EA/LAB 02/0078

Lab sheet 02

**01**

**MultiThreading Code**

public class SimpleThread extends Thread {

@Override

public void run() {

System.out.println(Thread.currentThread().getId() + " is executingthe thread.");

}

public static void main(String[] args) {

SimpleThread thread1 = new SimpleThread();

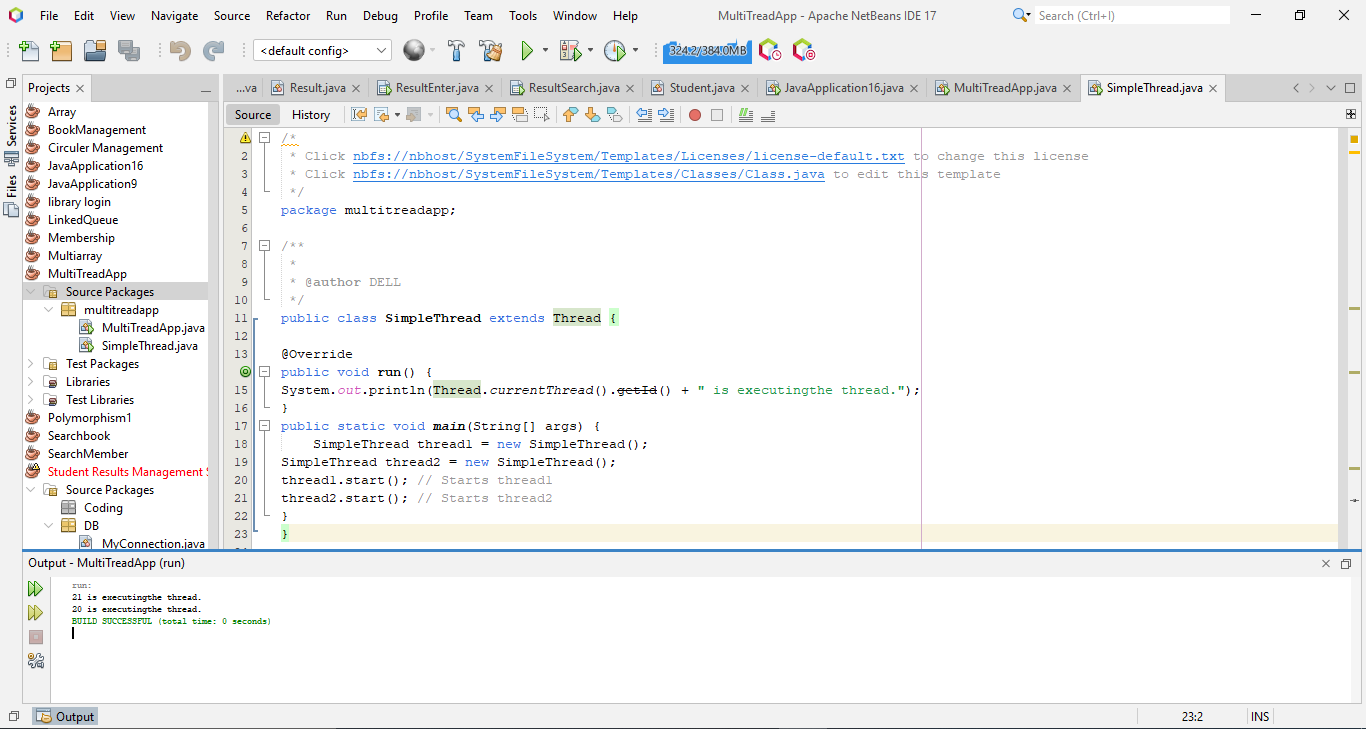
SimpleThread thread2 = new SimpleThread();

thread1.start(); // Starts thread1

thread2.start(); // Starts thread2

}

}

**Output**

**02**

**Runnable** **Task** **Code**

public class RunnableTask implements Runnable {

@Override

public void run() {

System.out.println(Thread.currentThread().getId() + " is executingthe runnable task.");

}

public static void main(String[] args) {

RunnableTask task1 = new RunnableTask();

RunnableTask task2 = new RunnableTask();

Thread thread1 = new Thread(task1);

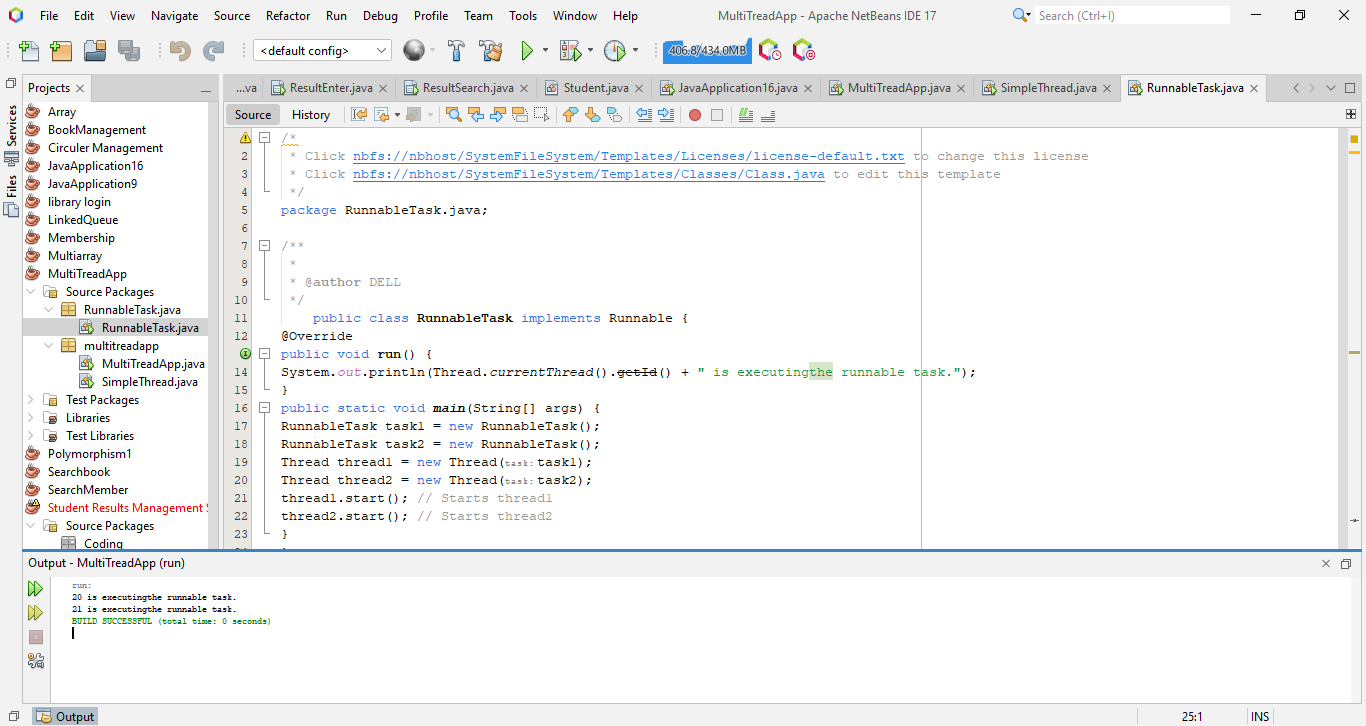
Thread thread2 = new Thread(task2);

thread1.start(); // Starts thread1

thread2.start(); // Starts thread2

}

}

**Output**

**03**

**Counter**

public class Counter {

private int count = 0;

// Synchronized method to ensure thread-safe access to the counter

public synchronized void increment() {

count++;

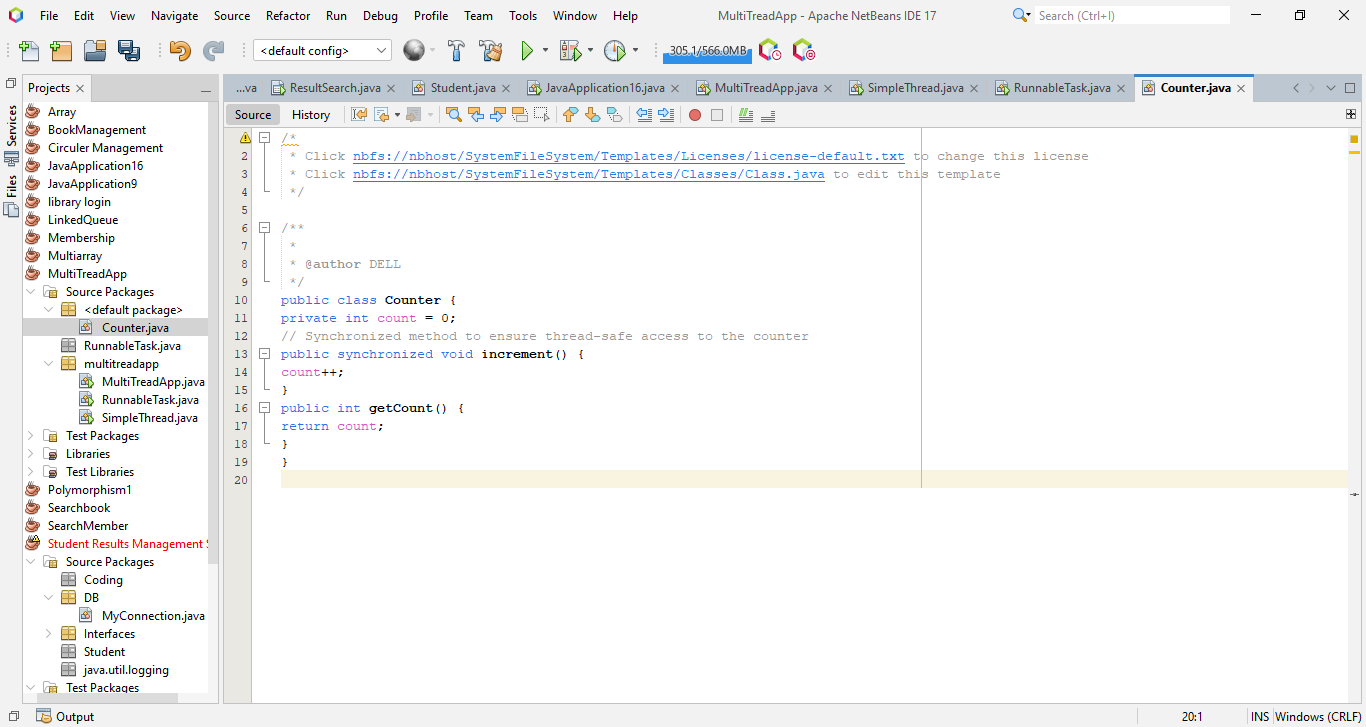
}

public int getCount() {

return count;

}

}

****

**SynchronizedExample**

**code**

public class SynchronizedExample extends Thread {

private Counter counter;

public SynchronizedExample(Counter counter) {

this.counter = counter;

}

@Override

public void run() {

for (int i = 0; i < 1000; i++) {

counter.increment();

}

}

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

Counter counter = new Counter();

// Create and start multiple threads

Thread thread1 = new SynchronizedExample(counter);

Thread thread2 = new SynchronizedExample(counter);

thread1.start();

thread2.start();

// Wait for threads to finish

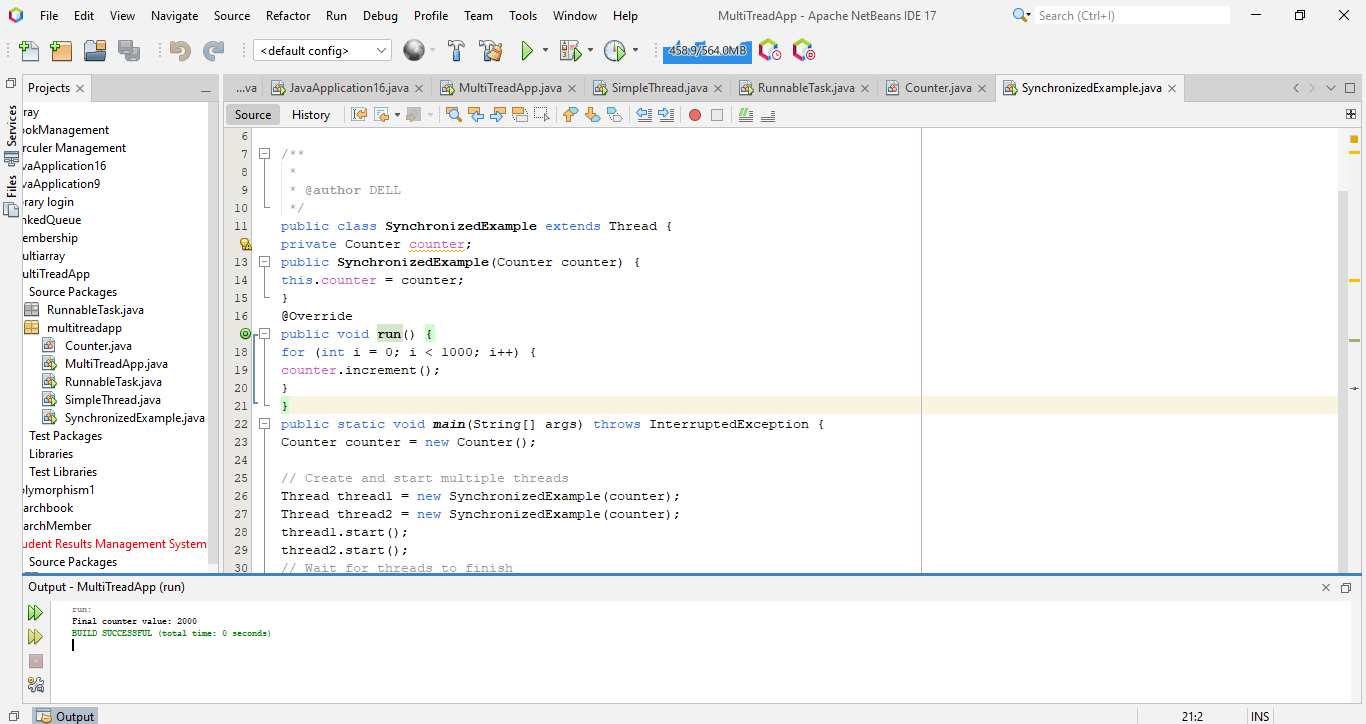
thread1.join();

thread2.join();

System.out.println("Final counter value: " + counter.getCount());

}

}

**output**