

SimpleThread

The screenshot shows an IDE with a project named 'MultiThreadApp'. The 'Source Packages' view on the left shows the package structure: 'multithreadapp' containing 'Counter.java', 'MultiThreadApp.java', 'RunnableTask.java', 'SimpleThread.java' (selected), and 'SynchronizedExample.java'. The main editor displays the code for 'SimpleThread.java'.

```
4  L  */
5  package multithreadapp;
6
7  L  /**
8  L  *
9  L  * @author USER
10 L  */
11 public class SimpleThread extends Thread{
12     @Override
13     public void run() {
14         System.out.println(Thread.currentThread().getId() + "is executing the thread");
15     }
16     public static void main(String[] args) {
17         SimpleThread thread1=new SimpleThread();
18         SimpleThread thread2=new SimpleThread();
19
20         thread1.start();
21         thread2.start();
22     }
23
24 }
25
```

The output window at the bottom shows the results of running the application:

```
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

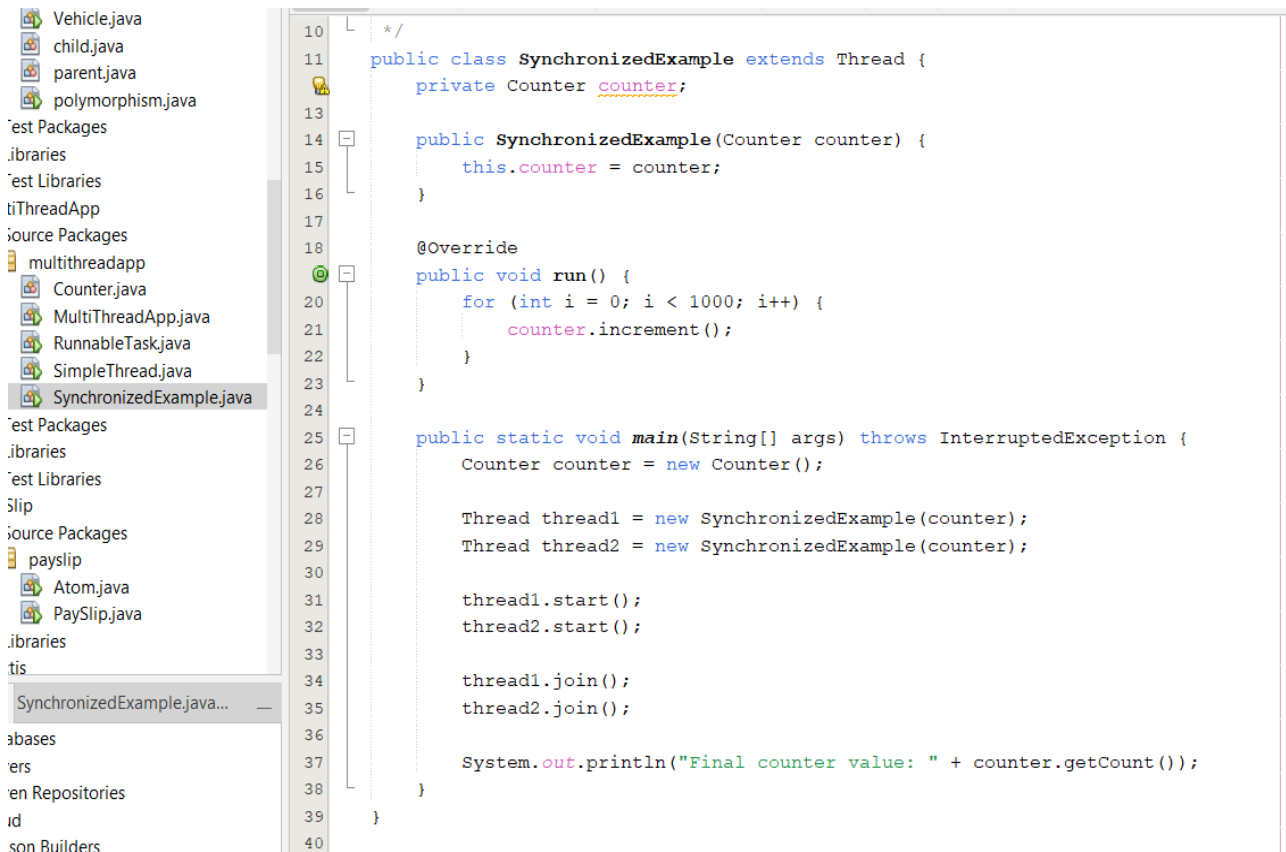
Practis

```
4  */
5  package multithreadapp;
6
7  /**
8   *
9   * @author USER
10  */
11  public class RunnableTask implements Runnable{
12      @Override
13      public void run(){
14          System.out.println(Thread.currentThread().getId() + "is executing the runnable task.");
15      }
16      public static void main(String[] args){
17          RunnableTask task1=new RunnableTask();
18          RunnableTask task2=new RunnableTask();
19
20          Thread thread1=new Thread(task: task1);
21          Thread thread2=new Thread(task: task2);
22
23          thread1.start();
24          thread2.start();
25      }
```

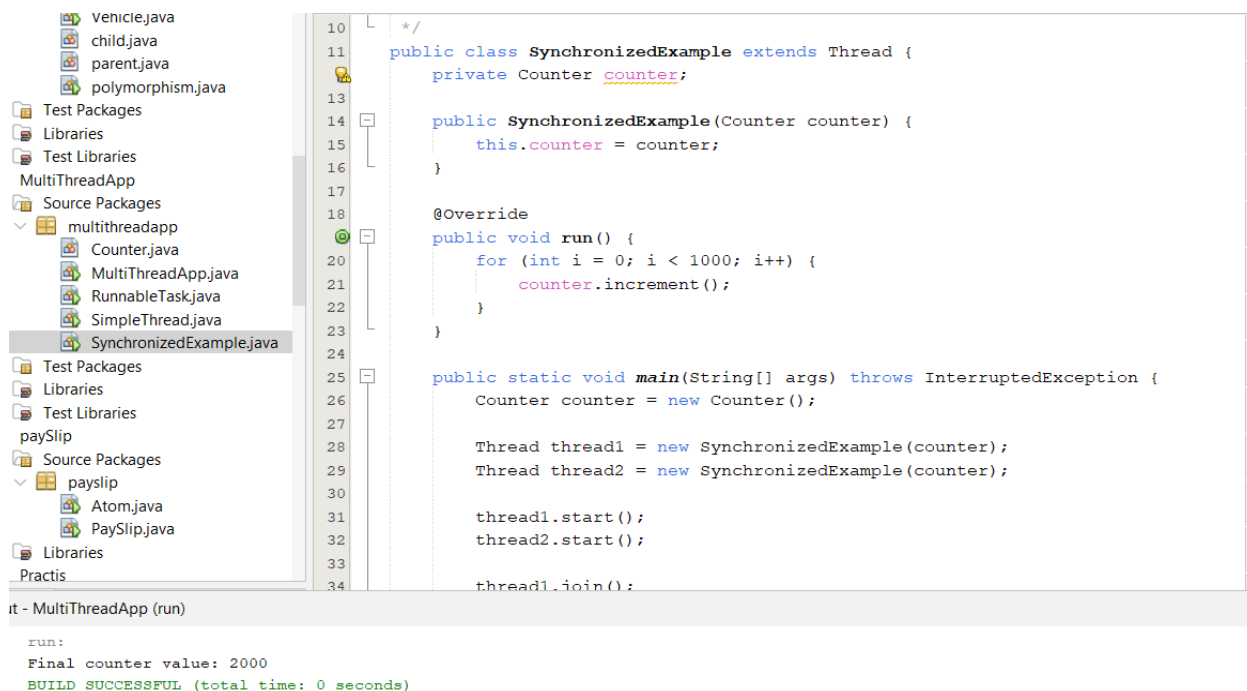
Output - MultiThreadApp (run)

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

Synchronizing Thread



```
10  L  */
11  public class SynchronizedExample extends Thread {
12      private Counter counter;
13
14      public SynchronizedExample(Counter counter) {
15          this.counter = counter;
16      }
17
18      @Override
19      public void run() {
20          for (int i = 0; i < 1000; i++) {
21              counter.increment();
22          }
23      }
24
25      public static void main(String[] args) throws InterruptedException {
26          Counter counter = new Counter();
27
28          Thread thread1 = new SynchronizedExample(counter);
29          Thread thread2 = new SynchronizedExample(counter);
30
31          thread1.start();
32          thread2.start();
33
34          thread1.join();
35          thread2.join();
36
37          System.out.println("Final counter value: " + counter.getCount());
38      }
39  }
40
```



```
10  L  */
11  public class SynchronizedExample extends Thread {
12      private Counter counter;
13
14      public SynchronizedExample(Counter counter) {
15          this.counter = counter;
16      }
17
18      @Override
19      public void run() {
20          for (int i = 0; i < 1000; i++) {
21              counter.increment();
22          }
23      }
24
25      public static void main(String[] args) throws InterruptedException {
26          Counter counter = new Counter();
27
28          Thread thread1 = new SynchronizedExample(counter);
29          Thread thread2 = new SynchronizedExample(counter);
30
31          thread1.start();
32          thread2.start();
33
34          thread1.join();
```

it - MultiThreadApp (run)

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

Thread Pooling

GPJ.java

LinearSearch.java

LinkedQueue.java

Linklist1.java

Push.java

Queue.java

Stack.java

StackLinkList.java

Vehicle.java

child.java

parent.java

polymorphism.java

Test Packages

Libraries

Test Libraries

MultiThreadApp

Source Packages

multithreadapp

Counter.java

MultiThreadApp.java

RunnableTask.java

SimpleThread.java

SynchronizedExample.java

ThreadPoolExample.java

Test Packages

```
11 import java.util.concurrent.ExecutorService;
12 import java.util.concurrent.Executors;
13
14 public class ThreadPoolExample {
15     static class Task implements Runnable {
16         private int taskId;
17
18         public Task(int taskId) { // Fixed constructor assignment
19             this.taskId = taskId;
20         }
21
22         public void run() {
23             System.out.println("Task " + taskId + " is being processed by " + Thread.currentThread().getName());
24         }
25     }
26
27     public static void main(String[] args) { // Fixed method signature
28         ExecutorService executorService = Executors.newFixedThreadPool(3); // Fixed typo
29
30         for (int i = 1; i <= 5; i++) {
31             executorService.submit(new Task(taskId:i)); // Correct instance creation
32         }
33
34         executorService.shutdown();
35     }
36 }
```

- MultiThreadApp (run)

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

parent.java

polymorphism.java

Test Packages

Libraries

Test Libraries

MultiThreadApp

Source Packages

multithreadapp

Counter.java

MultiThreadApp.java

RunnableTask.java

SimpleThread.java

SynchronizedExample.java

ThreadLifecycleExample.java

ThreadPoolExample.java

Test Packages

Libraries

Test Libraries

paySlip

Source Packages

payslip

Atom.java

PaySlip.java

```
9      * @author USER
10     */
11     public class ThreadLifecycleExample extends Thread {
12         @Override
13         public void run() {
14             System.out.println(Thread.currentThread().getName() + " - state: " + Thread.currentThread().getState());
15             try {
16                 Thread.sleep(2000);
17             } catch (InterruptedException e) {
18                 e.printStackTrace();
19             }
20             System.out.println(Thread.currentThread().getName() + " - state after sleep: " + Thread.currentThread().getState());
21         }
22     }
23     public static void main(String[] args) {
24         ThreadLifecycleExample thread = new ThreadLifecycleExample();
25         System.out.println(thread.getName() + " - state before start: " + thread.getState());
26
27         thread.start();
28
29         System.out.println(thread.getName() + " - state after start: " + thread.getState());
30     }
31 }
32
```

Output - MultiThreadApp (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)