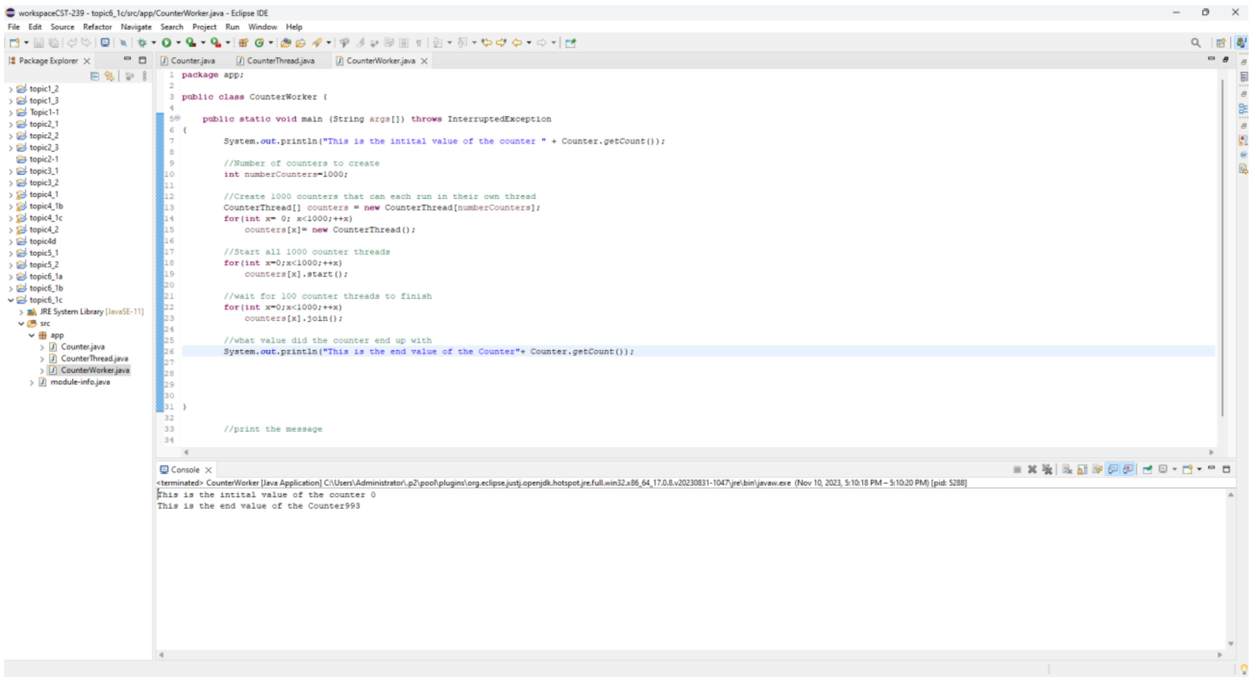
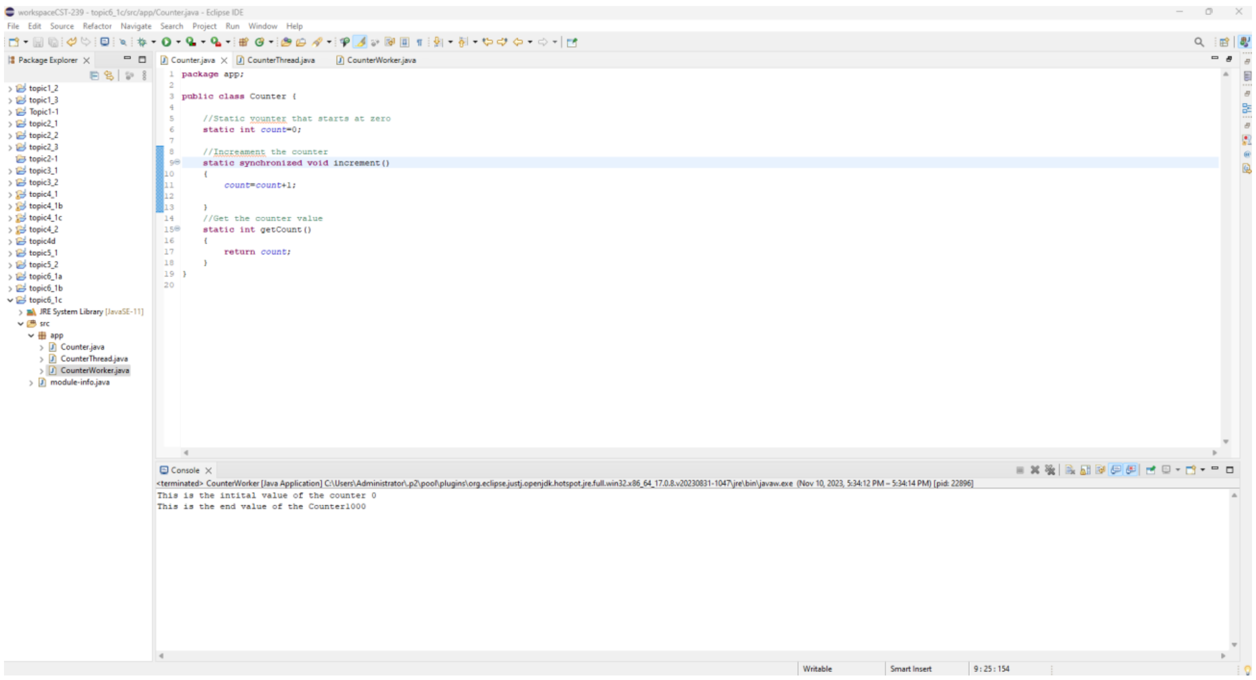
Activity 6-1c

Screenshot of output:



```java
package app;

public class CounterWorker {

	public static void main (String args[]) throws InterruptedException
{
		System.out.println("This is the intital value of the counter " + Counter.getCount());

		//Number of counters to create
		int numberCounters=1000;

		//Create 1000 counters that can each run in their own thread
		CounterThread[] counters = new CounterThread[numberCounters];
		for(int x= 0; x<1000;++x)
			counters[x]= new CounterThread();

		//Start all 1000 counter threads
		for(int x=0;x<1000;++x)
			counters[x].start();

		//wait for 100 counter threads to finish
		for(int x=0;x<1000;++x)
			counters[x].join();

		//what value did the counter end up with
		System.out.println("This is the end value of the Counter"+ Counter.getCount());




}

		//print the message

```

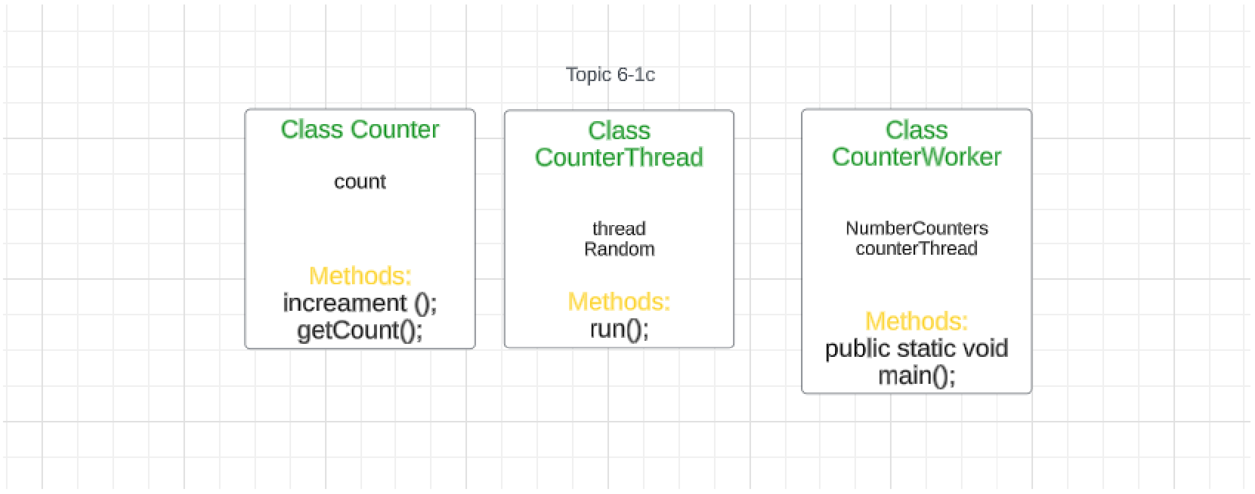
```
<terminated> CounterWorker [Java Application] C:\Users\Administrator\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (Nov 10, 2023, 5:10:18 PM – 5:10:20 PM) [pid: 5288]
This is the intital value of the counter 0
This is the end value of the Counter993
```



```java
package app;

public class Counter {

	//Static vounter that starts at zero
	static int count=0;

	//Increament the counter
	static synchronized void increment()
	{
		count=count+1;

	}
	//Get the counter value
	static int getCount()
	{
		return count;
	}
}

```

```
<terminated> CounterWorker [Java Application] C:\Users\Administrator\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (Nov 10, 2023, 5:34:12 PM – 5:34:14 PM) [pid: 22896]
This is the intital value of the counter 0
This is the end value of the Counter1000
```

UML:



Write up:

6-1c part 1

We had a method that had a try and catch error exception, inside that run method we had a generator of a new random and a sleeper method. I think the output not being able to generate a 1000 and increment by one was because of the sleeper method and the run method not being synchronized. The sleeper method was eliminating the thread continuation on an increment. I believe that may be the reason on why we weren't getting the incrementation because when I remove the sleeper method and the synchronized, I end up getting the 1000 and incrementation.

6-1c part 2

We had the same code as the previous code, the only thing we have changed is the synchronized method was added. When we use the synchronized, it gives us the ability to control whatever being accessed on multiple threads who share the same resources. When we added the synchronized to the method, it prevented thread interface which outputted the 1000.