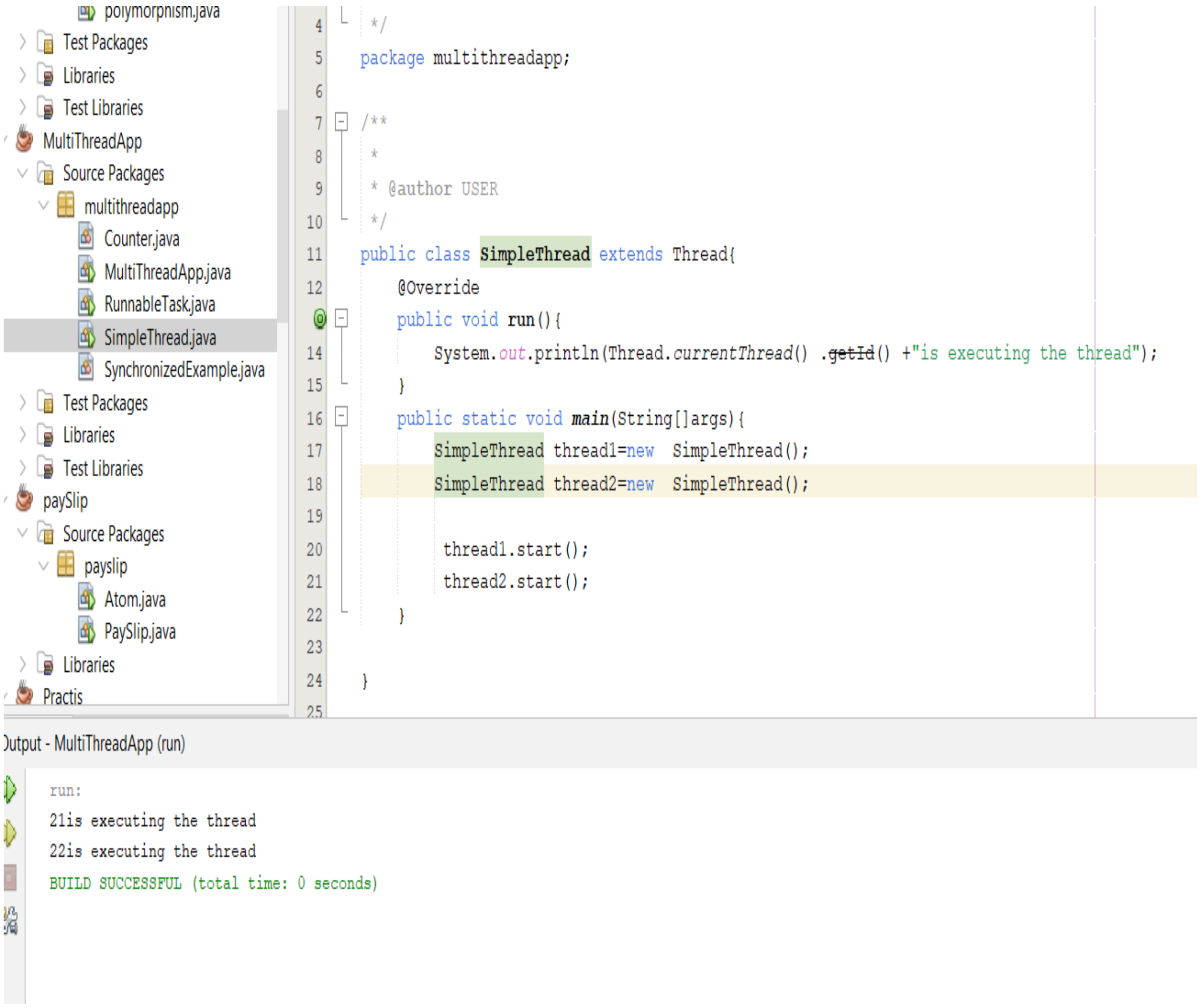
SimpleThread

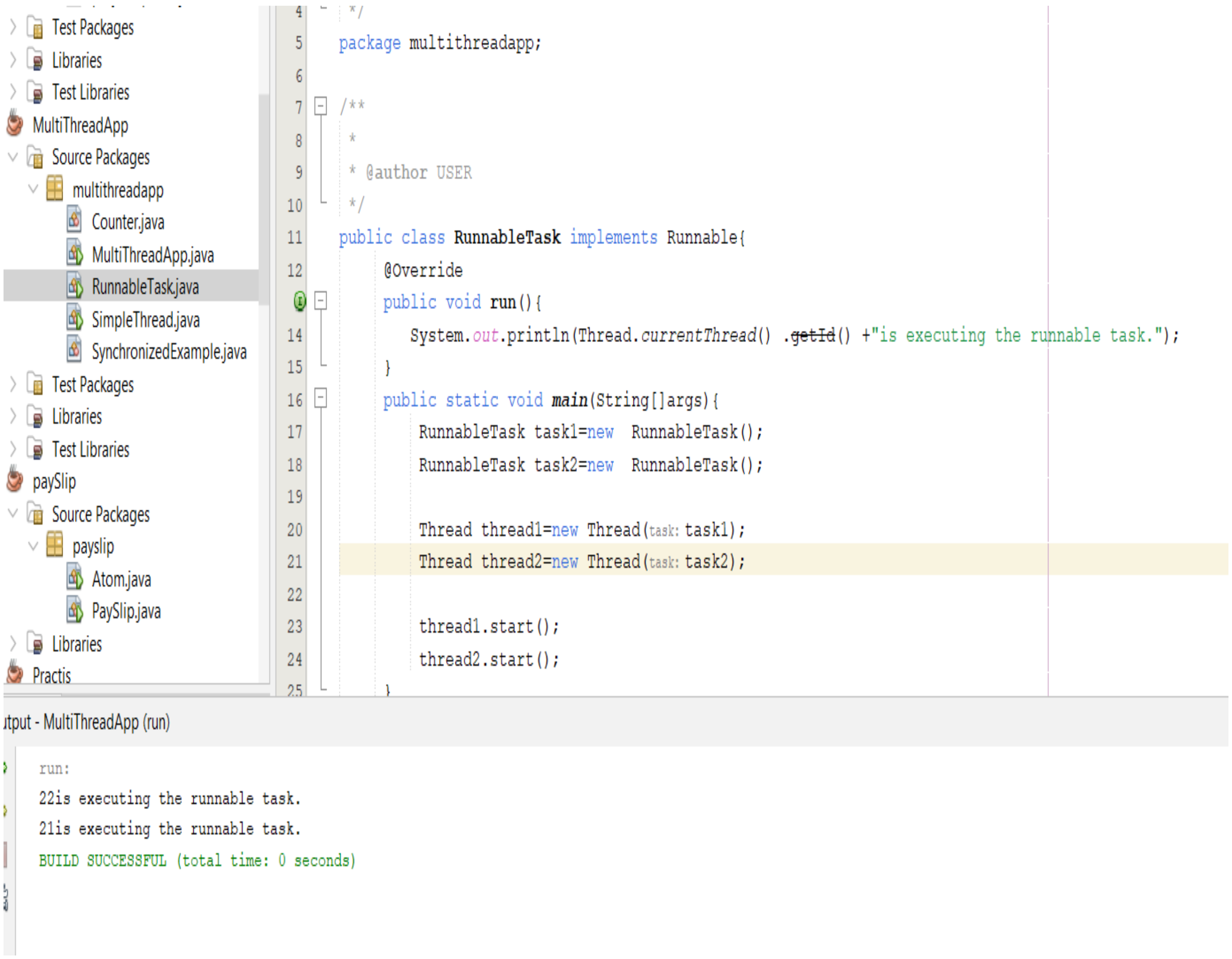


```java
 */
package multithreadapp;

/**
 *
 * @author USER
 */
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();
         thread2.start();
    }


}
```

Output - MultiThreadApp (run)

```
run:
21is executing the thread
22is executing the thread
BUILD SUCCESSFUL (total time: 0 seconds)
```

RunnableTask



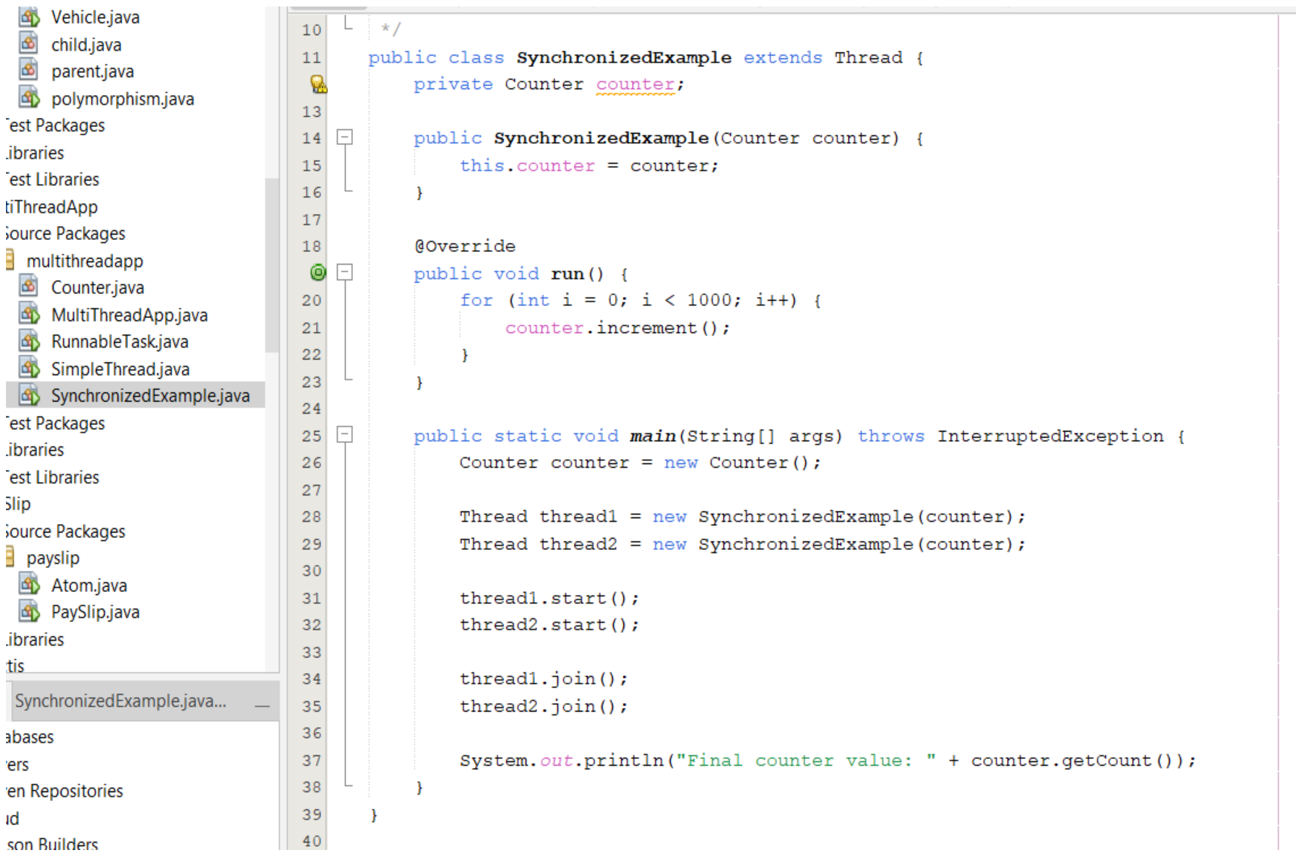
Test Packages
Libraries
Test Libraries
MultiThreadApp
Source Packages
multithreadapp
Counter.java
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
SynchronizedExample.java
Test Packages
Libraries
Test Libraries
paySlip
Source Packages
payslip
Atom.java
PaySlip.java
Libraries
Practis

```java
 */
package multithreadapp;

/**
 *
 * @author USER
 */
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(task: task1);
        Thread thread2=new Thread(task: task2);

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

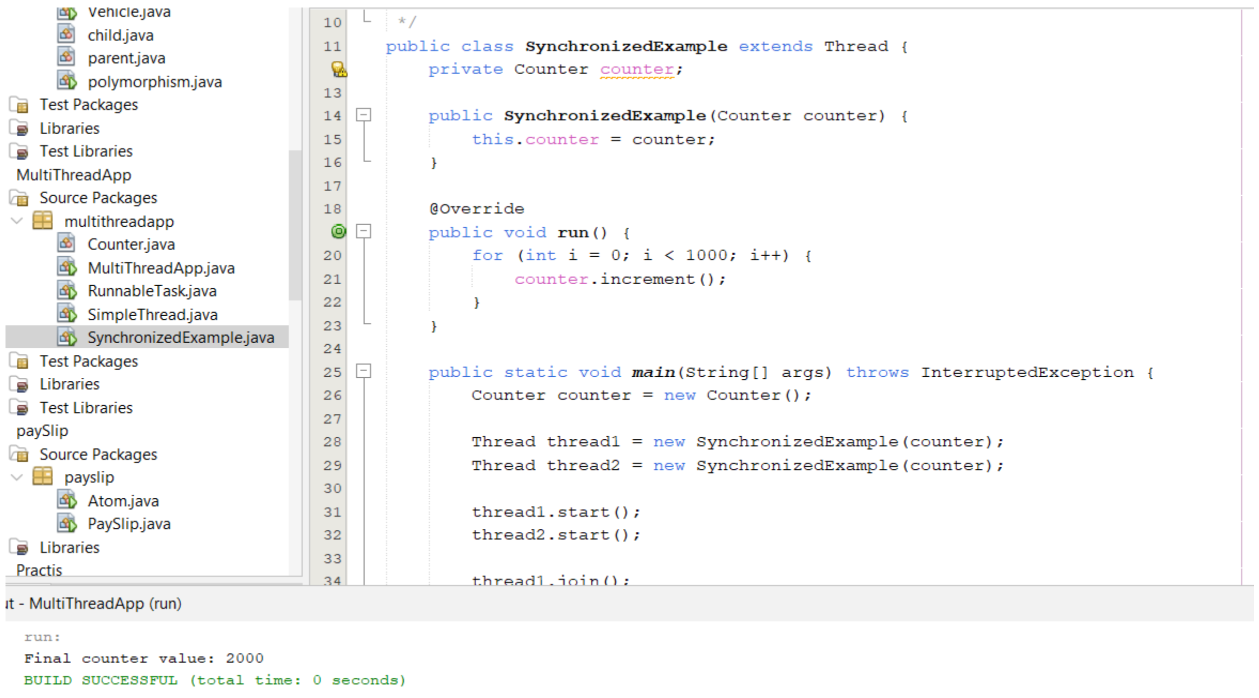
tput - MultiThreadApp (run)

```
run:
22is executing the runnable task.
21is executing the runnable task.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Synchronizing Thread



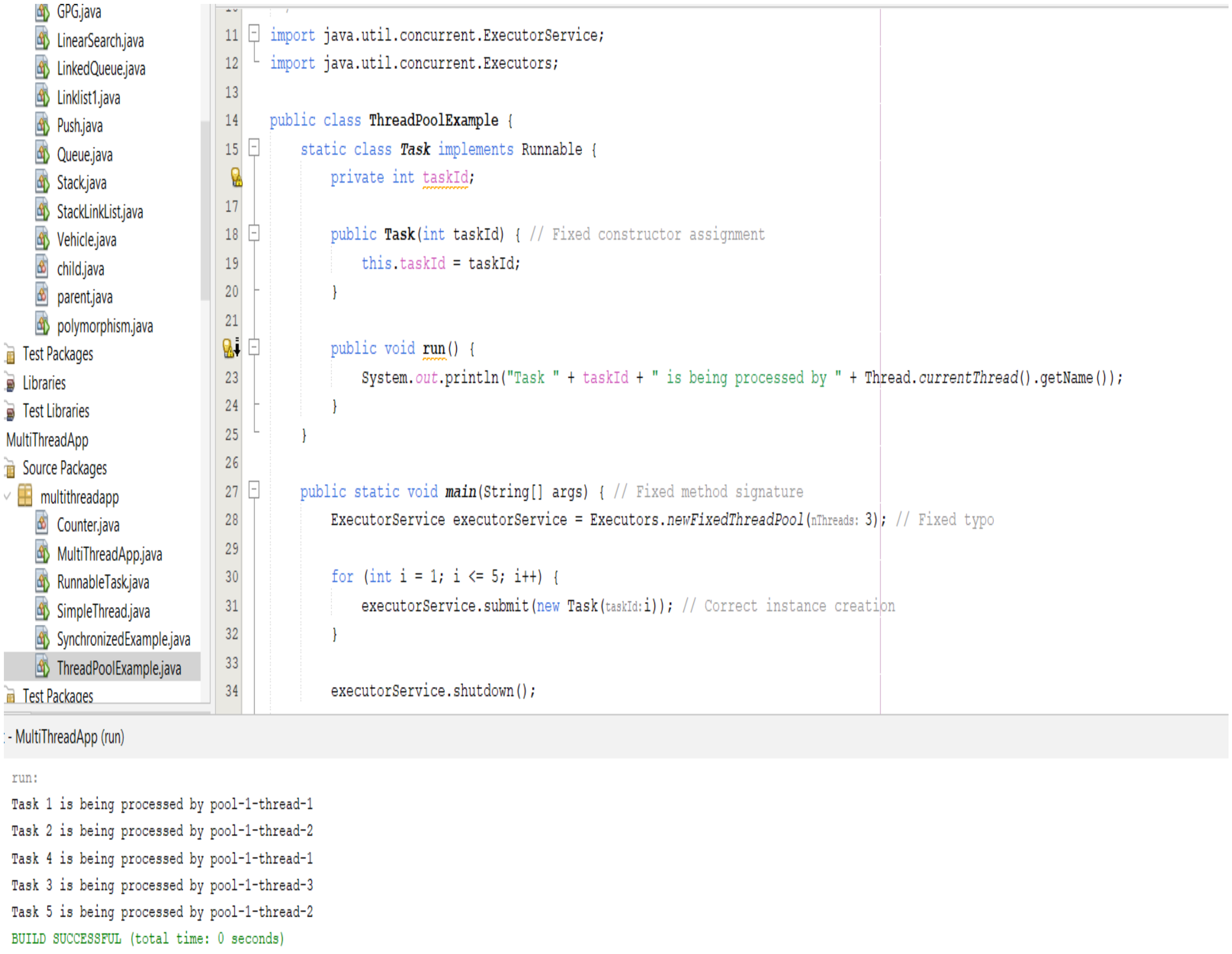
```java
 */
public class SynchronizedExample extends Thread {
    private Counter counter;

    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());
    }
}
```



```java
 */
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();
```

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

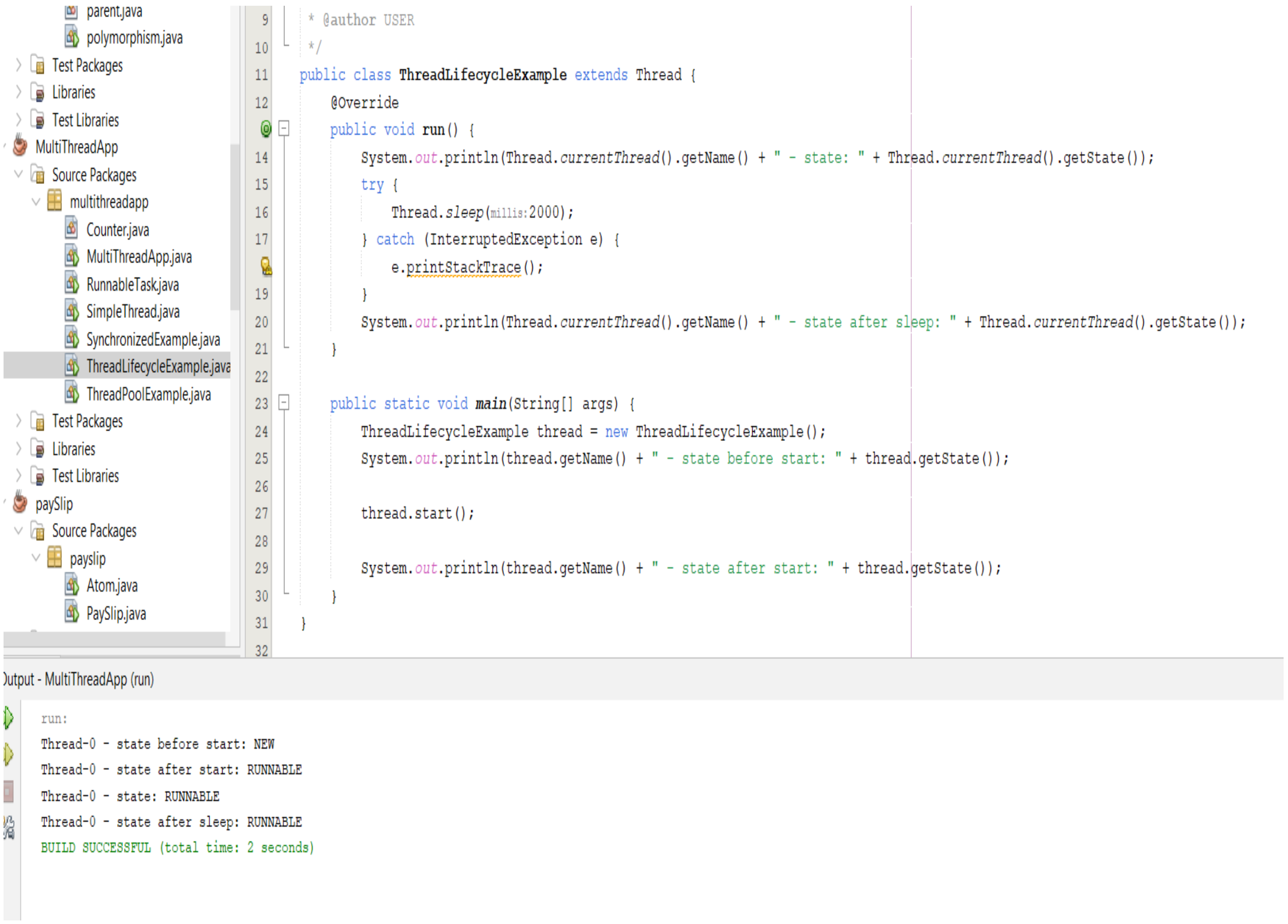
Thread Pooling



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

public class ThreadPoolExample {
    static class Task implements Runnable {
        private int taskId;

        public Task(int taskId) { // Fixed constructor assignment
            this.taskId = taskId;
        }

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

    public static void main(String[] args) { // Fixed method signature
        ExecutorService executorService = Executors.newFixedThreadPool(nThreads: 3); // Fixed typo

        for (int i = 1; i <= 5; i++) {
            executorService.submit(new Task(taskId:i)); // Correct instance creation
        }

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

Thread Lifecycle Example



```java
 * @author USER
 */
public class ThreadLifecycleExample extends Thread {
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " - state: " + Thread.currentThread().getState());
        try {
            Thread.sleep(millis: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());
    }
}
```

```
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)
```