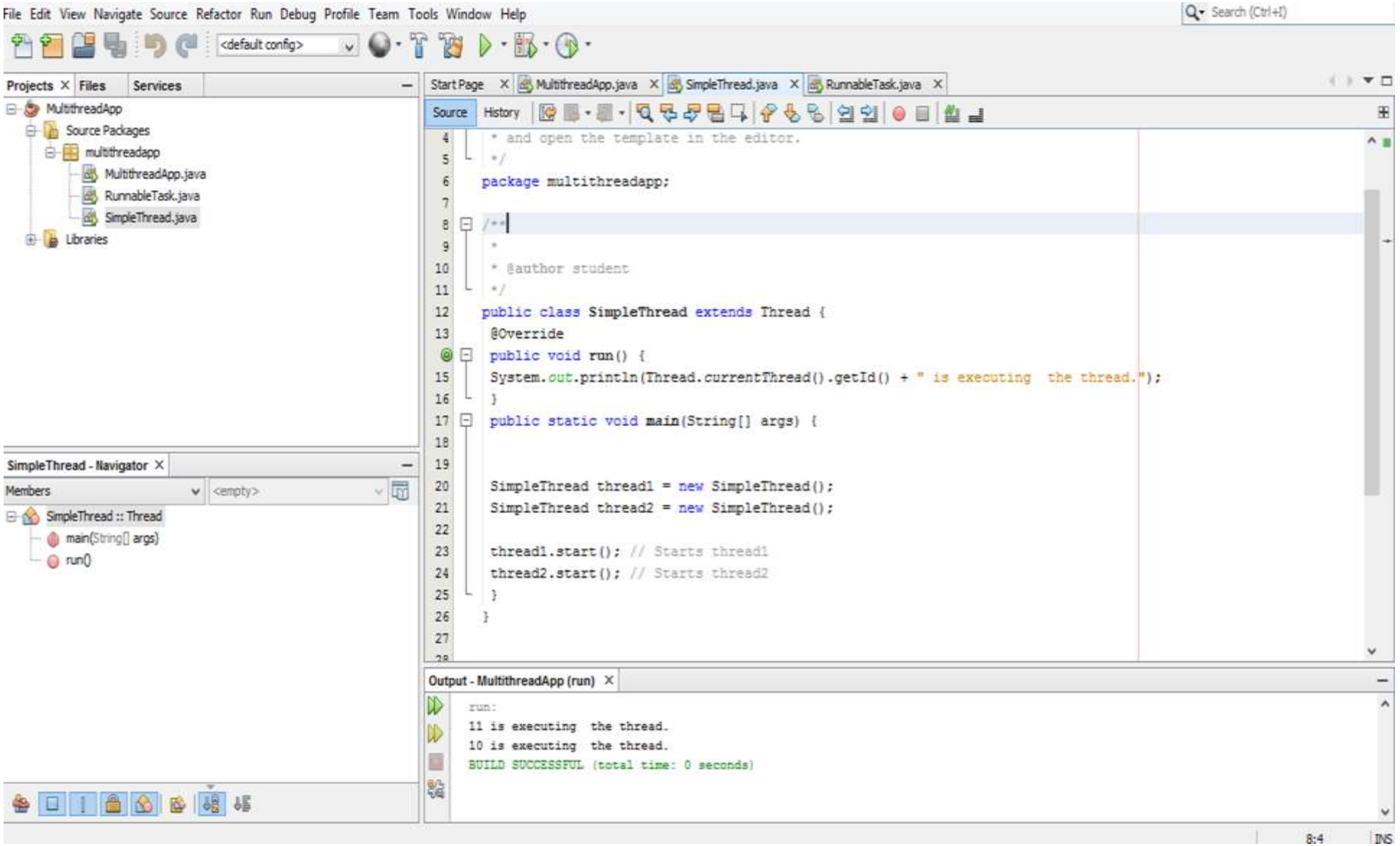
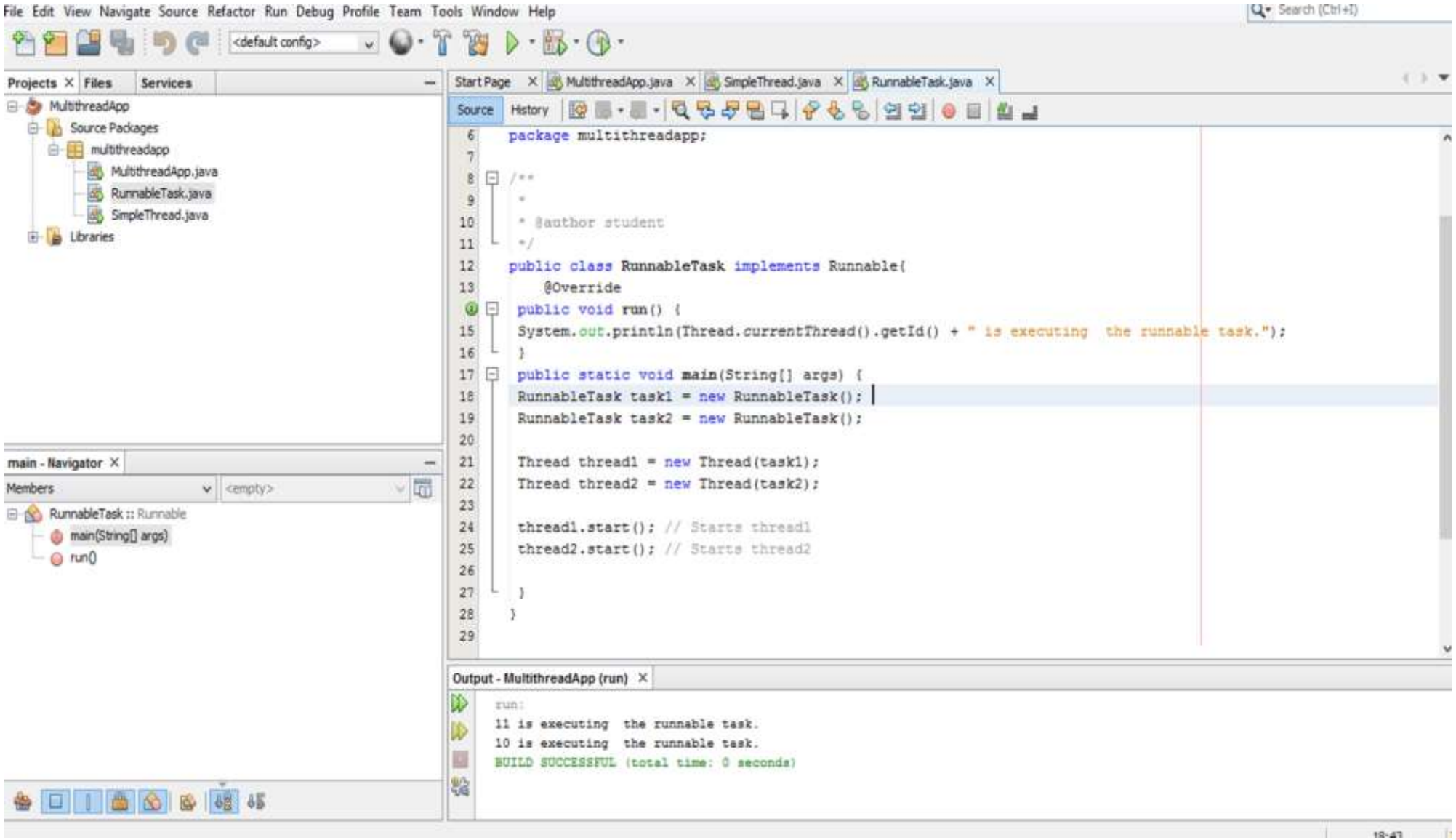
1. Create a Simple Thread Class

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

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


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
MultithreadApp
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
Start Page
MultithreadApp.java
SimpleThread.java
RunnableTask.java
Source
History
* and open the template in the editor.
*/
package multithreadapp;
/**
*
* @author student
*/
public class SimpleThread extends Thread {
@Override
public void run() {
System.out.println(Thread.currentThread().getId() + " is executing the thread.");
}
public static void main(String[] args) {
SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start(); // Starts thread1
thread2.start(); // Starts thread2
}
}
SimpleThread - Navigator
Members
<empty>
SimpleThread :: Thread
main(String[] args)
run()
Output - MultithreadApp (run)
run:
11 is executing the thread.
10 is executing the thread.
BUILD SUCCESSFUL (total time: 0 seconds)
8:4
INS

2. Create a Runnable Class

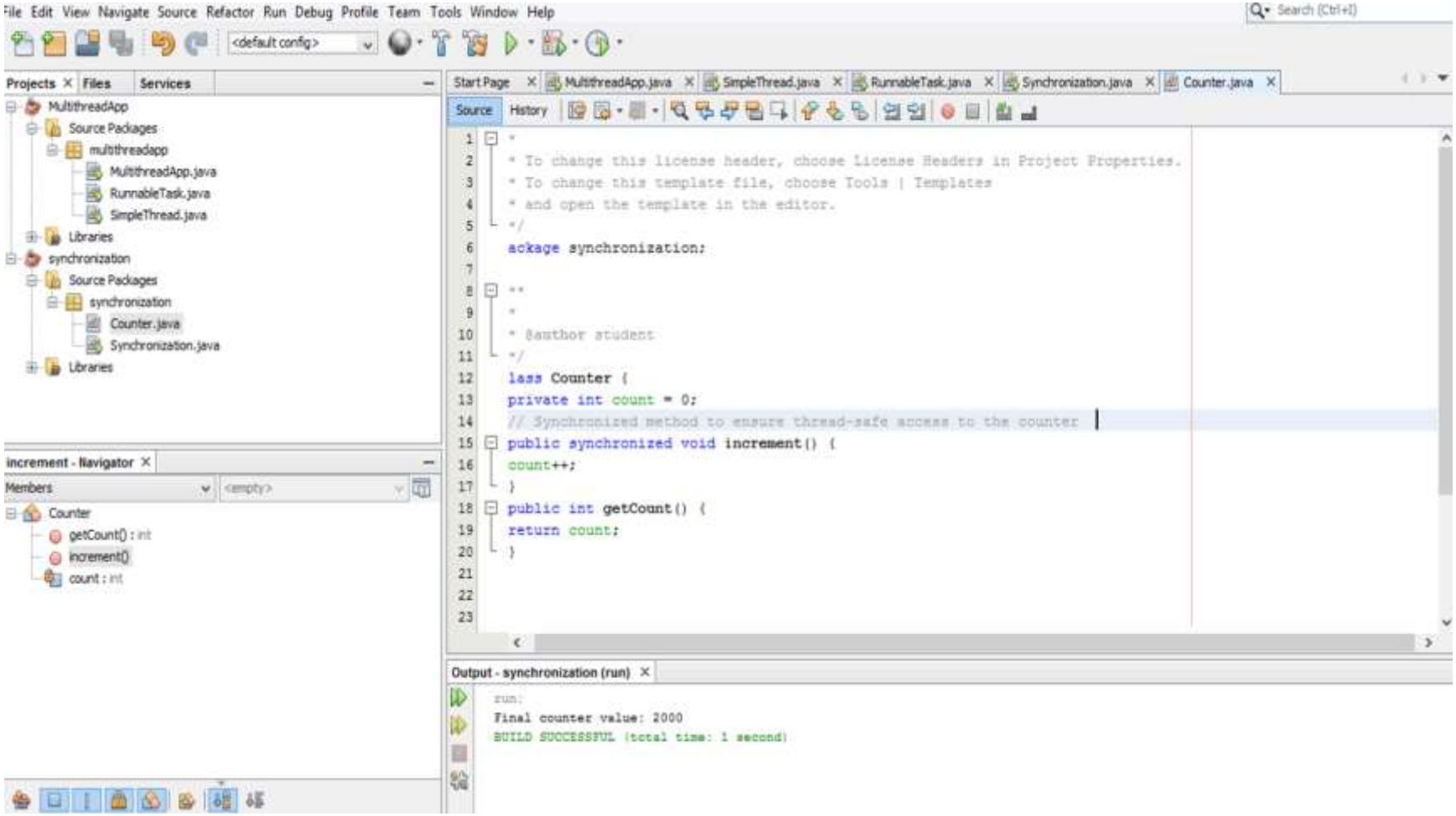
```
public class RunnableTask implements Runnable {
@Override
public void run() {
System.out.println(Thread.currentThread().getId() + " is executing the runnable task.");
}     public static void main(String[] args) {        RunnableTask task1 = new RunnableTask();
RunnableTask task2 = new RunnableTask();
Thread thread1 = new Thread(task1);
Thread thread2 = new Thread(task2);

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

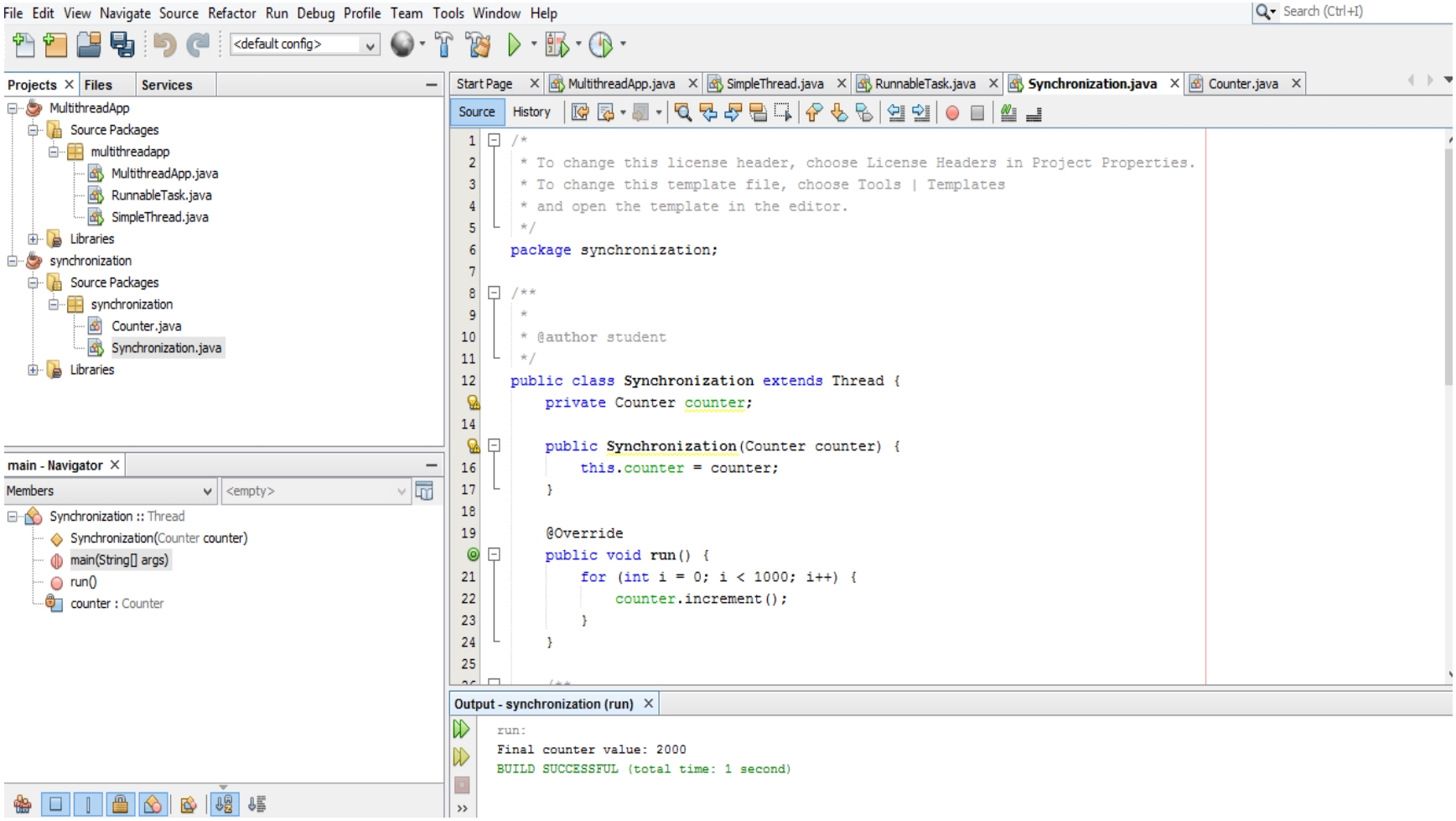

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
MultithreadApp
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
main - Navigator
Members
<empty>
RunnableTask :: Runnable
main(String[] args)
run()
Start Page
MultithreadApp.java
SimpleThread.java
RunnableTask.java
Source
History
6 package multithreadapp;
7
8 /**
9 *
10 * @author student
11 */
12 public class RunnableTask implements Runnable{
13 @Override
15 public void run() {
15 System.out.println(Thread.currentThread().getId() + " is executing the runnable task.");
16 }
17 public static void main(String[] args) {
18 RunnableTask task1 = new RunnableTask();
19 RunnableTask task2 = new RunnableTask();
20
21 Thread thread1 = new Thread(task1);
22 Thread thread2 = new Thread(task2);
23
24 thread1.start(); // Starts thread1
25 thread2.start(); // Starts thread2
26
27 }
28 }
29
Output - MultithreadApp (run)
run:
11 is executing the runnable task.
10 is executing the runnable task.
BUILD SUCCESSFUL (total time: 0 seconds)

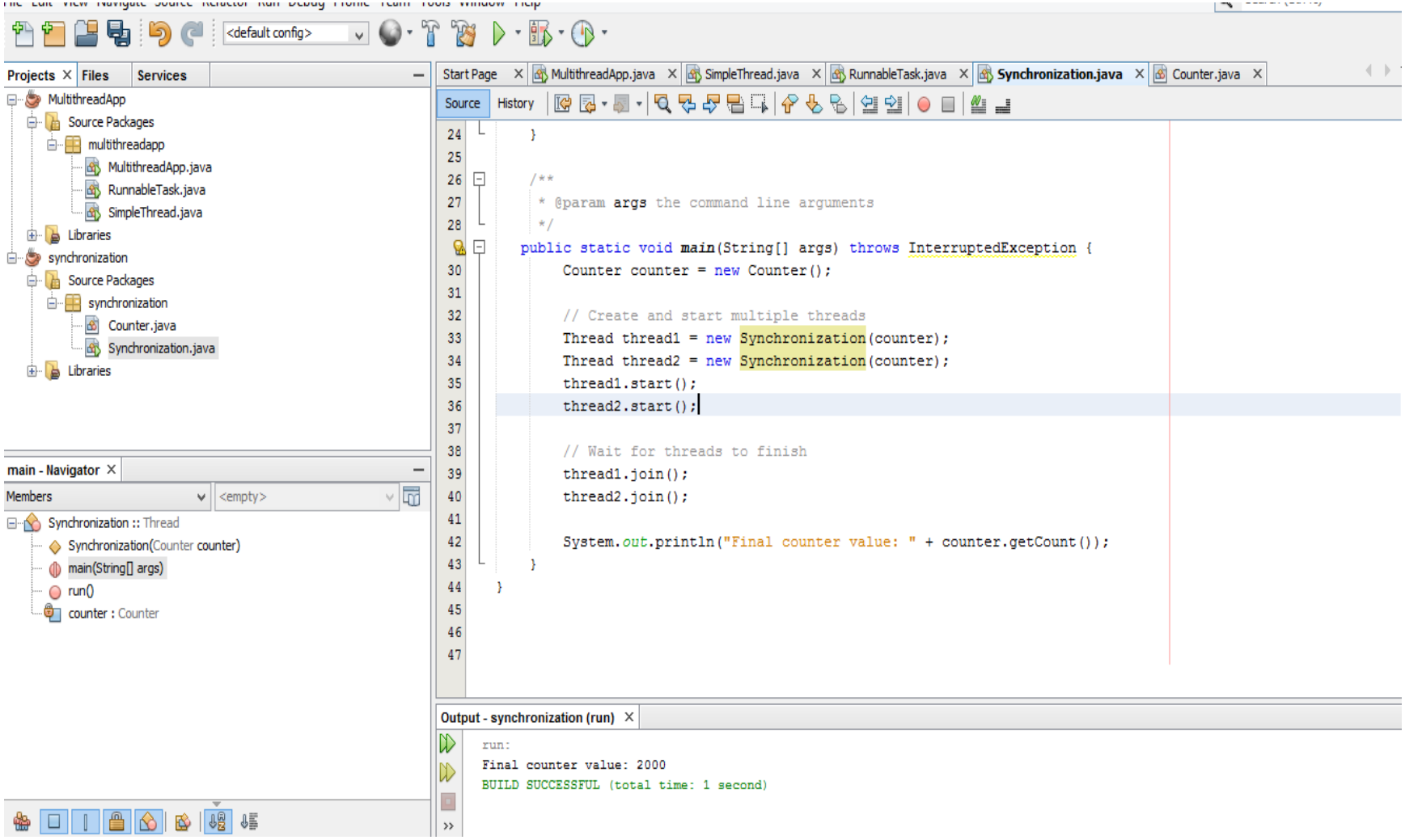
3. Synchronizing Shared Resources

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

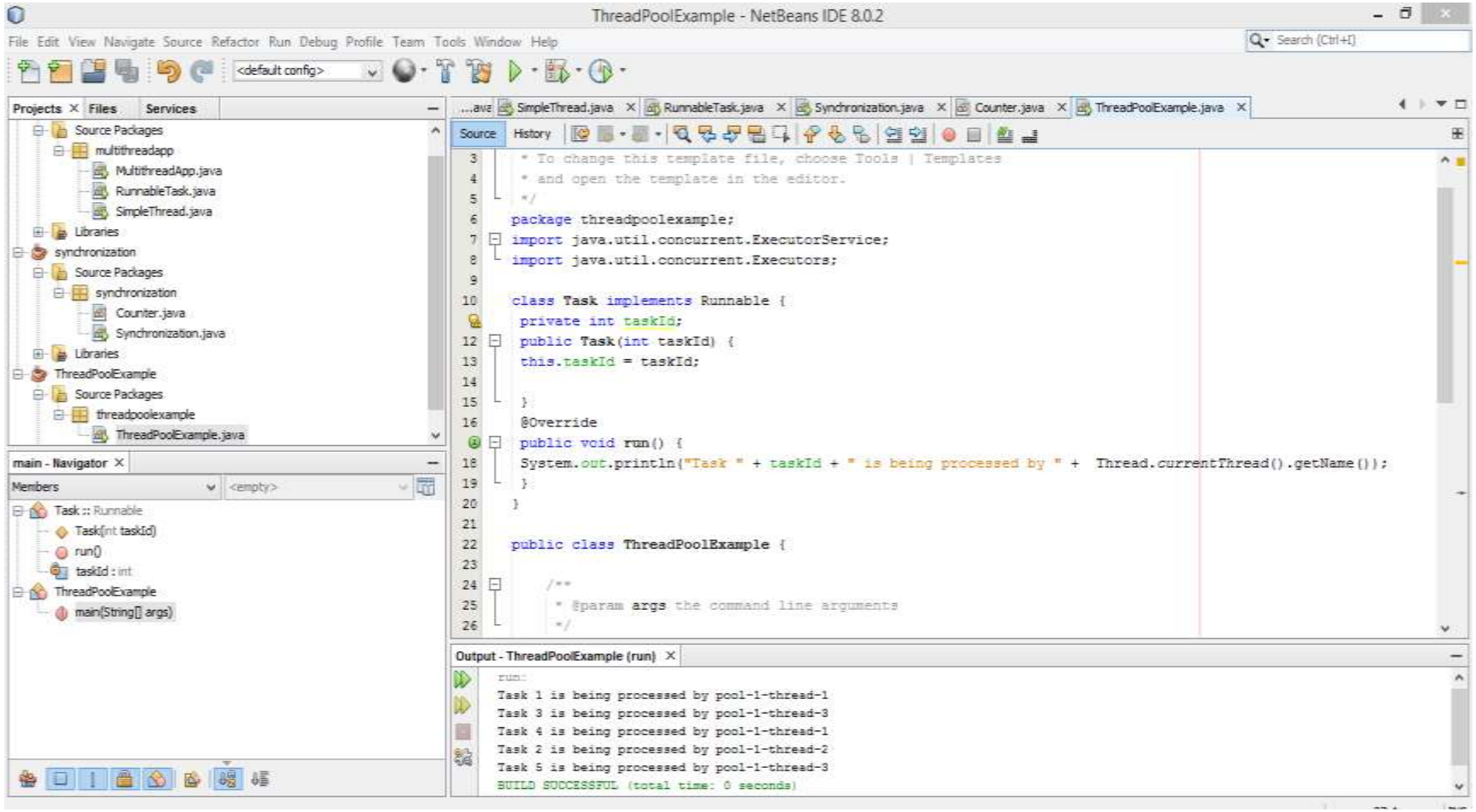


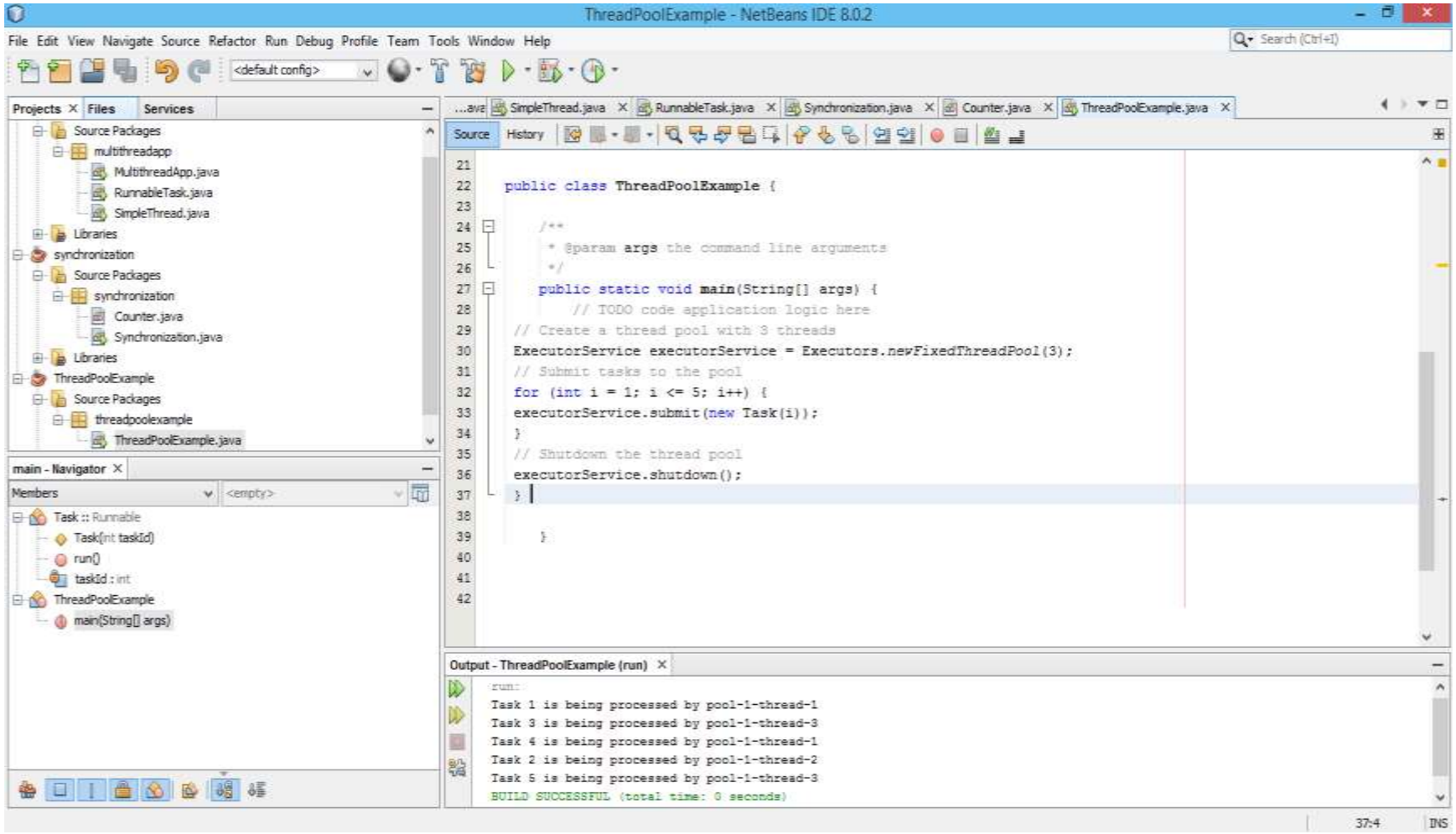

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
MultithreadApp
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
synchronization
Source Packages
synchronization
Counter.java
Synchronization.java
Libraries
main - Navigator
Members
<empty>
Synchronization :: Thread
Synchronization(Counter counter)
main(String[] args)
run()
counter : Counter
Start Page
MultithreadApp.java
SimpleThread.java
RunnableTask.java
Synchronization.java
Counter.java
Source
History
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package synchronization;
7
8 /**
9 *
10 * @author student
11 */
12 public class Synchronization extends Thread {
private Counter counter;
14
public Synchronization(Counter counter) {
16 this.counter = counter;
17 }
18
19 @Override
public void run() {
21 for (int i = 0; i < 1000; i++) {
22 counter.increment();
23 }
24 }
25
Output - synchronization (run)
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 1 second)


Projects
Files
Services
MultithreadApp
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
synchronization
Source Packages
synchronization
Counter.java
Synchronization.java
Libraries
main - Navigator
Members
<empty>
Synchronization :: Thread
Synchronization(Counter counter)
main(String[] args)
run()
counter : Counter
Start Page
MultithreadApp.java
SimpleThread.java
RunnableTask.java
Synchronization.java
Counter.java
Source
History
24 }
25
26 /**
27 * @param args the command line arguments
28 */
public static void main(String[] args) throws InterruptedException {
30 Counter counter = new Counter();
31
32 // Create and start multiple threads
33 Thread thread1 = new Synchronization(counter);
34 Thread thread2 = new Synchronization(counter);
35 thread1.start();
36 thread2.start();
37
38 // Wait for threads to finish
39 thread1.join();
40 thread2.join();
41
42 System.out.println("Final counter value: " + counter.getCount());
43 }
44 }
45
46
47
Output - synchronization (run)
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 1 second)

4. Using ExecutorService for Thread Pooling

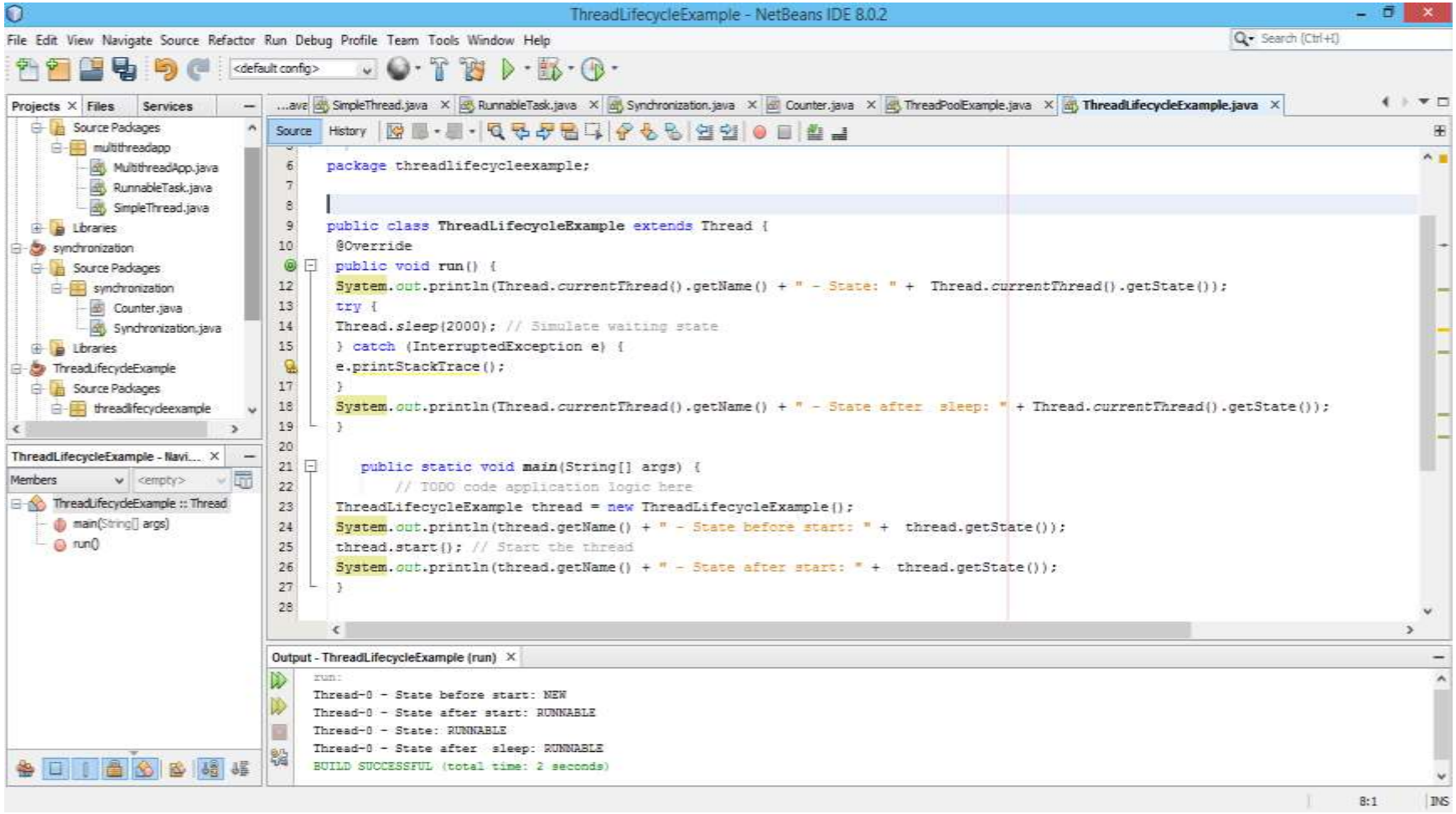
```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;


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();
} }
```


ThreadPoolExample - 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
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
synchronization
Source Packages
synchronization
Counter.java
Synchronization.java
Libraries
ThreadPoolExample
Source Packages
threadpoolexample
ThreadPoolExample.java
main - Navigator
Members
<empty>
Task :: Runnable
Task(int taskId)
run()
taskId : int
ThreadPoolExample
main(String[] args)
SimpleThread.java
RunnableTask.java
Synchronization.java
Counter.java
ThreadPoolExample.java
Source
History
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package threadpoolexample;
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 {
/**
* @param args the command line arguments
*/
Output - ThreadPoolExample (run)
run:
Task 1 is being processed by pool-1-thread-1
Task 3 is being processed by pool-1-thread-3
Task 4 is being processed by pool-1-thread-1
Task 2 is being processed by pool-1-thread-2
Task 5 is being processed by pool-1-thread-3
BUILD SUCCESSFUL (total time: 0 seconds)


ThreadPoolExample - 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
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
synchronization
Source Packages
synchronization
Counter.java
Synchronization.java
Libraries
ThreadPoolExample
Source Packages
threadpoolexample
ThreadPoolExample.java
main - Navigator
Members
<empty>
Task :: Runnable
Task(int taskId)
run()
taskId : int
ThreadPoolExample
main(String[] args)
SimpleThread.java
RunnableTask.java
Synchronization.java
Counter.java
ThreadPoolExample.java
Source
History
21
22 public class ThreadPoolExample {
23
24 /**
25 * @param args the command line arguments
26 */
27 public static void main(String[] args) {
28 // TODO code application logic here
29 // Create a thread pool with 3 threads
30 ExecutorService executorService = Executors.newFixedThreadPool(3);
31 // Submit tasks to the pool
32 for (int i = 1; i <= 5; i++) {
33 executorService.submit(new Task(i));
34 }
35 // Shutdown the thread pool
36 executorService.shutdown();
37 }
38
39 }
40
41
42
Output - ThreadPoolExample (run)
run:
Task 1 is being processed by pool-1-thread-1
Task 3 is being processed by pool-1-thread-3
Task 4 is being processed by pool-1-thread-1
Task 2 is being processed by pool-1-thread-2
Task 5 is being processed by pool-1-thread-3
BUILD SUCCESSFUL (total time: 0 seconds)
37:4
INS

5. Thread Lifecycle Example

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


ThreadLifecycleExample - 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
Source Packages
multithreadapp
MultithreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
synchronization
Source Packages
synchronization
Counter.java
Synchronization.java
Libraries
ThreadLifecycleExample
Source Packages
threadlifecycleexample
ThreadLifecycleExample - Navi...
Members
<empty>
ThreadLifecycleExample :: Thread
main(String[] args)
run()
SimpleThread.java
RunnableTask.java
Synchronization.java
Counter.java
ThreadPoolExample.java
ThreadLifecycleExample.java
Source
History
6 package threadlifecycleexample;
7
8
9 public class ThreadLifecycleExample extends Thread {
10 @Override
11 public void run() {
12 System.out.println(Thread.currentThread().getName() + " - State: " + Thread.currentThread().getState());
13 try {
14 Thread.sleep(2000); // Simulate waiting state
15 } catch (InterruptedException e) {
16 e.printStackTrace();
17 }
18 System.out.println(Thread.currentThread().getName() + " - State after sleep: " + Thread.currentThread().getState());
19 }
20
21 public static void main(String[] args) {
22 // TODO code application logic here
23 ThreadLifecycleExample thread = new ThreadLifecycleExample();
24 System.out.println(thread.getName() + " - State before start: " + thread.getState());
25 thread.start(); // Start the thread
26 System.out.println(thread.getName() + " - State after start: " + thread.getState());
27 }
28
Output - ThreadLifecycleExample (run)
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)
8:1
INS