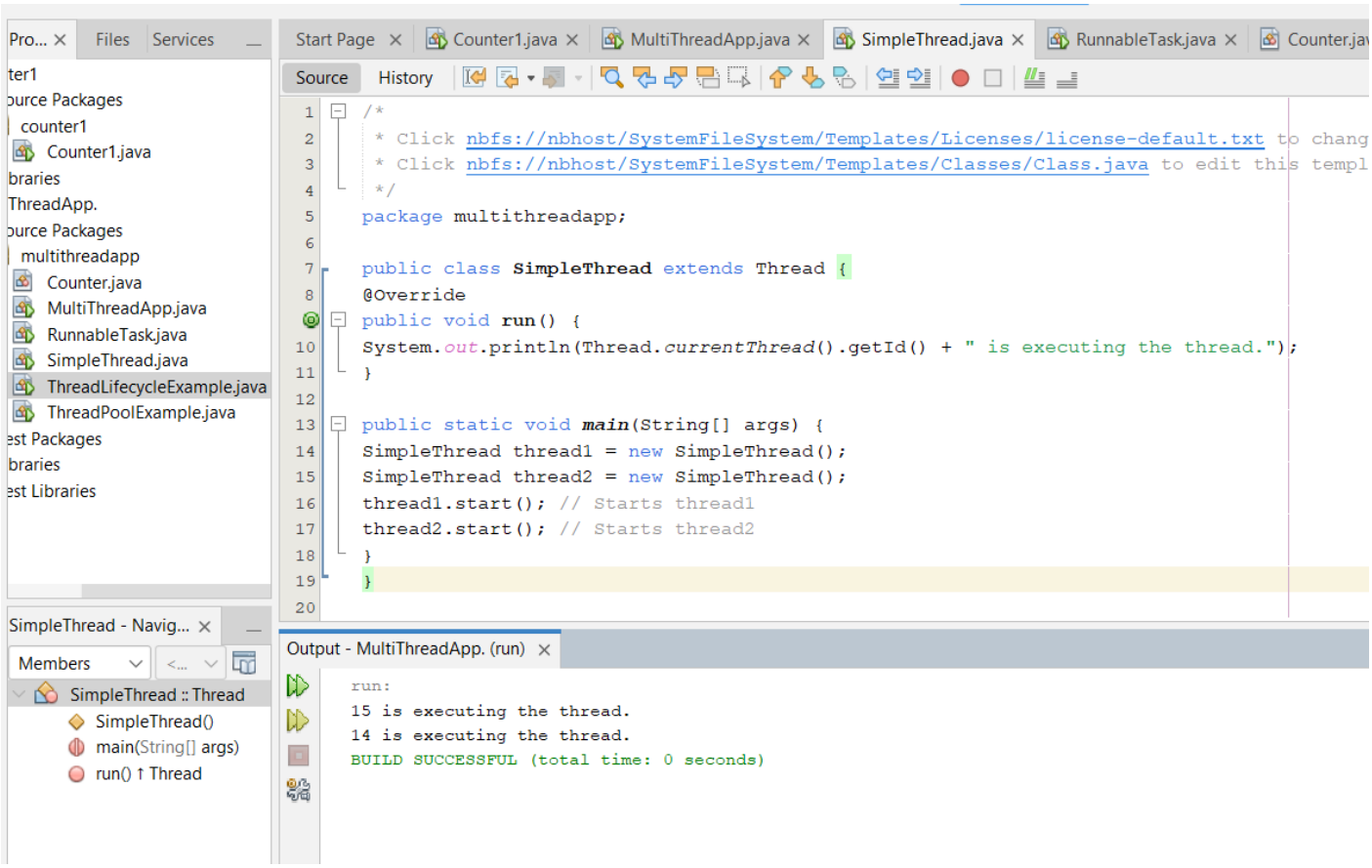
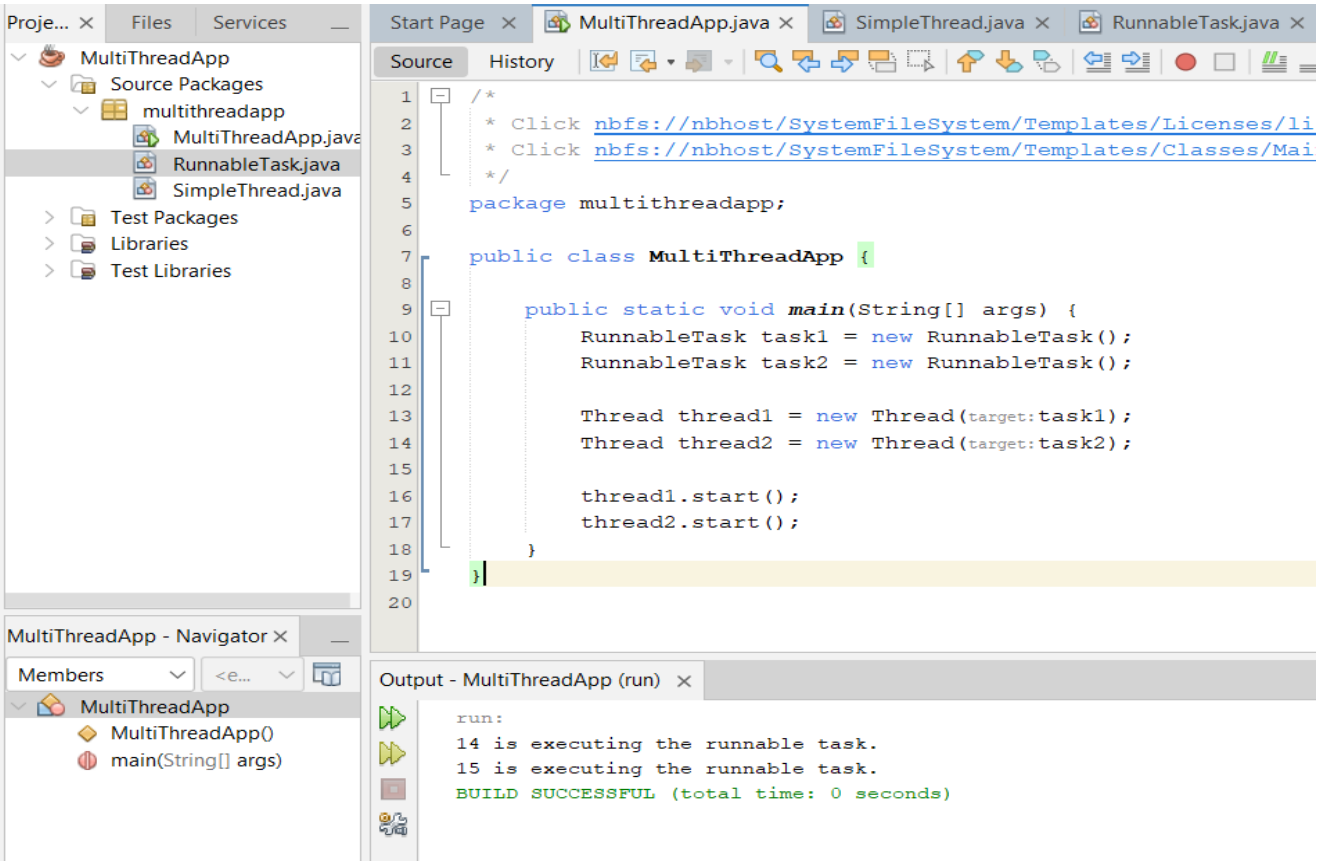
1. **Create a Simple Thread Class**

```
public class SimpleThread extends Thread {
    @Override
    public void run(){
        System.out.println(Thread.currentThread().getId() + " is executing the thread.");
    }
}

public class MultiThreadApp {
    public static void main(String[] args) {
        SimpleThread thread1 = new SimpleThread();
        SimpleThread thread2 = new SimpleThread();

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



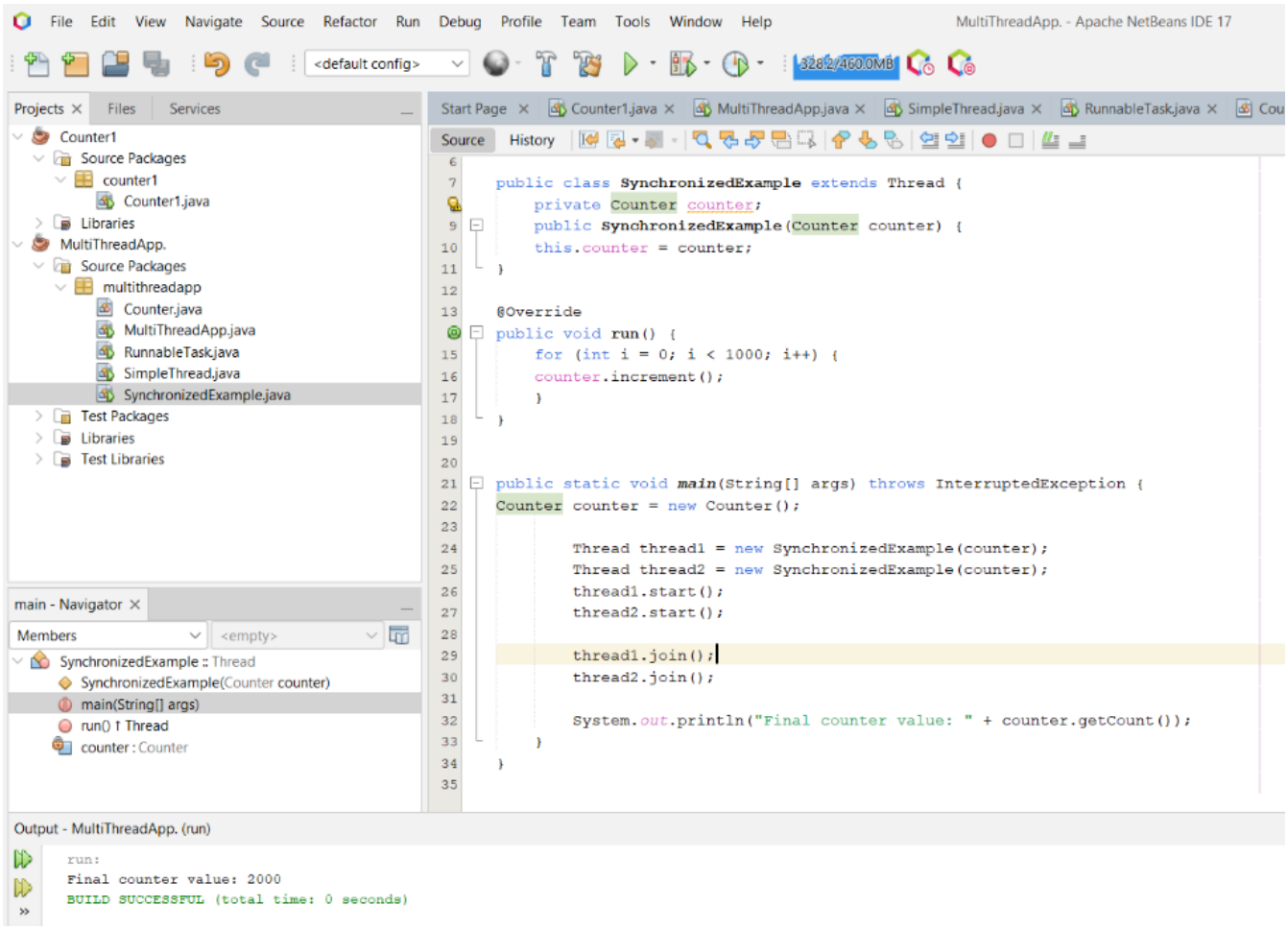
2) **Create a Runnable Class**

```
public class RunnableTask implements Runnable {
    @Override
    public void run(){
        System.out.println(Thread.currentThread().getId() + " is executing the runnable task.");
    }
}


public class MultiThreadApp {
    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();
    }
}
```



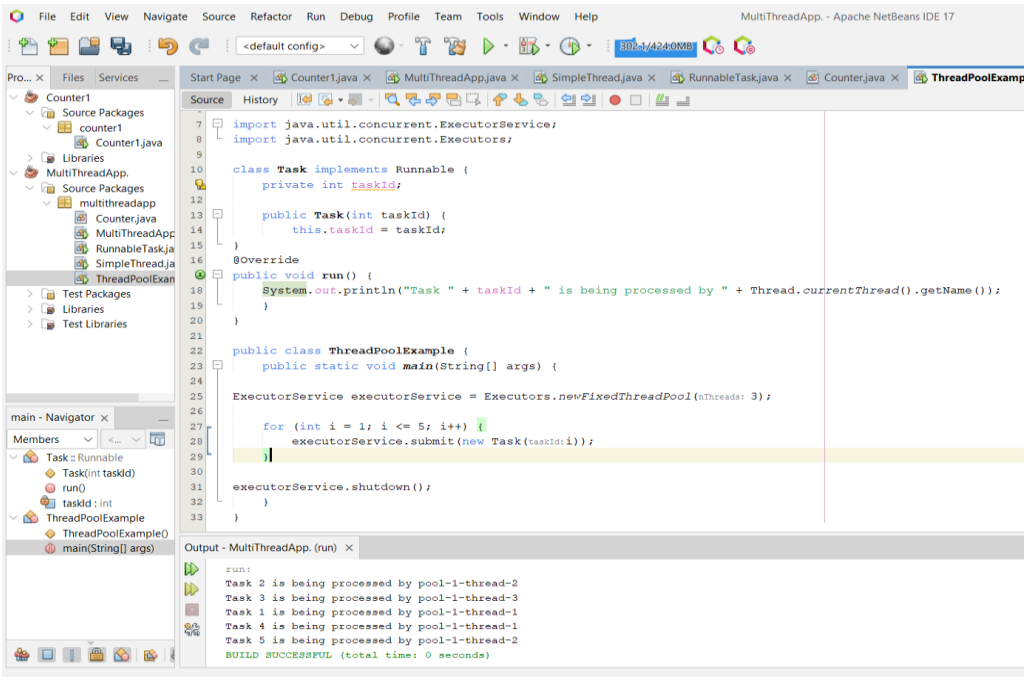
3.) Synchronizing Threads

```
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();

      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());
   }
}
```


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
MultiThreadApp. - Apache NetBeans IDE 17
<default config>
3282/460.0MB
Projects
Files
Services
Counter1
Source Packages
counter1
Counter1.java
Libraries
MultiThreadApp.
Source Packages
multithreadapp
Counter.java
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
SynchronizedExample.java
Test Packages
Libraries
Test Libraries
Start Page
Counter1.java
MultiThreadApp.java
SimpleThread.java
RunnableTask.java
Source
History
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();
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());
}
}
main - Navigator
Members
<empty>
SynchronizedExample :: Thread
SynchronizedExample(Counter counter)
main(String[] args)
run() ↑ Thread
counter : Counter
Output - MultiThreadApp. (run)
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 0 seconds)

4.) 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) {

ExecutorService executorService = Executors.newFixedThreadPool(3);

    for (int i = 1; i <= 5; i++) {
        executorService.submit(new Task(i));
    }

executorService.shutdown();
    }
}
```



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);
   } 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();
   System.out.println(thread.getName() + " - State after start: " + thread.getState());
   }
}
```