Java Thread

Runnable Class

package eractivity; 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();

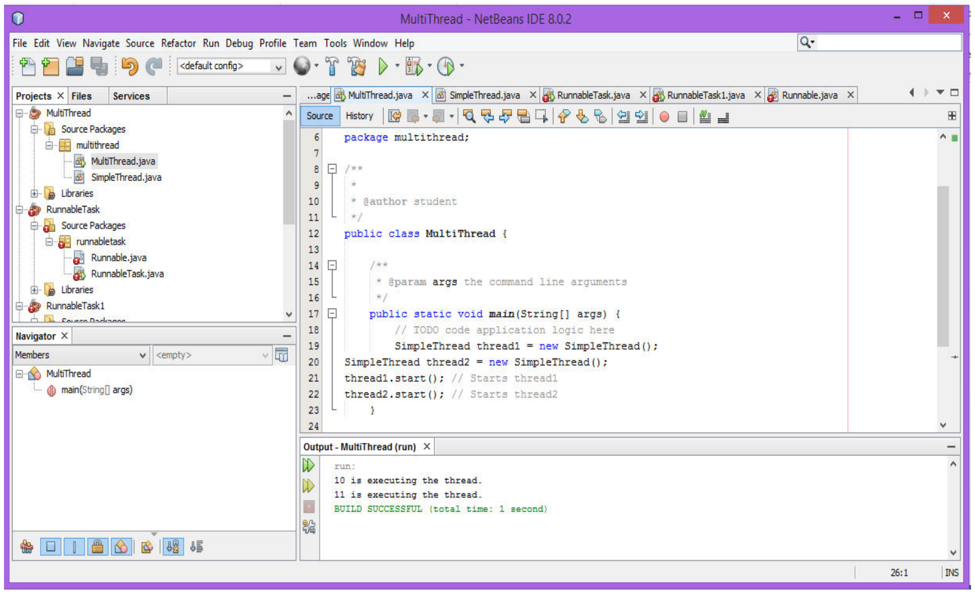
Thread thread1 = new Thread(task1); Thread

thread2 = new Thread(task2); thread1.start(); //

Starts thread1 thread2.start(); // Starts thread2

}

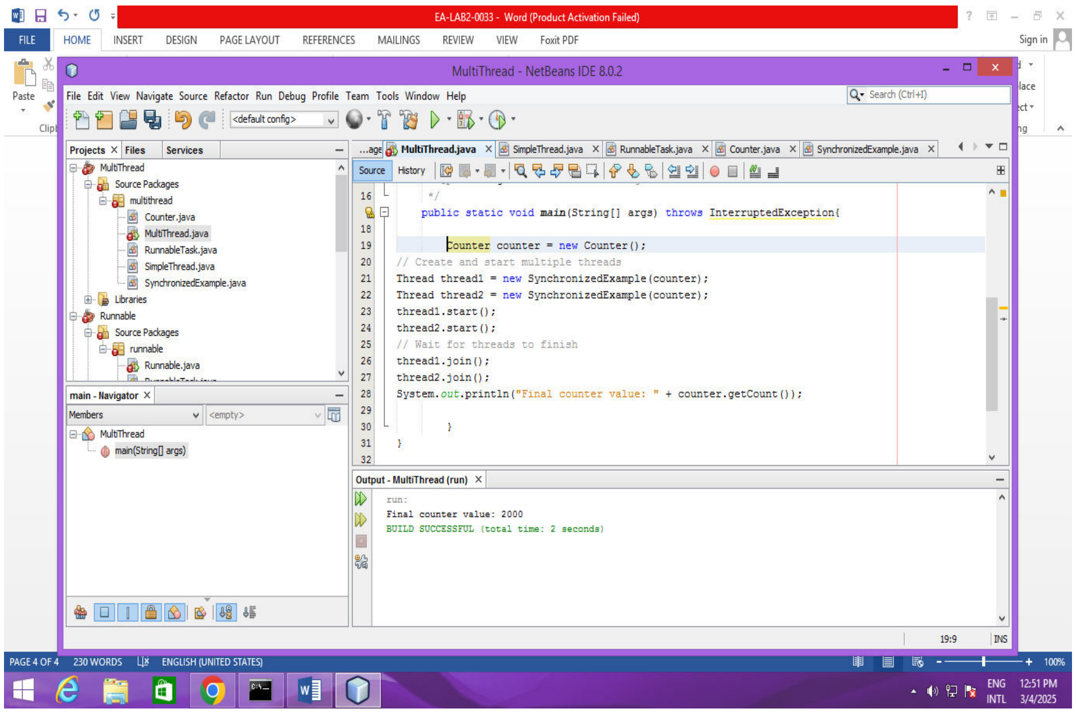
}


MultiThread - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
MultiThread.java SimpleThread.java RunnableTask.java RunnableTask1.java Runnable.java
Source History
Output - MultiThread (run)
26:1 INS

```
package multithread;

/**
 *
 * @author student
 */
public class MultiThread {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start(); // Starts thread1
thread2.start(); // Starts thread2
    }
```

```
run:
10 is executing the thread.
11 is executing the thread.
BUILD SUCCESSFUL (total time: 1 second)
```

```
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;
 }
}

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()); }}
```

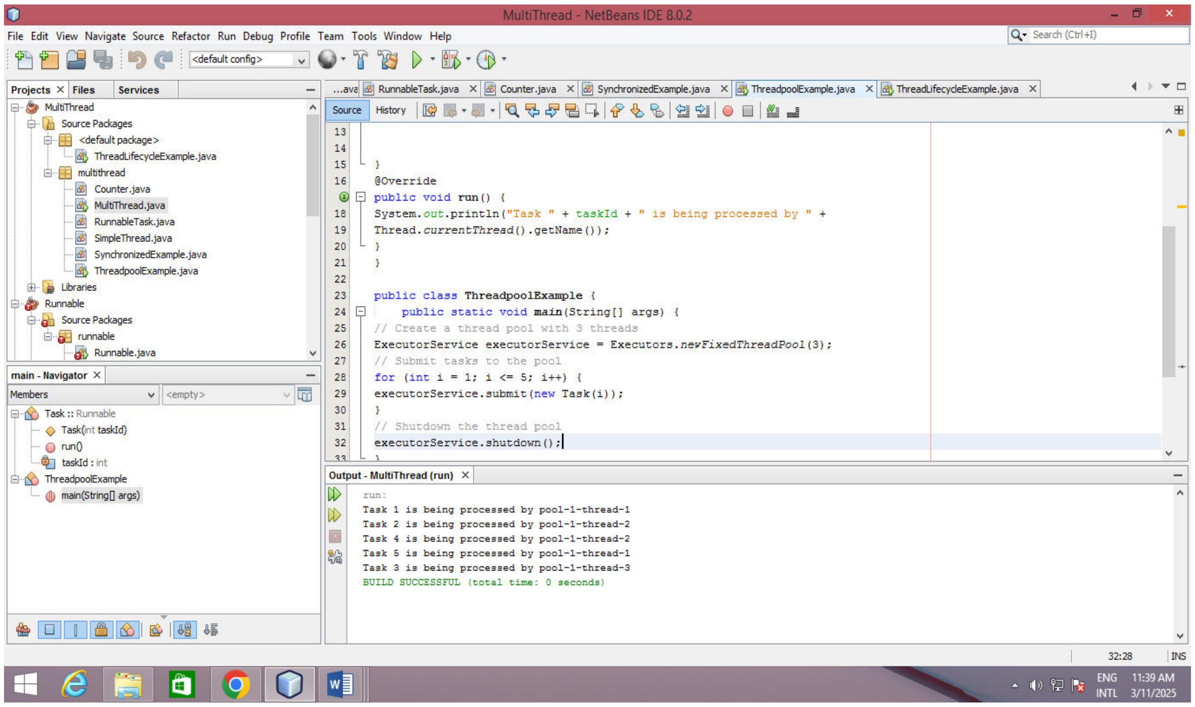


```
     */
    public static void main(String[] args) throws InterruptedException{

        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());

        }
}
```

```
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 2 seconds)
```

Thread Pooling

```
 import java.util.concurrent.ExecutorService; import
java.util.concurrent.Executors;

    class Task implements Runnable {
    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 class ThreadPoolExample {

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));
}
// Shutdown the thread pool executorService.shutdown();
}

}
```



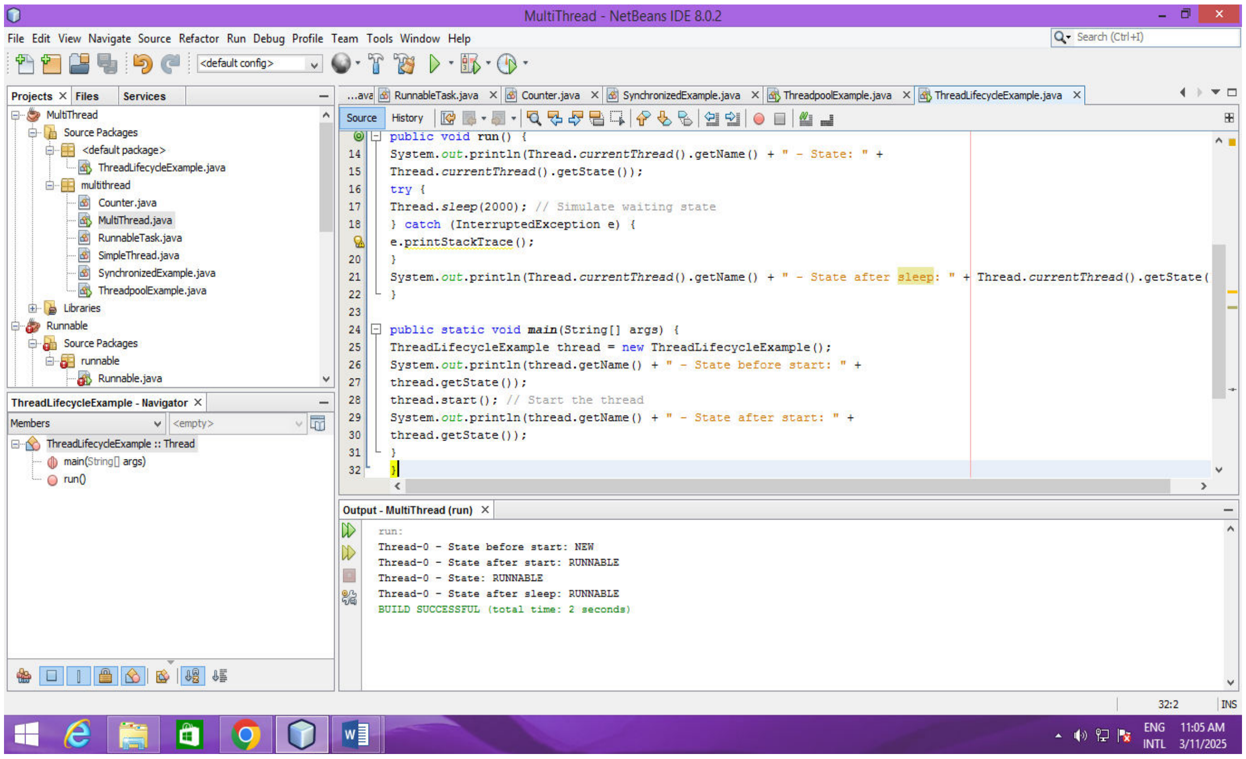
```
    }
    @Override
    public void run() {
    System.out.println("Task " + taskId + " is being processed by " +
    Thread.currentThread().getName());
    }
    }

    public class ThreadpoolExample {
        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));
    }
    // Shutdown the thread pool
    executorService.shutdown();
```

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

Thread Lifecycle and States

```
public class ThreadLifecycleExample extends Thread{
  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 after sleep: " +
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());

}

}
```


MultiThread - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects Files Services
MultiThread
Source Packages
<default package>
ThreadLifecycleExample.java
multithread
Counter.java
MultiThread.java
RunnableTask.java
SimpleThread.java
SynchronizedExample.java
ThreadpoolExample.java
Libraries
Runnable
Source Packages
runnable
Runnable.java
ThreadLifecycleExample - Navigator
Members
<empty>
ThreadLifecycleExample :: Thread
main(String[] args)
run()
...ava RunnableTask.java Counter.java SynchronizedExample.java ThreadpoolExample.java ThreadLifecycleExample.java
Source History
Output - MultiThread (run)
32:2
INS
ENG INTL 11:05 AM 3/11/2025

```
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 after sleep: " + 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());
}
}
```

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