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

Program:

**package** Home;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**public** **class** FileCopy {

**public** **static** **void** main(String[] args) {

String inputFileName = "input2.txt";

String outputFileName = "output2.txt";

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(inputFileName));

BufferedWriter writer = **new** BufferedWriter(**new** FileWriter(outputFileName))) {

String line;

**while** ((line = reader.readLine()) != **null**) {

writer.write(line);

writer.newLine();

}

System.***out***.println("Contents copied successfully from " + inputFileName + " to " + outputFileName);

} **catch** (IOException e) {

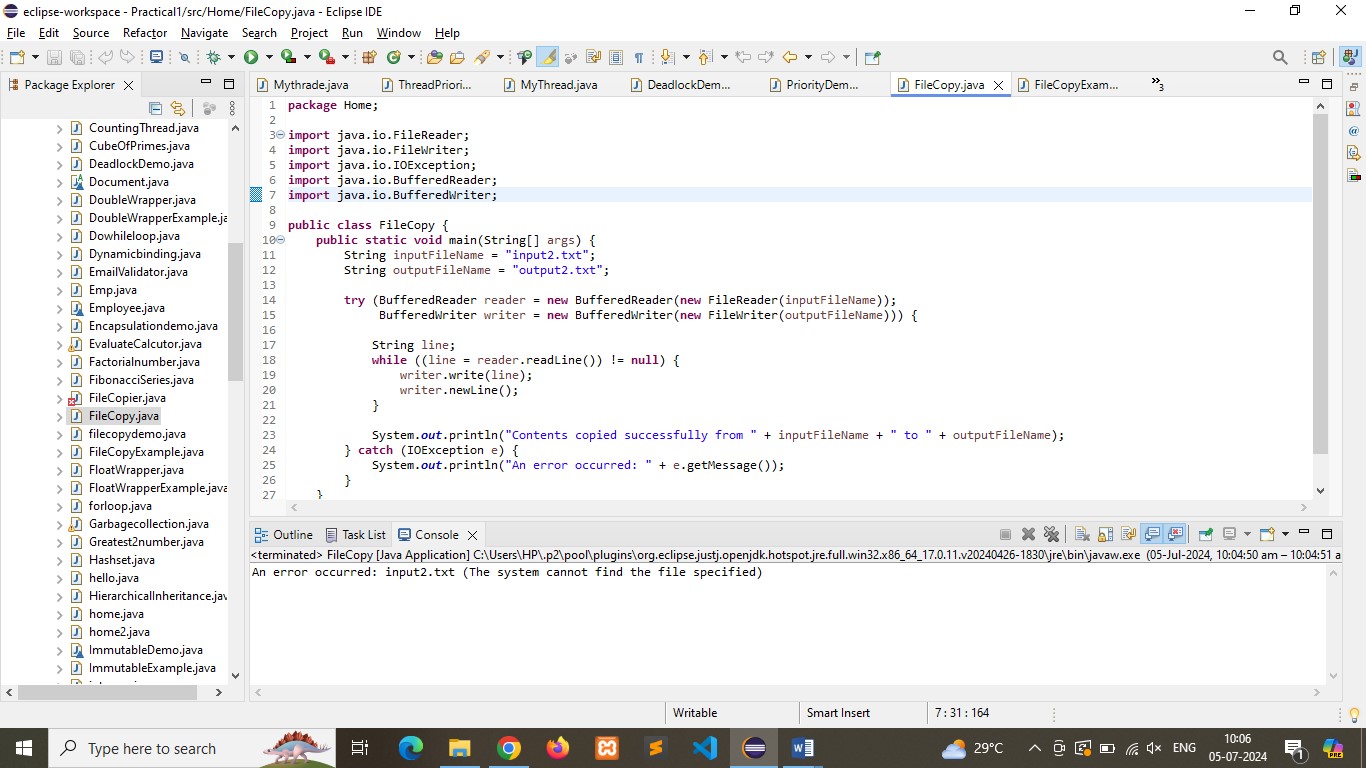
System.***out***.println("An error occurred: " + e.getMessage());

}

}

}

Output:



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

Program:

**package** Home;

**public** **class** MultithreadingExample {

**public** **static** **void** main(String[] args) {

String inputString = "multi threads";

// Create an array of threads

Thread[] threads = **new** Thread[inputString.length()];

// Create and start a thread for each character in the string

**for** (**int** i = 0; i < inputString.length(); i++) {

**final** **char** character = inputString.charAt(i);

threads[i] = **new** Thread(() -> *processCharacter*(character));

threads[i].start();

}

// Wait for all threads to complete

**for** (Thread thread : threads) {

**try** {

thread.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

// Method to process each character

**public** **static** **void** processCharacter(**char** c) {

System.***out***.println("Processing character: " + c + " on thread " + Thread.*currentThread*().getName());

// Simulate some processing time

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

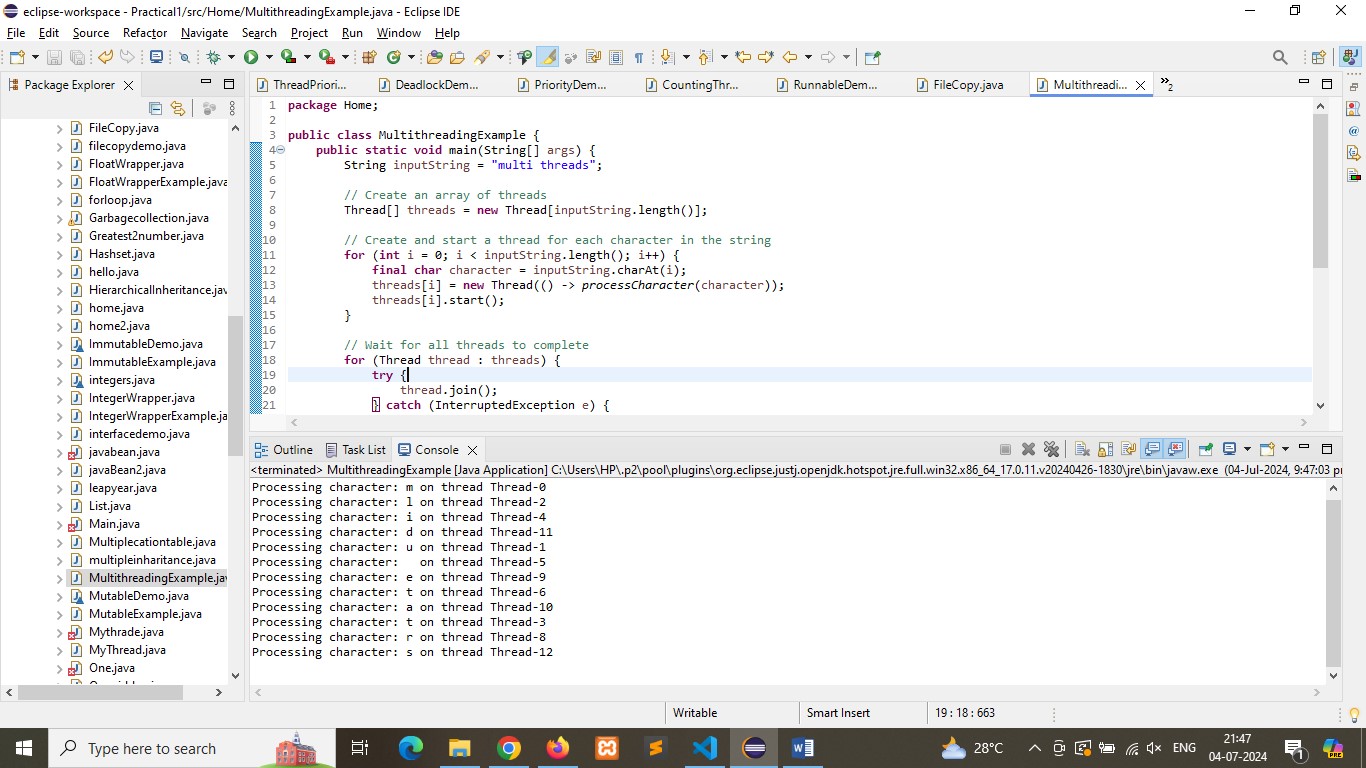
e.printStackTrace();

}

}

}

Output:



**Q-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.

Program:

**package** Home;

**public** **class** RunnableDemo **implements** Runnable {

**public** **void** run() {

**try** {

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

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

Thread.*sleep*(1000); // Delay of 1 second (1000 milliseconds)

}

} **catch** (InterruptedException e) {

System.***out***.println("Thread was interrupted.");

}

}

**public** **static** **void** main(String[] args) {

// Create an instance of the Runnable implementation

RunnableDemo runnableDemo = **new** RunnableDemo();

// Create a Thread object and pass the Runnable instance to it

Thread thread = **new** Thread(runnableDemo);

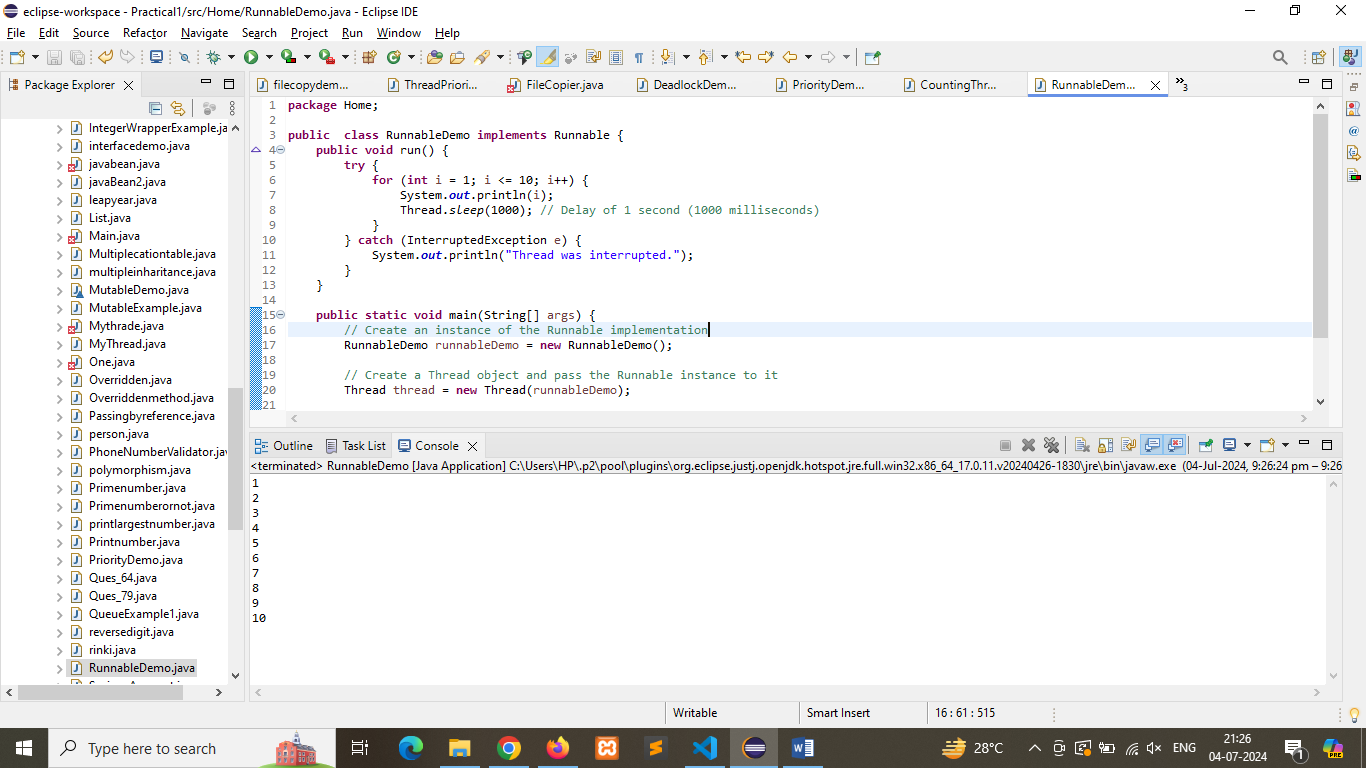
// Start the thread

thread.start();

}

}

Output:



**Q-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.

Program:

**package** Home;

**public** **class** CountingThread **extends** Thread {

**public** CountingThread(String name) {

**super**(name);

}

**public** **void** run() {

**try** {

**for** (**int** i = 1; i <= 5; i++) {

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

Thread.*sleep*(500); // Delay of 500 milliseconds

}

} **catch** (InterruptedException e) {

System.***out***.println(Thread.*currentThread*().getName() + " was interrupted.");

}

}

**public** **static** **void** main(String[] args) {

// Create three threads

CountingThread thread1 = **new** CountingThread("Thread-1");

CountingThread thread2 = **new** CountingThread("Thread-2");

CountingThread thread3 = **new** CountingThread("Thread-3");

// Start the threads

thread1.start();

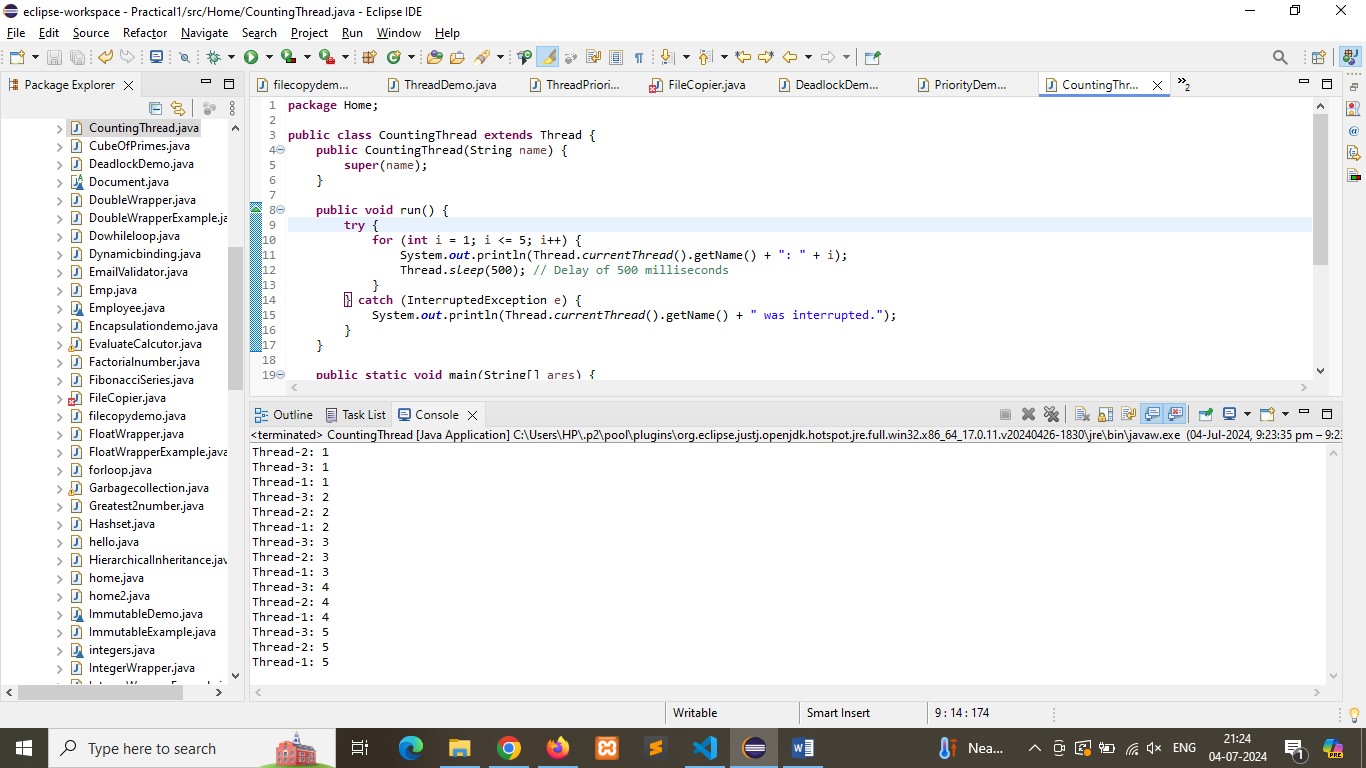
thread2.start();

thread3.start();

}

}

Output:



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

Program:

**package** Home;

**public** **class** PriorityDemo **extends** Thread {

**public** PriorityDemo(String name) {

**super**(name);

}

**public** **void** run() {

**for** (**int** i = 1; i <= 5; i++) {

System.***out***.println(Thread.*currentThread*().getName() + " with priority " +

Thread.*currentThread*().getPriority() + " is running. Iteration: " + i);

}

}

**public** **static** **void** main(String[] args) {

// Create threads

PriorityDemo thread1 = **new** PriorityDemo("Thread-1");

PriorityDemo thread2 = **new** PriorityDemo("Thread-2");

PriorityDemo thread3 = **new** PriorityDemo("Thread-3");

// Set priorities

thread1.setPriority(Thread.***MIN\_PRIORITY***); // Priority 1

thread2.setPriority(Thread.***NORM\_PRIORITY***); // Priority 5

thread3.setPriority(Thread.***MAX\_PRIORITY***); // Priority 10

// Start threads

thread1.start();

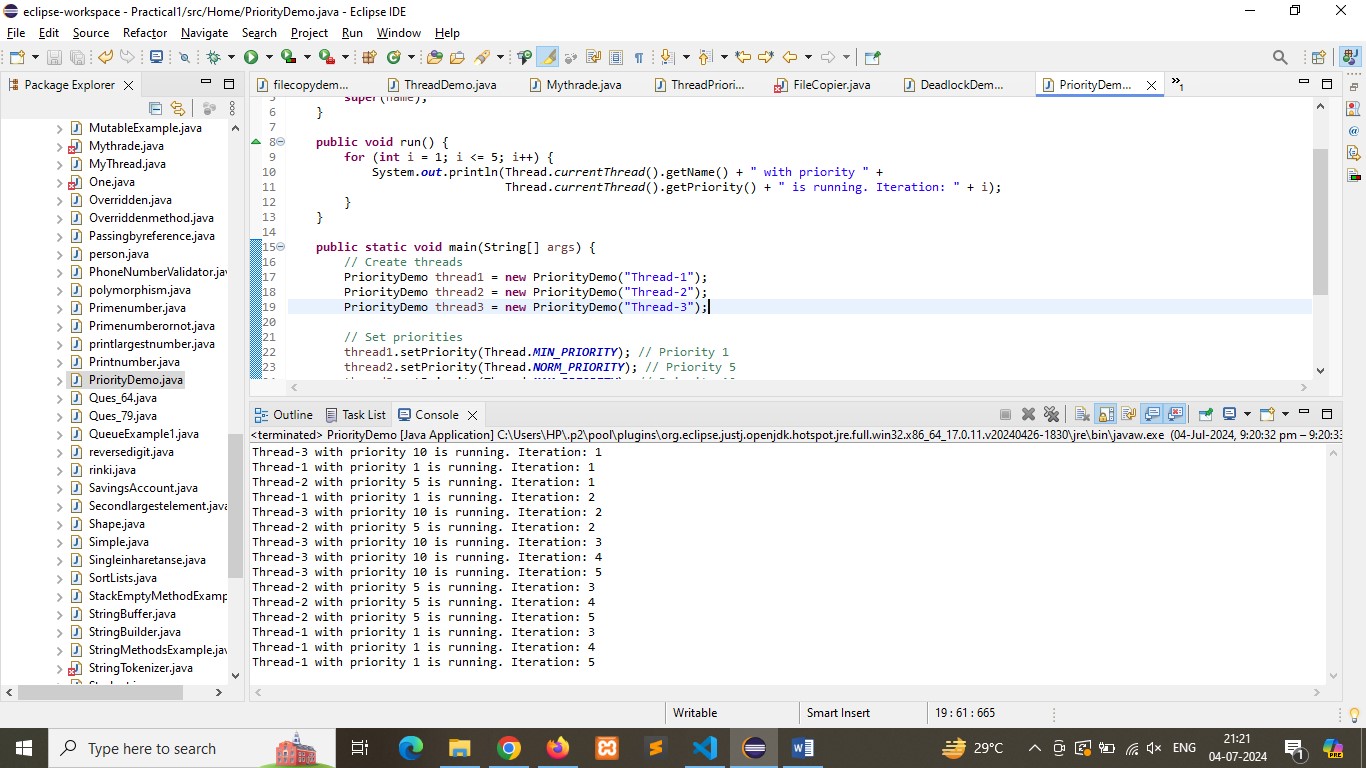
thread2.start();

thread3.start();

}

}

Output:



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

Program:

**package** Home;

**public** **class** DeadlockDemo {

// Resource 1

**private** **final** Object resource1 = **new** Object();

// Resource 2

**private** **final** Object resource2 = **new** Object();

**public** **static** **void** main(String[] args) {

DeadlockDemo deadlockDemo = **new** DeadlockDemo();

// Thread 1

Thread t1 = **new** Thread(() -> deadlockDemo.method1());

// Thread 2

Thread t2 = **new** Thread(() -> deadlockDemo.method2());

t1.start();

t2.start();

}

// Method for Thread 1

**public** **void** method1() {

**synchronized** (resource1) {

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

// Adding a sleep to ensure that Thread 2 locks resource 2

**try** { Thread.*sleep*(50); } **catch** (InterruptedException e) {}

**synchronized** (resource2) {

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

}

}

}

// Method for Thread 2

**public** **void** method2() {

**synchronized** (resource2) {

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

// Adding a sleep to ensure that Thread 1 locks resource 1

**try** { Thread.*sleep*(50); } **catch** (InterruptedException e) {}

**synchronized** (resource1) {

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

}

}

}

}

Output:

