**LAB 7**

**1.     Write the programme to open a text file named input 2, and copy its contents to an output text file output 2.**

**CODE:-**

package Demo;

import java.io.\*;

public class Copydemo {

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

{

File inputfile=new File("C:\\\\Prachit\\input.txt");

File outputfile=new File("C:\\Prachit\\output.txt");

FileReader in = new FileReader(inputfile);

FileWriter out= new FileWriter(outputfile);

int a;

while((a=in.read())!=-1)

{

out.write(a);

}

System.out.println("file executed");

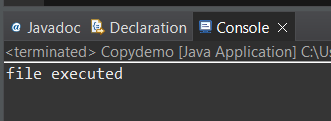
in.close();

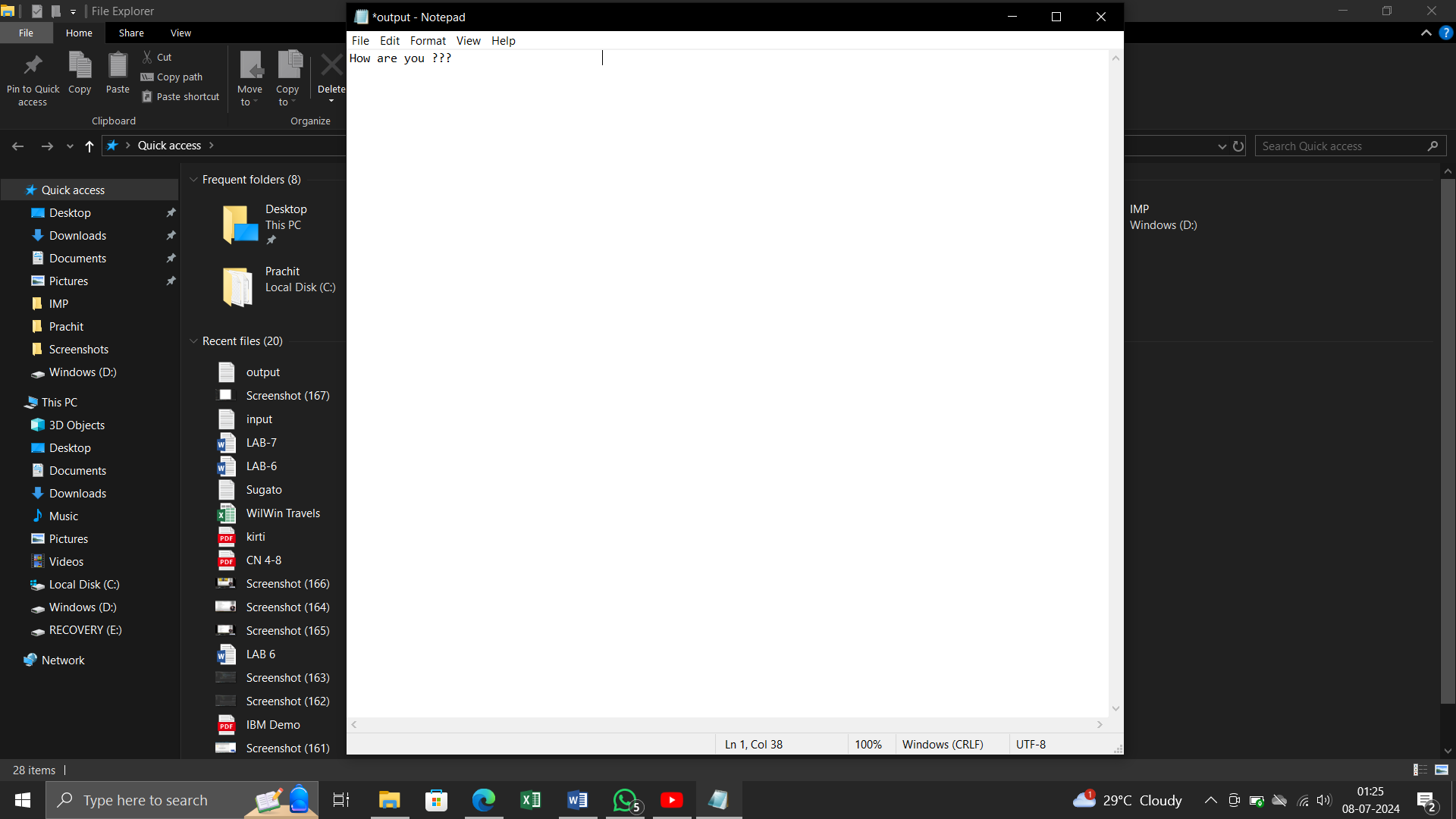
out.close();

}

}

**OUTPUT:-**





**2.     Write the programme to show multithreading for the string “multi threads”. Show the resulting output.**

**CODE:-**

package demo2;

public class MultiThreads {

public static void main(String[] args) {

String str = "multi threads";

int numThreads = str.length();

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

int index = i;

new Thread(() -> {

System.***out***.print(str.charAt(index) + " ");

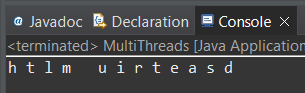
}).start();

}

}

}

**OUTPUT:-**



**3.     Implement a Java program that creates a thread using the Runnable interface. The thread should print numbers from 1 to 10 with a delay of 1 second between each number.**

**CODE:-**

package demo2;

public class RunnableExample implements Runnable {

public void run() {

for (int i = 1; i <= 10; i++) {

System.***out***.println(i);

try {

Thread.*sleep*(1000); // Sleep for 1 second

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

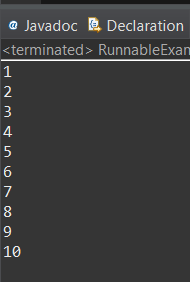
Thread thread = new Thread(new RunnableExample());

thread.start();

}

}

**OUTPUT:-**



**4.     Write a Java program that creates and starts three threads. Each thread should print its name and count from 1 to 5 with a delay of 500 milliseconds between each count.**

**CODE:-**

package demo2;

public class ThreeThreadsExample {

public static void main(String[] args) {

for (int i = 0; i < 3; i++) {

Thread thread = new Thread(() -> {

for (int j = 1; j <= 5; j++) {

System.***out***.println(Thread.*currentThread*().getName() + ": " + j);

try {

Thread.*sleep*(500); // Sleep for 500 milliseconds

} catch (InterruptedException e) {

e.printStackTrace();

}

}

});

thread.setName("Thread-" + (i + 1));

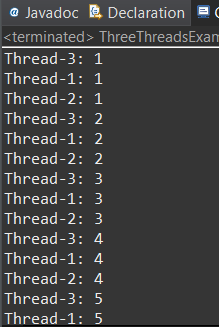
thread.start();

}

}

}

**OUTPUT:-**



**5.     Create a Java program that demonstrates thread priorities. Create three threads with different priorities and observe the order in which they execute.**

**CODE:-**

package demo2;

public class ThreadPriorityExample {

public static void main(String[] args) {

Thread highPriorityThread = new Thread(() -> {

for (int i = 0; i < 1; i++) {

System.***out***.println(Thread.*currentThread*().getName() + " (Priority: " + Thread.*currentThread*().getPriority() + ")");

}

}, "High Priority Thread");

Thread mediumPriorityThread = new Thread(() -> {

for (int i = 0; i < 1; i++) {

System.***out***.println(Thread.*currentThread*().getName() + " (Priority: " + Thread.*currentThread*().getPriority() + ")");

}

}, "Medium Priority Thread");

Thread lowPriorityThread = new Thread(() -> {

for (int i = 0; i < 1; i++) {

System.***out***.println(Thread.*currentThread*().getName() + " (Priority: " + Thread.*currentThread*().getPriority() + ")");

}

}, "Low Priority Thread");

// Set thread priorities

highPriorityThread.setPriority(Thread.***MAX\_PRIORITY***); // 10

mediumPriorityThread.setPriority(Thread.***NORM\_PRIORITY***); // 5

lowPriorityThread.setPriority(Thread.***MIN\_PRIORITY***); // 1

// Start threads

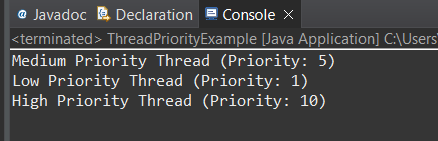
lowPriorityThread.start();

mediumPriorityThread.start();

highPriorityThread.start();

} }

**OUTPUT:-**



**6.     Write a Java program that creates a deadlock scenario with two threads and two resources.**

**CODE:-**

package demo2;

public class TestDeadlockDemo {

public static void main(String[] args) {

final String resource1 = "karan kumar";

final String resource2 = "ratan kumar";

Thread t1 = new Thread() {

public void run() {

synchronized (resource1) {

System.***out***.println("Thread 1: resource 1 locked");

try { Thread.*sleep*(200);} catch (Exception e) {}

synchronized (resource2) {

System.***out***.println("Thread 1: resource 1 locked");

}

}

}

};

Thread t2 = new Thread() {

public void run() {

synchronized (resource2) {

System.***out***.println("Thread 2: resource 2 locked ");

try { Thread.*sleep*(100);} catch (Exception e) {}

synchronized (resource1) {

System.***out***.println("Thread 2: resource 1 locked ");

}

}

}

};

t1.start();

t2.start();

}

}

**OUTPUT:-**

