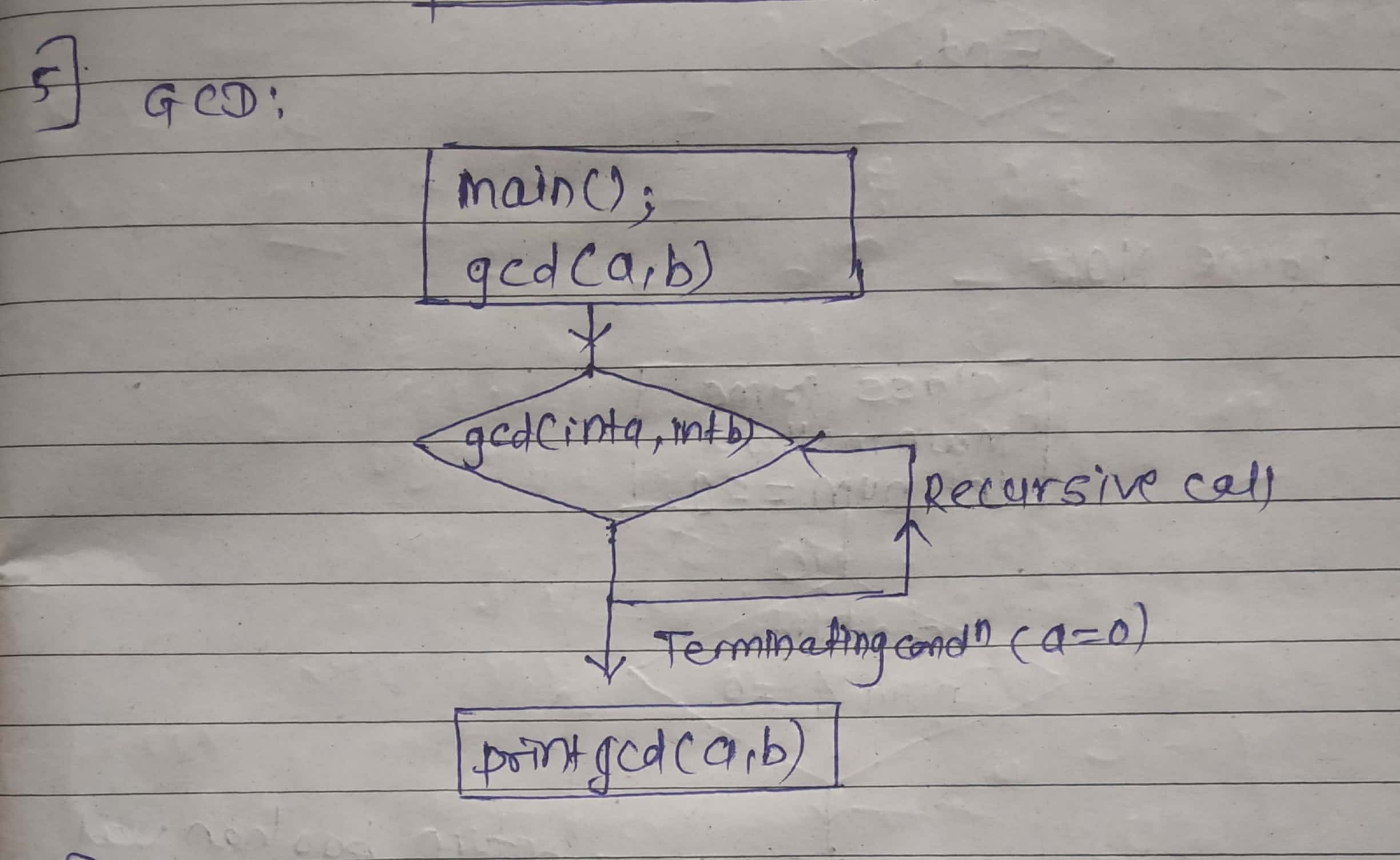
}

System.out.print("]");}

} // [0 1 1 2 3 5]

5. Find GCD

Here we calculate gcd of two numbers by calling Gcd(b,c) inside main function and its recursively call inside its static function.



import java.util.Scanner;

class Gcd{

public static int gcd(int a, int b)

{

if(b==0){

return a;

}

if(a>b){

return gcd(a%b,b);

} else{

return gcd(a,b%a);

}

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println(" Enter a and b");

int a = sc.nextInt();

int b = sc.nextInt();

System.out.println("GCD is"+gcd(a,b));

}

}

6. Find Square Root

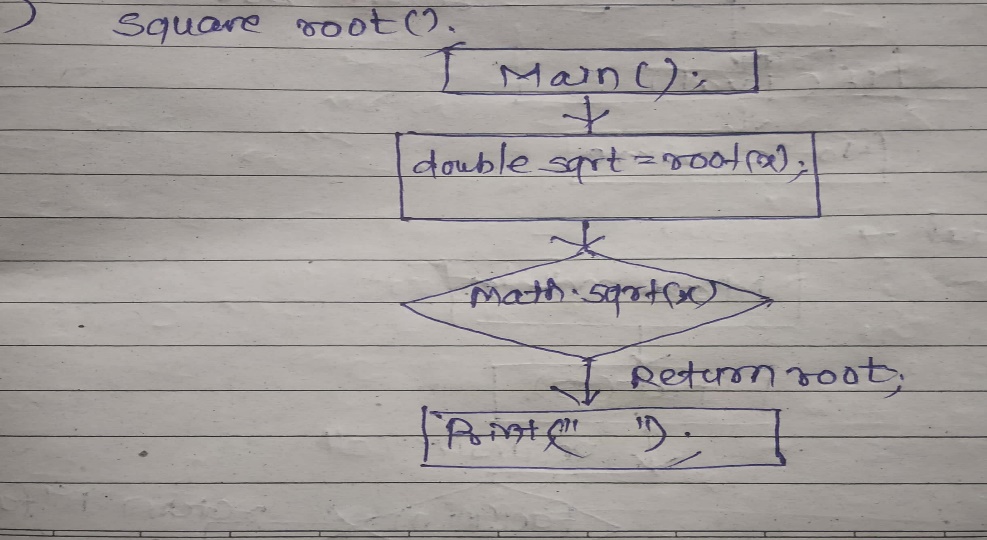
Problem: Write a Java program to find the square root of a given number

First we take input of number using scanner

Then define function for square root in main as, double squrt = root(x);

Define new method and first compare the no is equal to 0 or not if 0 then return as invalid.

If no > 0 then calculate the square root as -🡪 double root = Math.sqrt.(x);



class Sqroot{

public static double root(int x){

if (x == 0){

System.out.println(" the number is invalid");

return 0;

}

else {

double root = Math.sqrt(x);

return root;

}

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number : ");

int x = sc.nextInt();

double sqrt = root(x);

System.out.println("square root of a number is"+sqrt);

}

7. Find Repeated Characters in a String

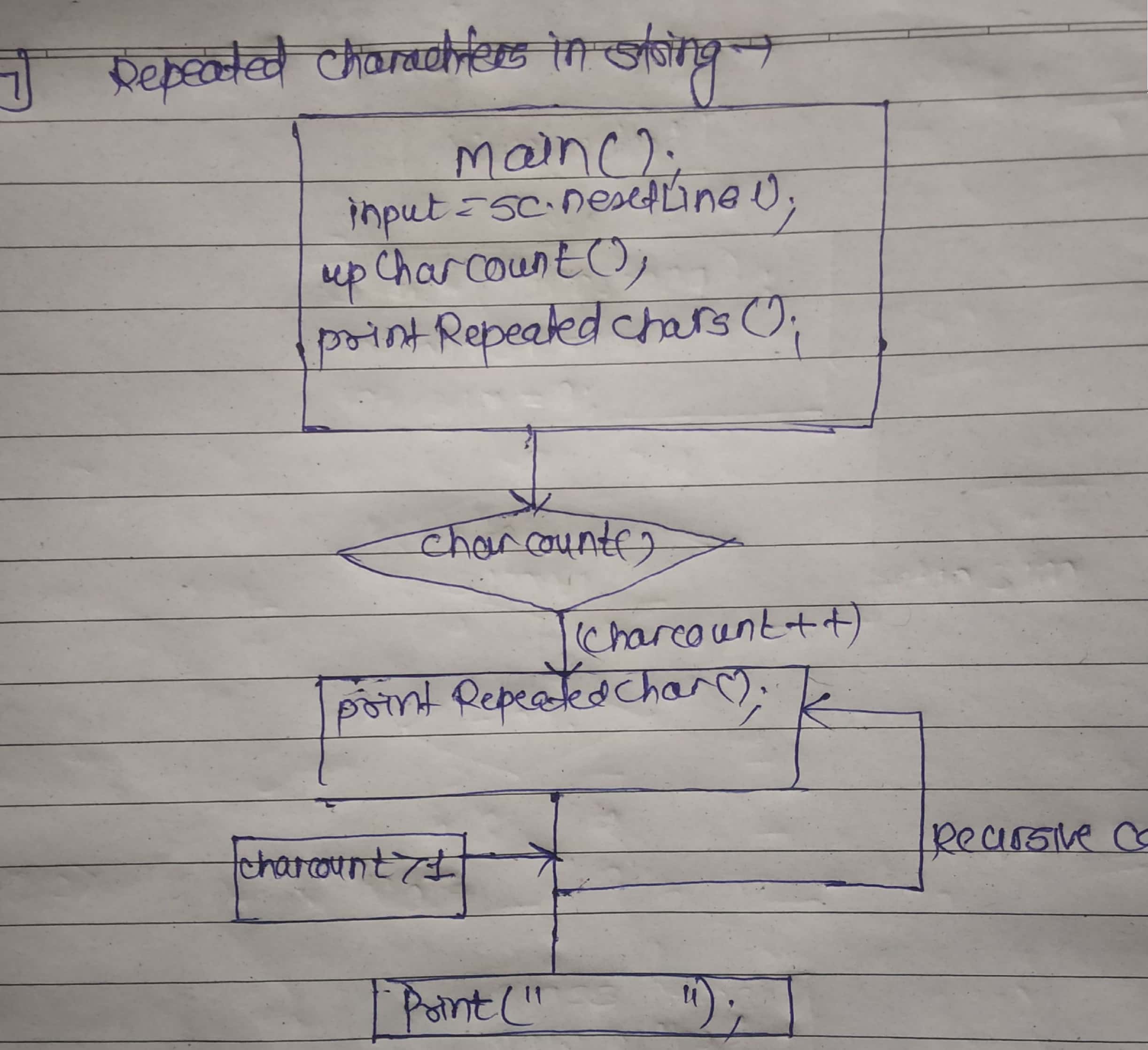
Problem: Write a Java program to find all repeated characters in a string.

1.define charcount function to count characters of string

2.define up charcount to get count of characters according to its index value

3. print the characters by condition where charcount > 1 , this will print only repeated characters of string .

4. here we call as charcount which traverse the string with respective of its index value.



class RepeatedCharacters {

//Recursive function to characters

public static void upCharCount(String str, int[] charCount, int index) {

// Base case: If index reaches the end of the string, stop recursion

if (index == str.length()) {

return;

}

// Get the character

char currentChar = str.charAt(index);

charCount[currentChar]++;

// Recursively call

upCharCount(str, charCount, index + 1);

}

// Recursive function

public static void printRepeatedChars(String str, int[] charCount, int index) {

// Base case: If index reaches the end of the string, stop recursion

if (index == str.length()) {

return;

}

char currentChar = str.charAt(index);

if (charCount[currentChar] > 1) {

System.out.print(currentChar + " ");

charCount[currentChar] = 0; // Set to 0 to avoid printing it again

}

printRepeatedChars(str, charCount, index + 1);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string: ");

String input = sc.nextLine();

int[] charCount = new int[256];

upCharCount(input, charCount, 0);

System.out.print("Repeated characters are: ")

printRepeatedChars(input, charCount, 0);

}

}

8. First Non-Repeated Character

Problem: Write a Java program to find the first non-repeated character in a string.

The program is same as repeated characters string only difference is in the condition of

Char Count [currentChar] <= 1; this will insure the repeatation is avoided.

9. Integer Palindrome

Problem: Write a Java program to check if a given integer is a palindrome

Palindrome means number is same if transverse from either sides.

For this , first we take input 121, then we check if no .is palindrome or not by using Boolean isPalindrome method.

Conditions.

Num > 0;

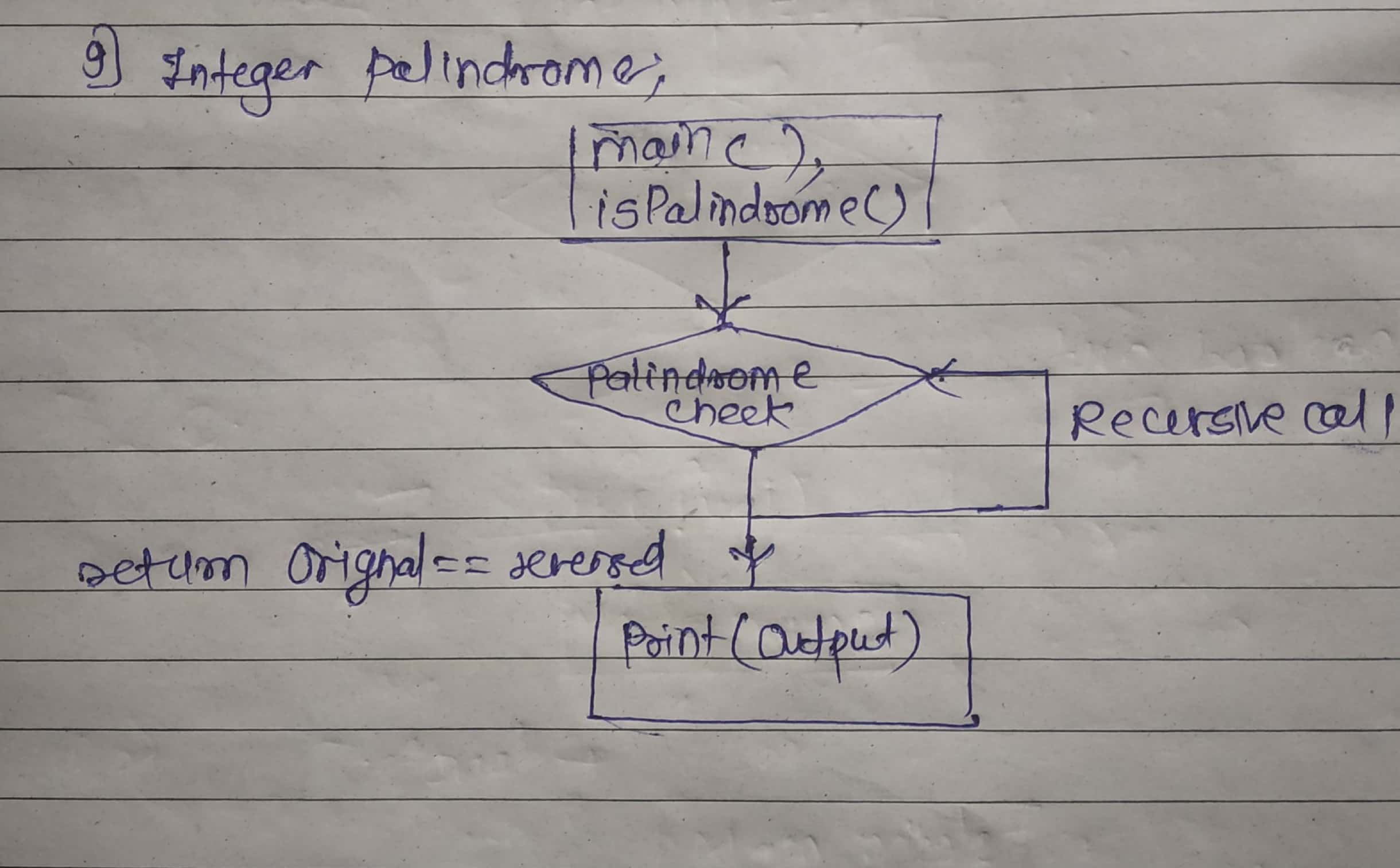
To reverse the number we take modulus of number by using % operator;

This will gove last digit which we will append into reversed number.

To check => compare if original number is same as reversed number;

Revesed == original;

Print the output in main method.



class PalindromeCheck {

public static boolean isPalindrome(int num) {

if (num < 0 ) {

return false;

}

int original = num; // input number

int reversed = 0;

// Reverse the integer

while (num > 0) {

int digit = num % 10; // Get the last digit

reversed = (reversed \* 10) + digit; // Append it to the reversed number

num /= 10; // Remove the last digit from new value

}

// Compare

return original == reversed;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter an integer: ");

int num = sc.nextInt();

if (isPalindrome(num)) {

System.out.println("Output: true");

} else {

System.out.println("Output: false");

}

}

}