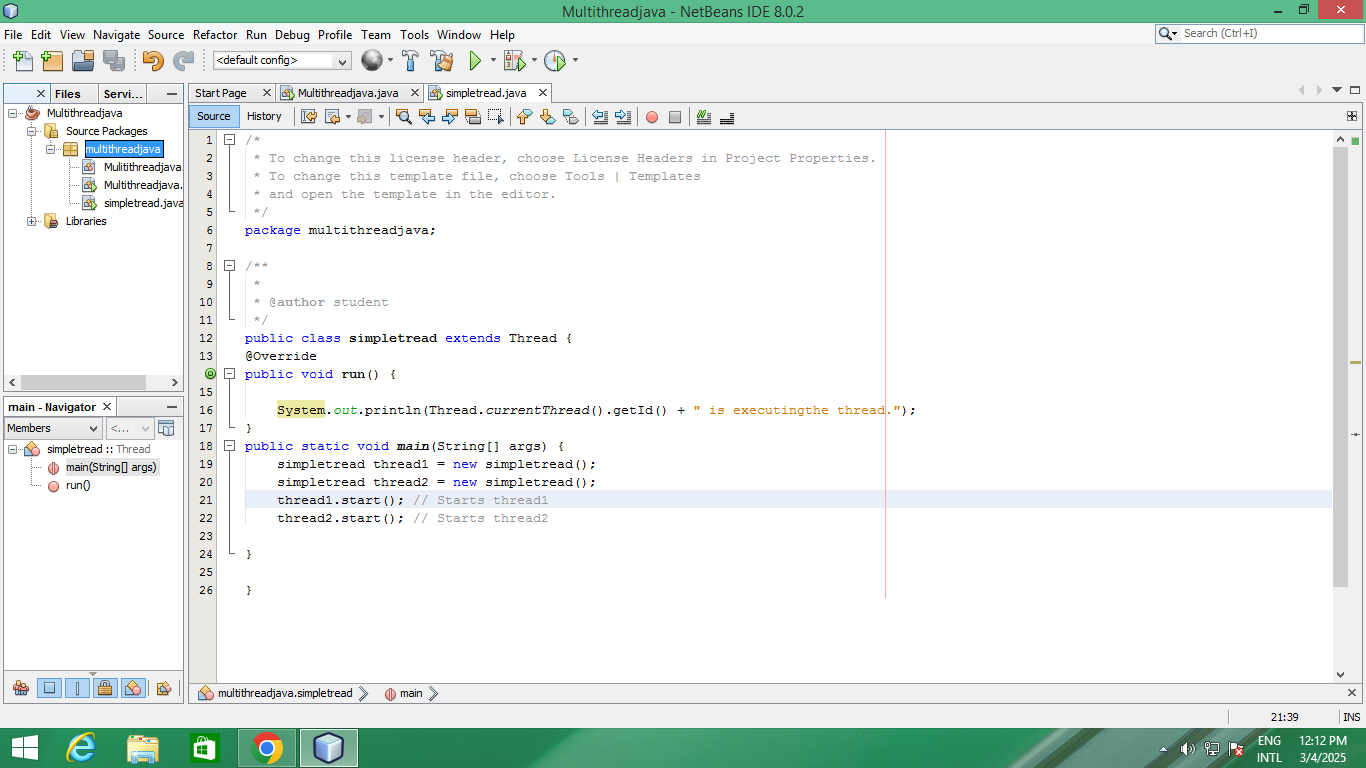
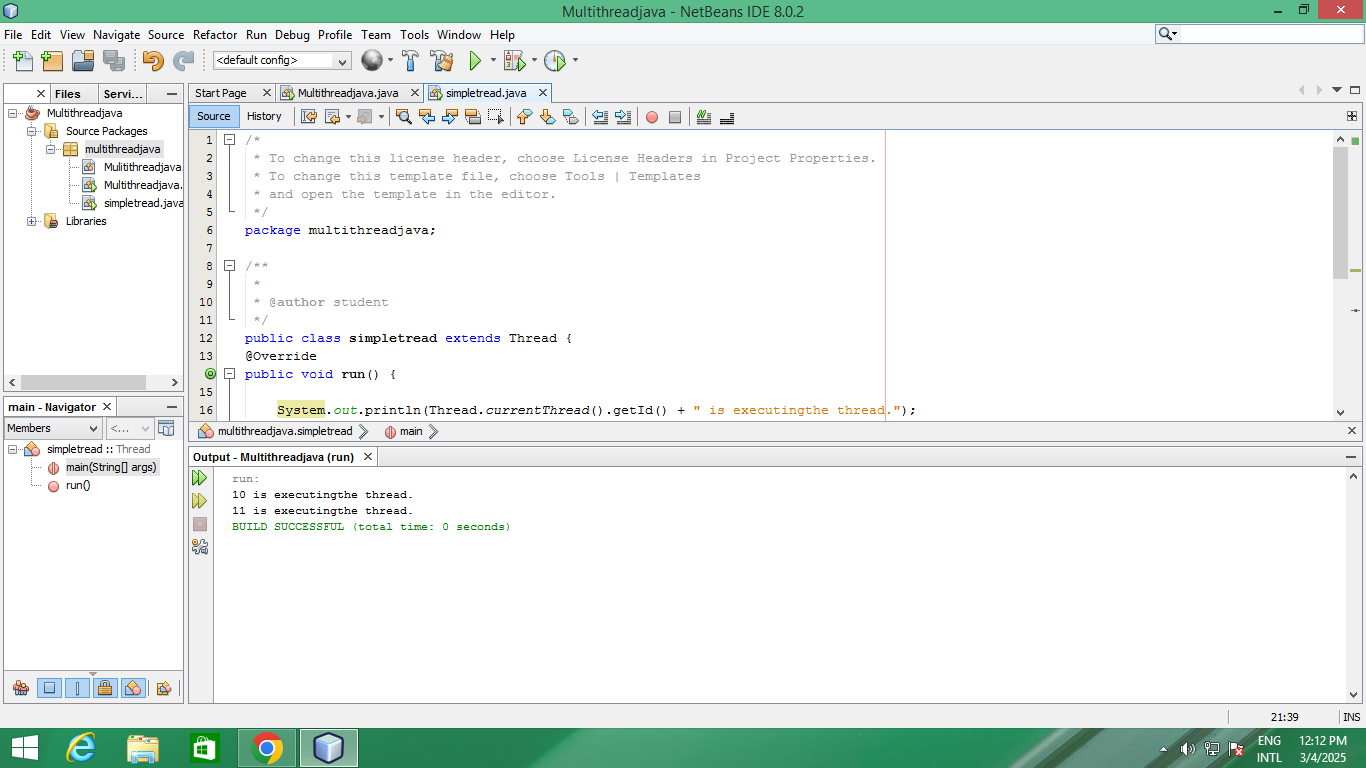
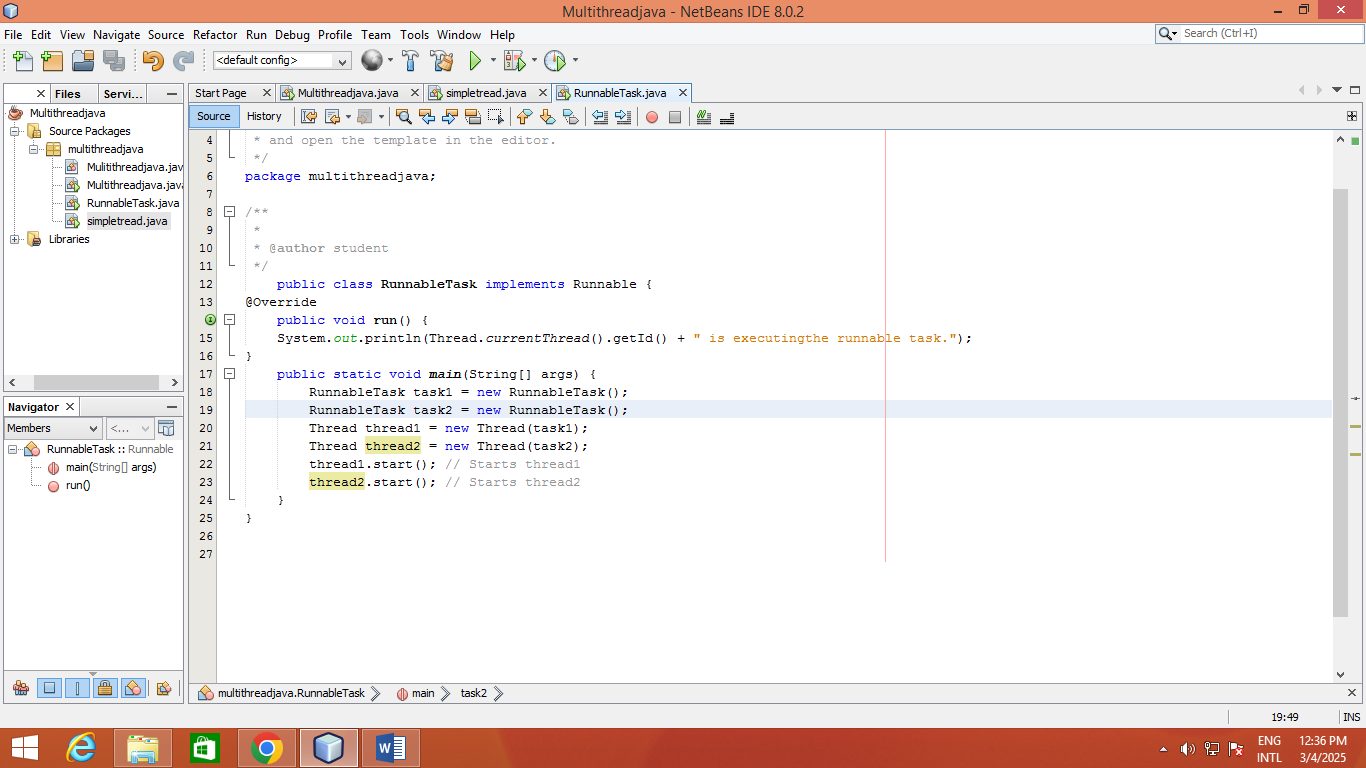
EA/LAB 02/0108

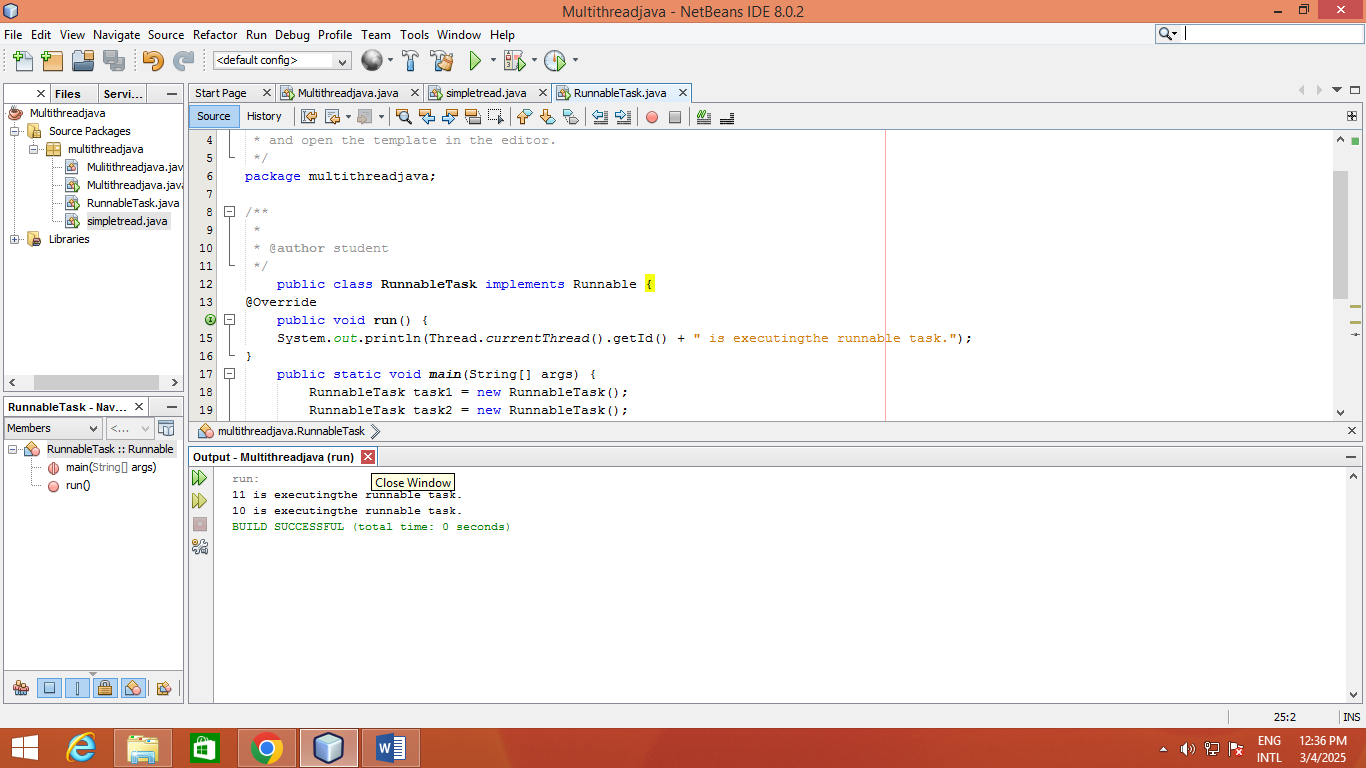
Labsheet-01

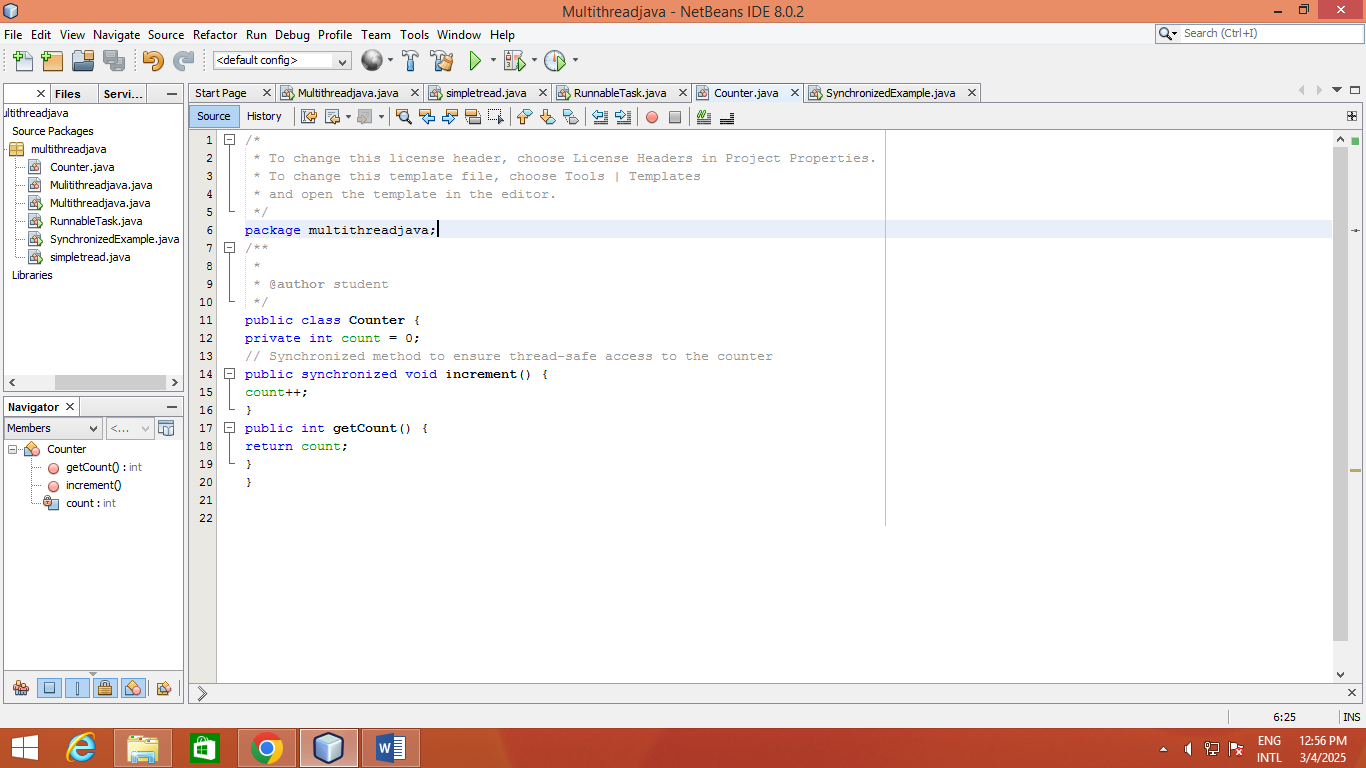


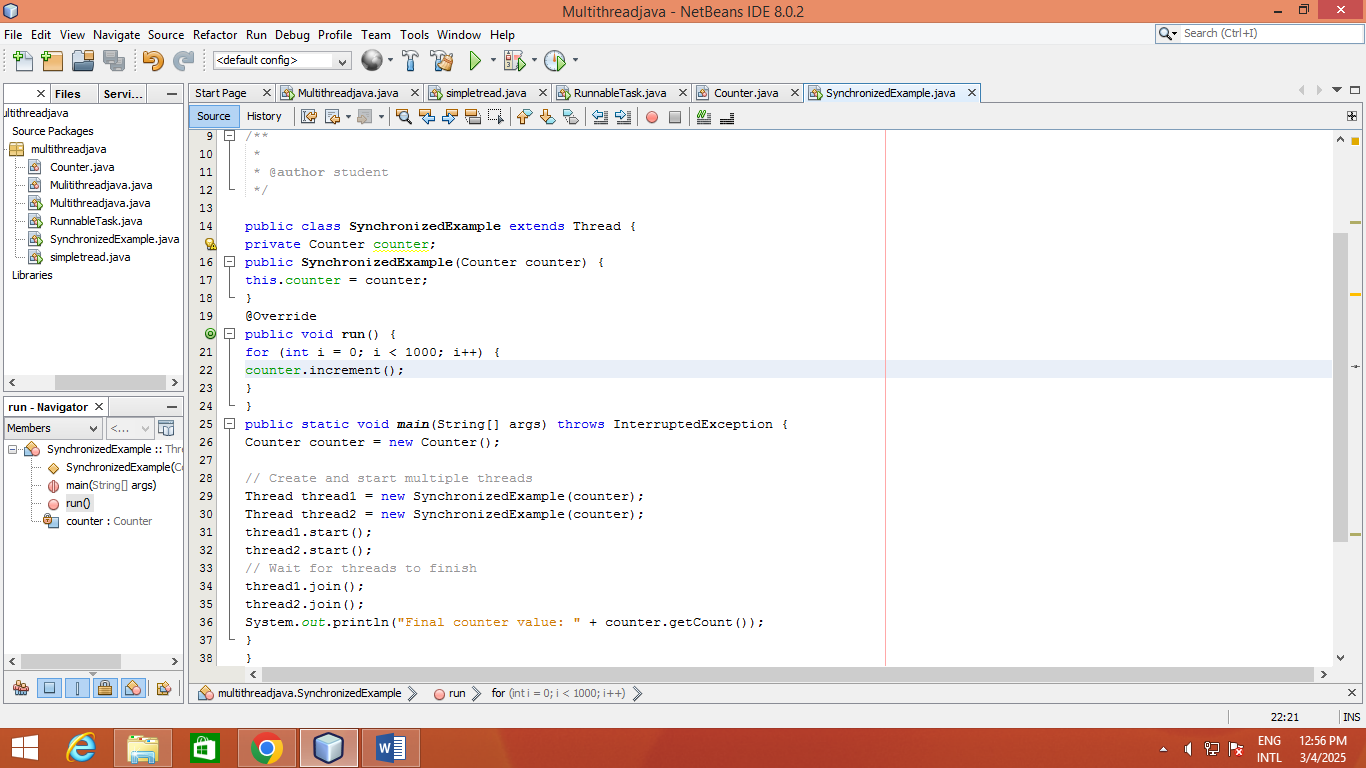
Output

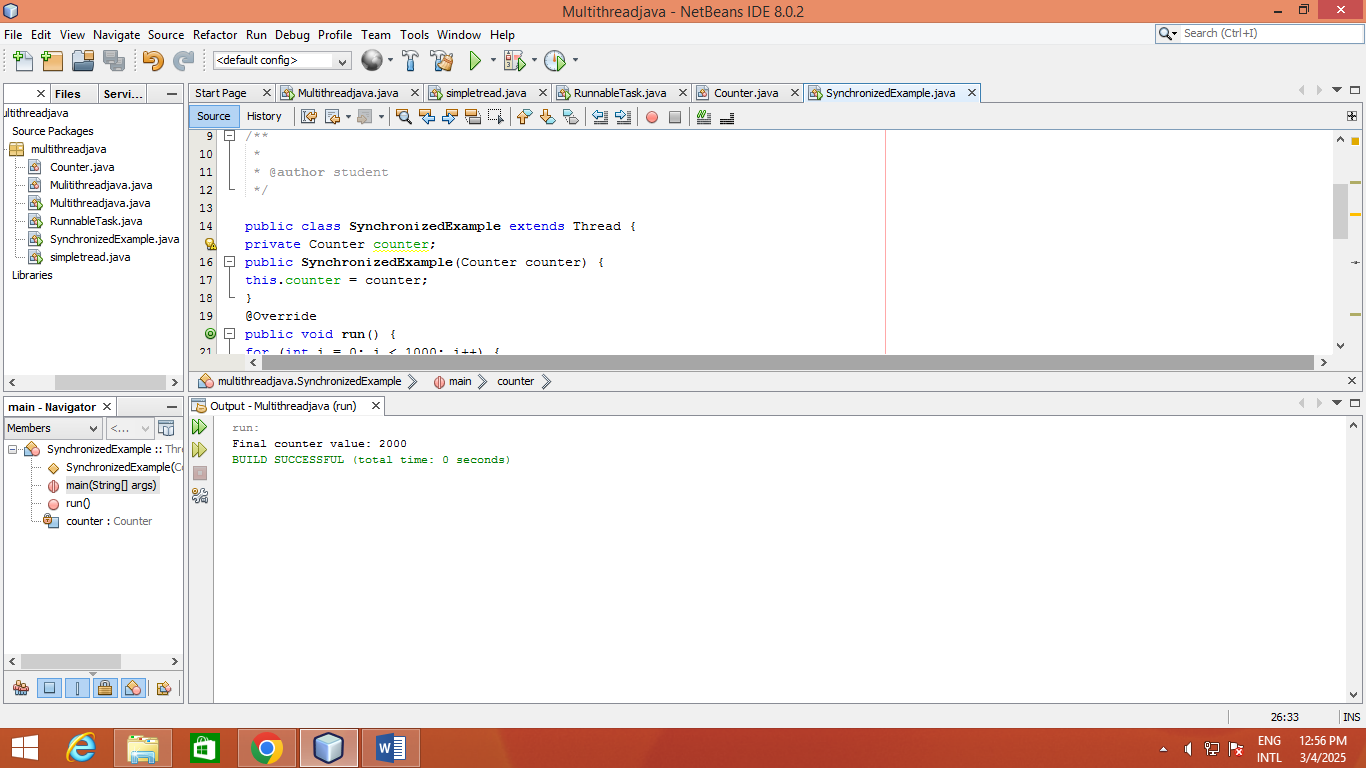


Out put









Lesson 04

package multithreadjava;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class ThreadPoolExample {

static class Task implements Runnable { // Make Task static

private int taskId;

public Task(int taskId) {

this.taskId = taskId;

}

@Override

public void run() {

System.out.println("Task " + taskId + " is being processed by " +

Thread.currentThread().getName());

}

}

public static void main(String[] args) {

// Create a thread pool with 3 threads

ExecutorService executorService = Executors.newFixedThreadPool(3);

// Submit tasks to the pool

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

executorService.submit(new Task(i)); // Task can now be instantiated directly

}

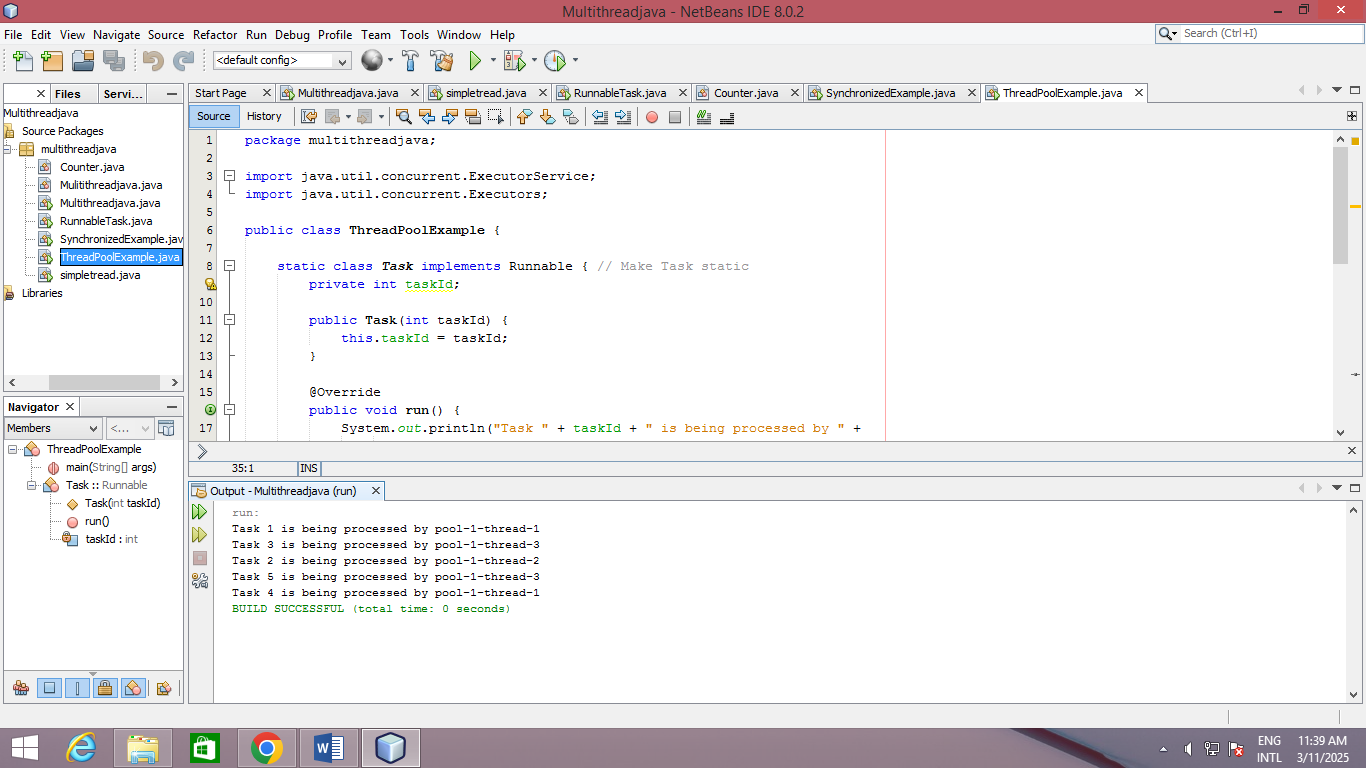
// Shutdown the thread pool

executorService.shutdown();

}

}

OUT PUT



Lesson 05

package multithreadjava;

public class ThreadLifecycleExample extends Thread {

@Override

public void run() {

System.out.println(Thread.currentThread().getName() + " - State: " +

Thread.currentThread().getState());

try {

Thread.sleep(2000); // Simulate waiting state

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName() + " - State aftersleep: " + Thread.currentThread().getState());

}

public static void main(String[] args) {

ThreadLifecycleExample thread = new ThreadLifecycleExample();

System.out.println(thread.getName() + " - State before start: " +

thread.getState());

thread.start(); // Start the thread

System.out.println(thread.getName() + " - State after start: " +

thread.getState());

}

}

