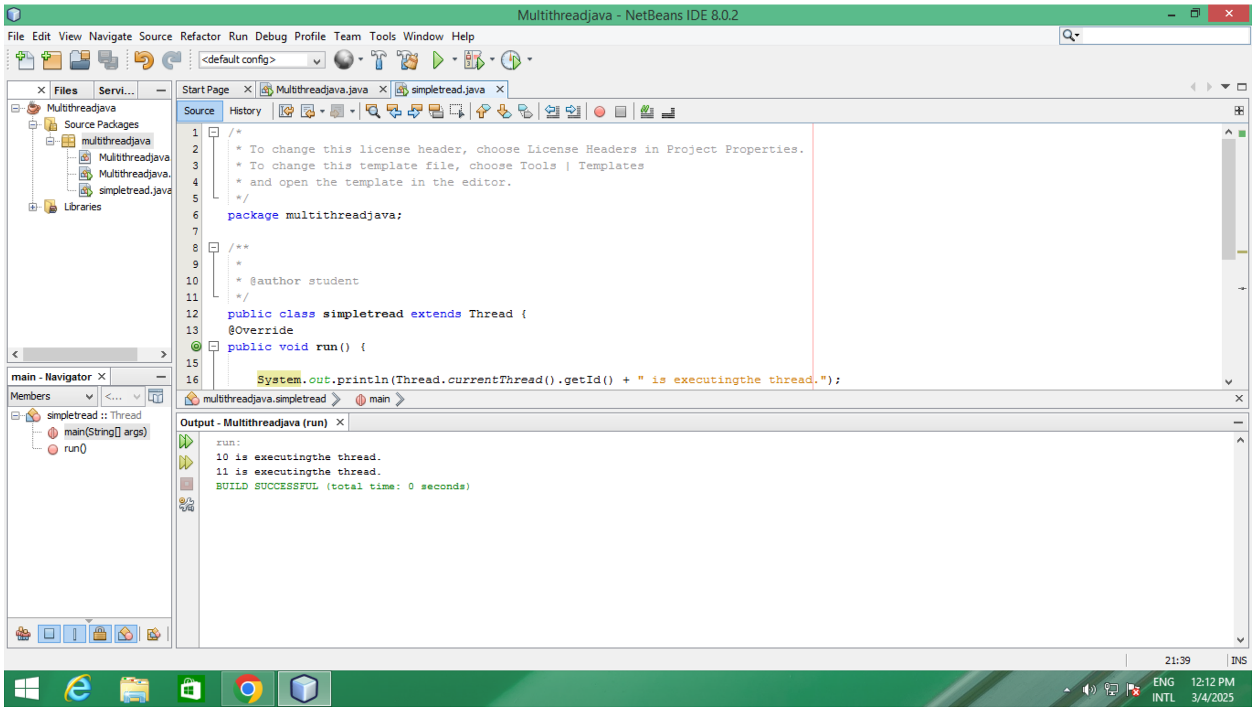
EA/LAB 02/0108

Lab sheet-01

```
public class simpletread extends Thread {
@Override
public void run() {
System.out.println(Thread.currentThread().getId()  +  " is executing the thread.");
}
public static void main (String [] args) {
Simpletread thread1 = new Simpletread();
Simpletread thread2 = new Simpletread();
thread1.start();
thread2.start();
   }
   }
```

Output

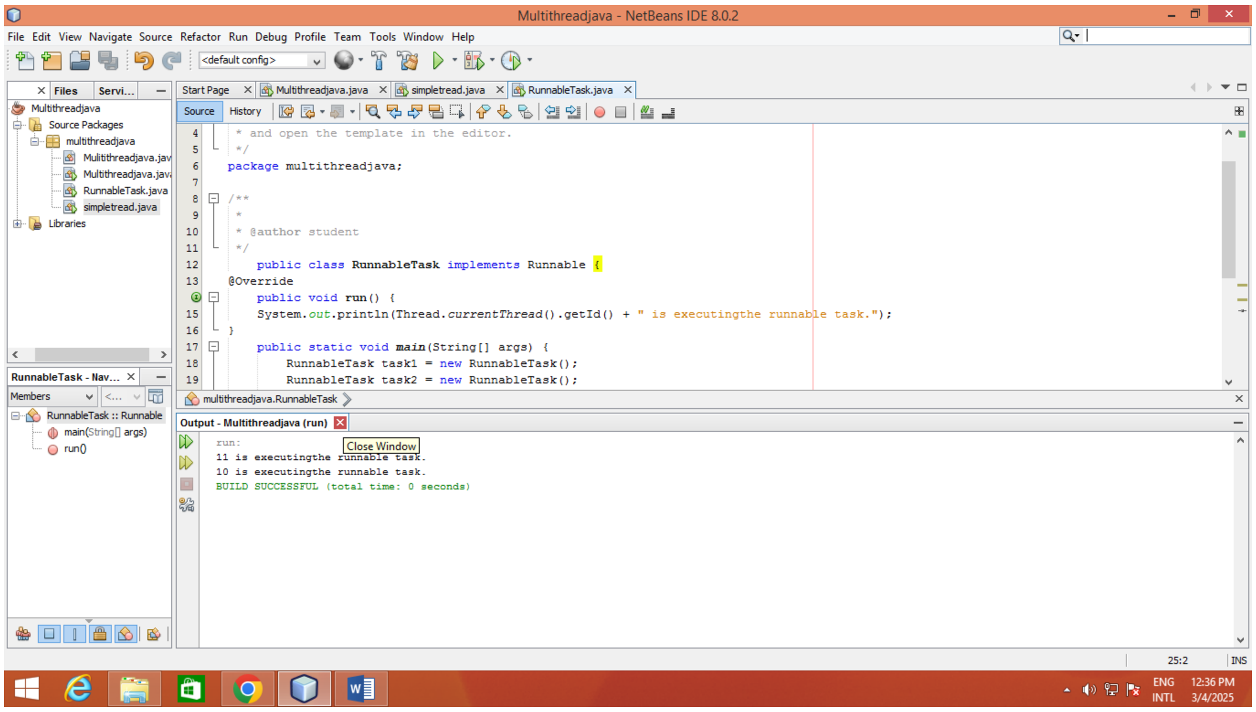


```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package multithreadjava;

/**
 *
 * @author student
 */
public class simpletread extends Thread {
@Override
public void run() {

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

```
run:
10 is executingthe thread.
11 is executingthe thread.
BUILD SUCCESSFUL (total time: 0 seconds)
```

2) Lab02- TASK 02

```
public class RunnableTask implements Runnable {

public void run() {

System.out.println(Thread.currentThread().getId() + " is executing the 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();

thread2.start();

   }

   }
```



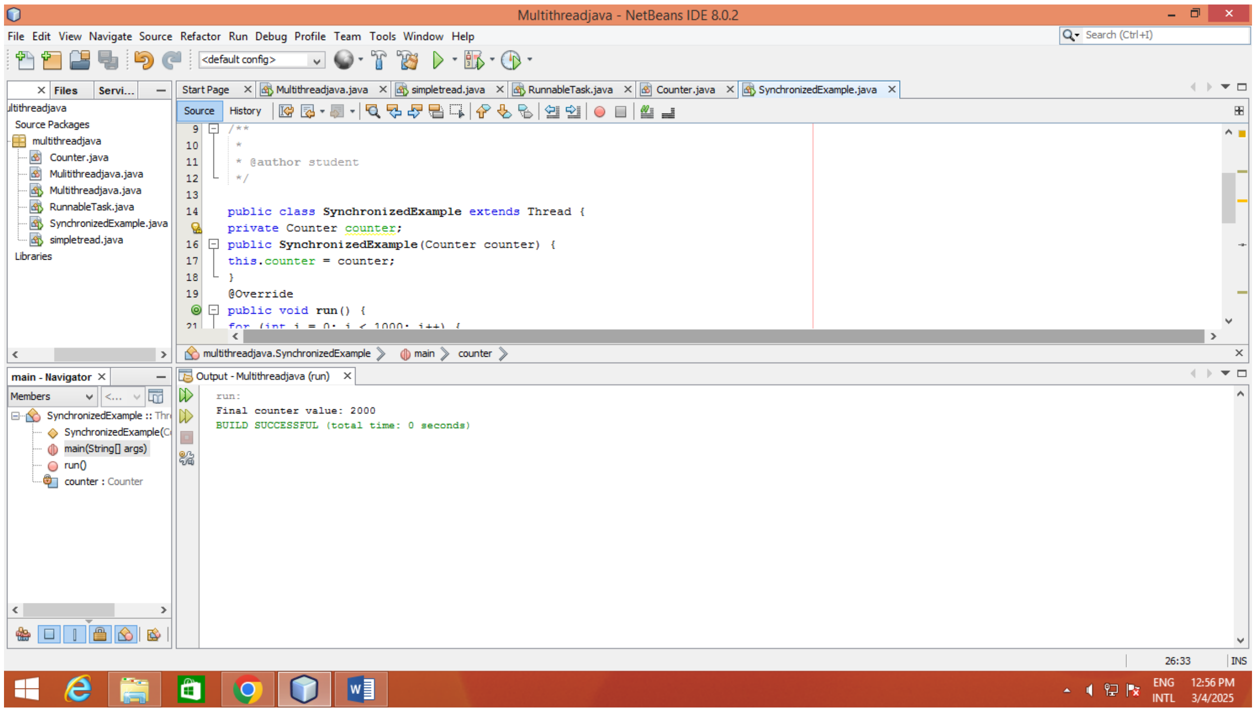
```
 * and open the template in the editor.
 */
package multithreadjava;

/**
 *
 * @author student
 */
    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();
```

```
run:
11 is executingthe runnable task.
10 is executingthe runnable task.
BUILD SUCCESSFUL (total time: 0 seconds)
```

3) LAB03-TASK-03

```
public class Counter {

privateint count = 0;

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

public synchronized void increment() {

count++;

}

Public int getCount() {

return count;  }


public class synchronizedExample extends Thread{

private Counter counter;

publicSynchronizedExample(Counter counter) {

this.counter = counter;  }

@Override

public void run() {

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

counter.increment();

}}}

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

Counter counter = new Counter;

Thread thread1 = new SynchronizedExample(counter);

Thread thread2 = new SynchronizedExample(counter);

thread1.start();

thread2.start();


thread1.join();

thread2.join();

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

Out put



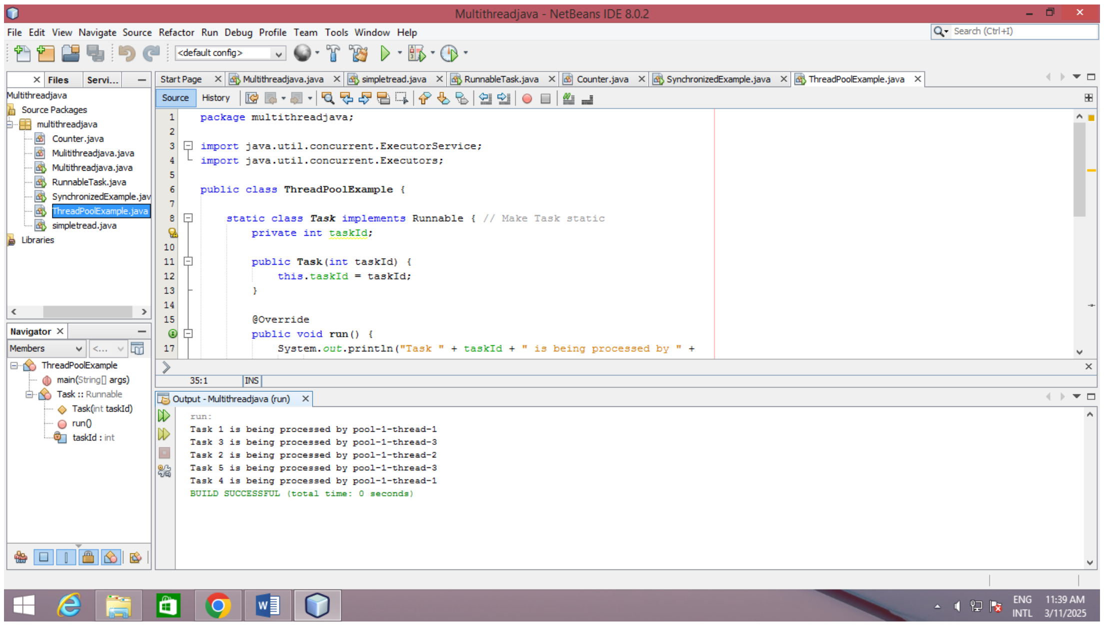
```
/**
 *
 * @author student
 */

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++) {
```

```
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 0 seconds)
```

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



```
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 " +
```

```
run:
Task 1 is being processed by pool-1-thread-1
Task 3 is being processed by pool-1-thread-3
Task 2 is being processed by pool-1-thread-2
Task 5 is being processed by pool-1-thread-3
Task 4 is being processed by pool-1-thread-1
BUILD SUCCESSFUL (total time: 0 seconds)
```

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();

}

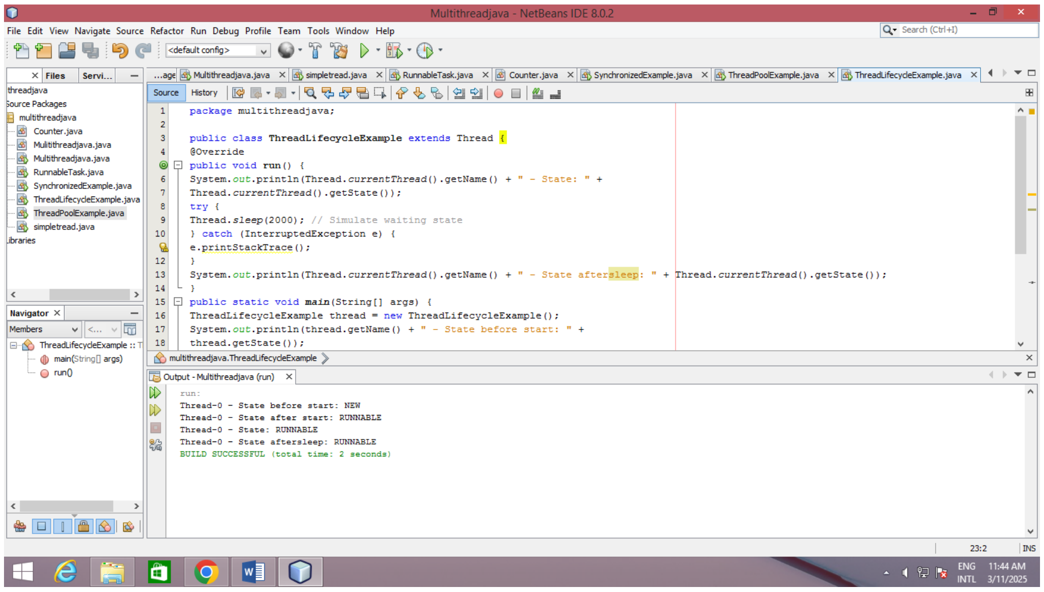
System.out.println(Thread.currentThread().getName() + " - State aftersleep: " +
Thread.currentThread().getState());

}

public static void main(String[] args) {

```java
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());
}
}
```

Out put



```java
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());
```

```
run:
Thread-0 - State before start: NEW
Thread-0 - State after start: RUNNABLE
Thread-0 - State: RUNNABLE
Thread-0 - State aftersleep: RUNNABLE
BUILD SUCCESSFUL (total time: 2 seconds)
```