**PRACTICAL NO – 18**

Write a java file handling program to count and display the number of palindromes present in a text file "myfile.txt". Example:

If the file "myfile.txt" contains the following lines,

My name is NITIN

Hello aaa and bbb word

How are You

ARORA is my friend

Output will be => 4

**SOURCE CODE –**

import java.io.\*;

import java.util.\*;

public class PalindromeCounter {

public static void main(String[] args) throws IOException {

int count = 0;

FileReader fr = new FileReader("first.txt");

BufferedReader br = new BufferedReader(fr);

String line;

System.out.println("Palindrome words are:");

while ((line = br.readLine()) != null) {

String[] words = line.split("\\s+");

for (String word : words) {

if (isPalindrome(word)) {

System.out.println(word);

count++;

}

}

}

br.close();

System.out.println("Total Palindromes: " + count);

}

public static boolean isPalindrome(String word) {

word = word.toLowerCase();

int left = 0, right = word.length() - 1;

while (left < right) {

if (word.charAt(left) != word.charAt(right)) {

return false;

}

left++;

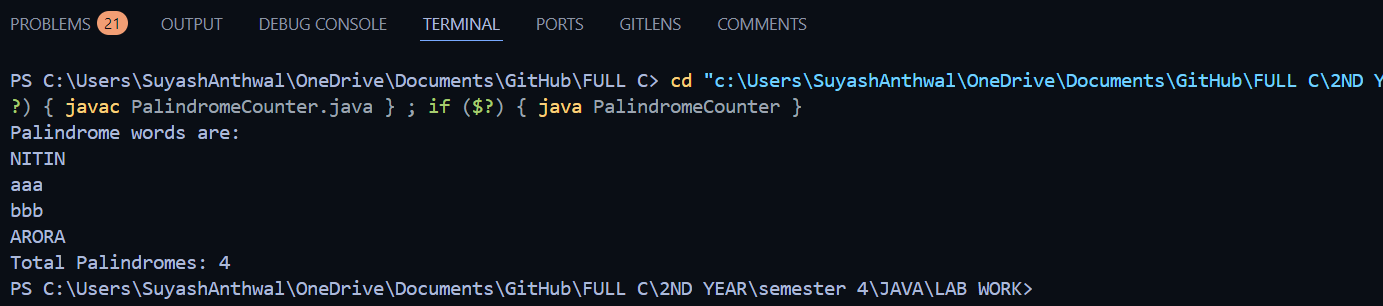
right--;

}

return true ;

}

}

**OUTPUT-**

**PRACTICAL NO – 19**

Write a program MultiThreads that creates two threads one thread with the name CSthread and the other thread named ITthread. Each thread should display its respective name and execute after a gap of 500 milliseconds. Each thread should also display a number indicating the number of times it got a chance to execute.

**SOURCE CODE –**

class CSThread extends Thread {

public void run() {

for (int i = 0; i < 5; i++) {

System.out.println(getName() + " executed " + i + " times");

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

class ITThread extends Thread {

public void run() {

for (int i = 0; i < 5; i++) {

System.out.println(getName() + " executed " + i + " times");

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

public class thread {

public static void main(String[] args) {

// Create and start the threads

CSThread t1 = new CSThread();

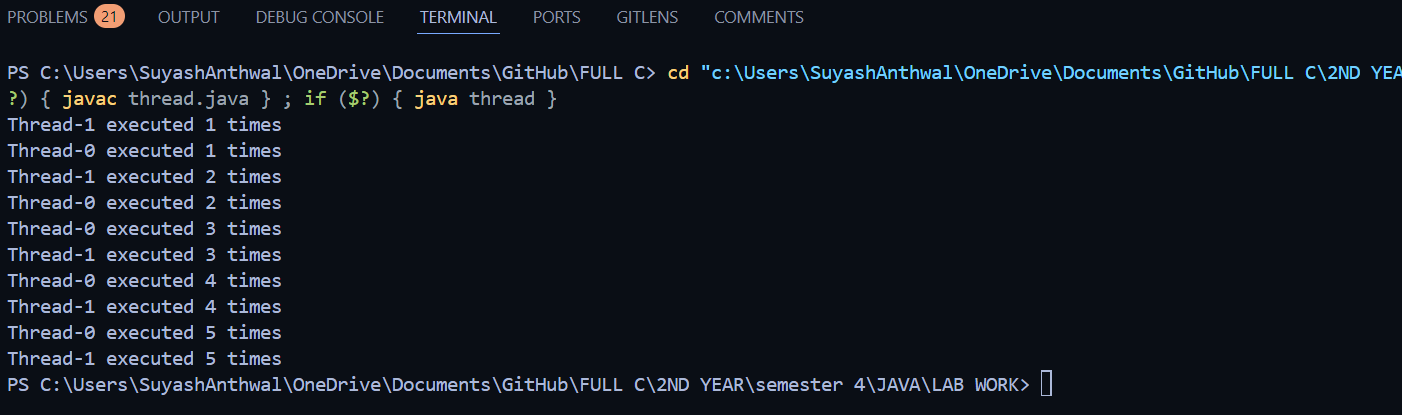
ITThread t2 = new ITThread();

t1.start();

t2.start();

}

}

**OUTPUT-**

**PRACTICAL NO – 20**

Write a java program for to solve producer consumer problem in which a producer produce a value and consumer consume the value before producer generate the next value.

**SOURCE CODE –**

import java.util.\* ;

class PC1 {

LinkedList<Integer> list = new LinkedList<>();

int capacity = 2;

int value = 0;

public void produce() throws Exception {

while (true) {

synchronized (this) {

while (list.size() == capacity) {

wait();

}

System.out.println("Producer is going to produce...... " + value);

list.add(value++);

notifyAll();

Thread.sleep(500);

}

}

}

public void consume() throws Exception {

while (true) {

synchronized (this) {

while (list.size() == 0) {

wait();

}

System.out.println("Consumer is going to consume ...... " + list.removeFirst());

notifyAll();

Thread.sleep(500);

}

}

}

}

class pro extends Thread {

PC1 p;

public pro(PC1 p) {

this.p = p;

}

public void run() {

try {

p.produce();

} catch (Exception e) {

System.out.println(e);

}

}

}

class con extends Thread {

PC1 p;

public con(PC1 p) {

this.p = p;

}

public void run() {

try {

p.consume();

} catch (Exception e) {

System.out.println(e);

}

}

}

public class proANDcon {

public static void main(String[] args) {

PC1 p = new PC1();

pro t1 = new pro(p);

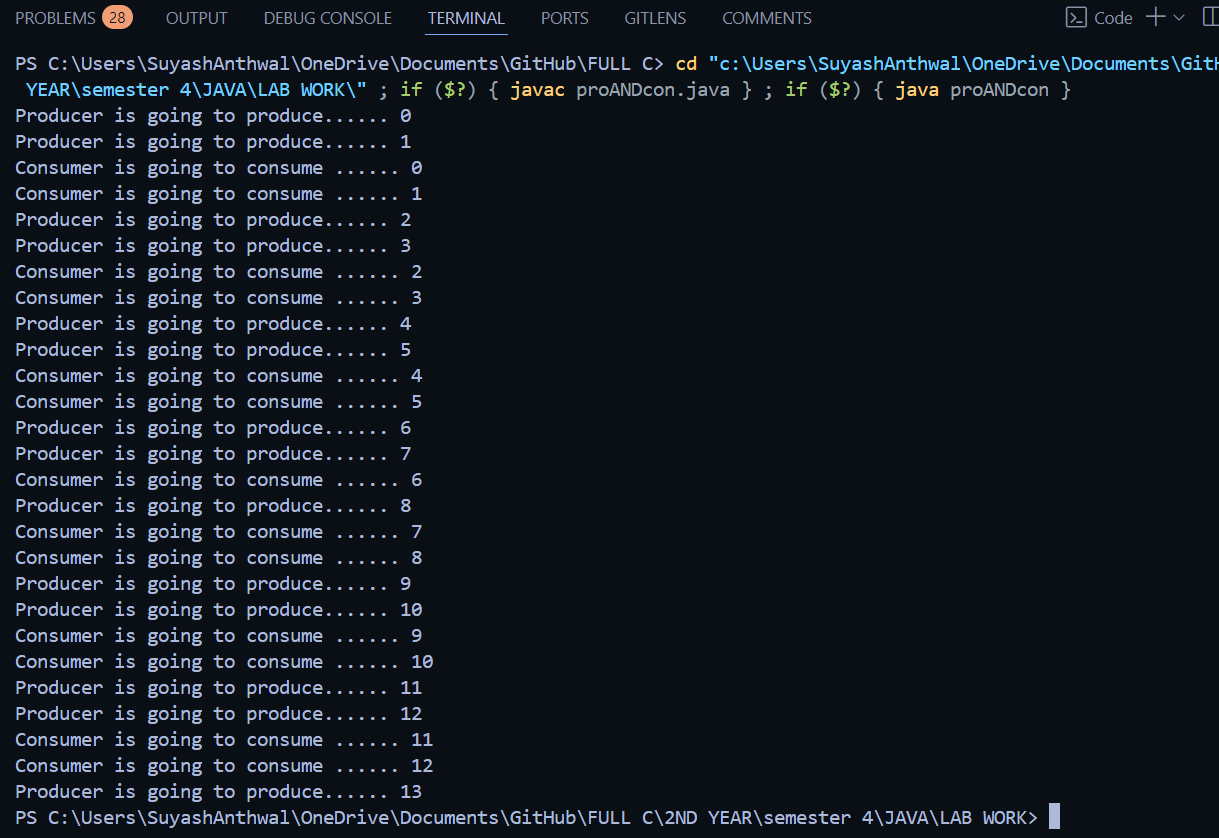
con t2 = new con(p);

t1.start();

t2.start();

}

}

**OUTPUT-**