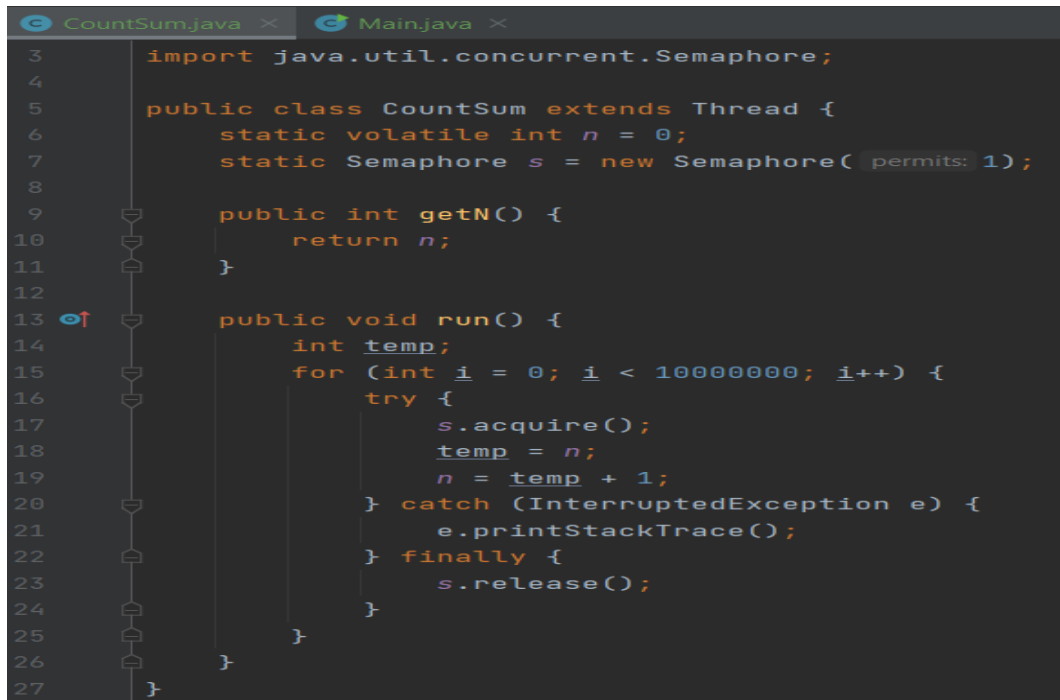
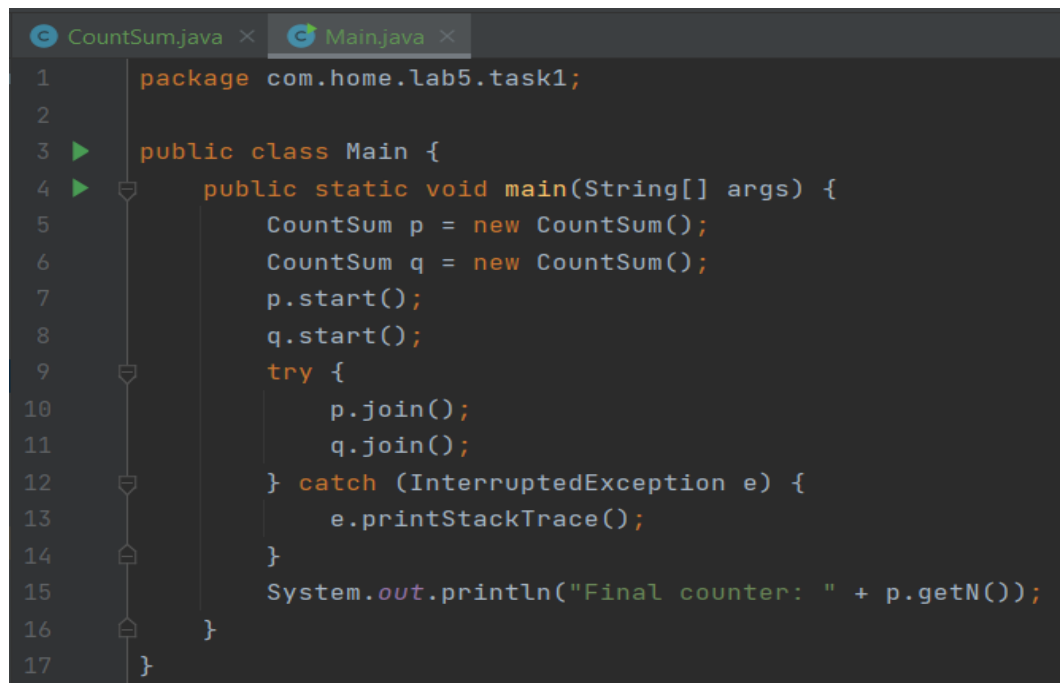


## Laborator 6

### Problema 1



```
3      import java.util.concurrent.Semaphore;
4
5      public class CountSum extends Thread {
6          static volatile int n = 0;
7          static Semaphore s = new Semaphore( permits: 1 );
8
9          public int getN() {
10             return n;
11         }
12
13         public void run() {
14             int temp;
15             for (int i = 0; i < 100000000; i++) {
16                 try {
17                     s.acquire();
18                     temp = n;
19                     n = temp + 1;
20                 } catch (InterruptedException e) {
21                     e.printStackTrace();
22                 } finally {
23                     s.release();
24                 }
25             }
26         }
27     }
```



```
1      package com.home.lab5.task1;
2
3      public class Main {
4          public static void main(String[] args) {
5              CountSum p = new CountSum();
6              CountSum q = new CountSum();
7              p.start();
8              q.start();
9              try {
10                 p.join();
11                 q.join();
12             } catch (InterruptedException e) {
13                 e.printStackTrace();
14             }
15             System.out.println("Final counter: " + p.getN());
16         }
17     }
```

### Output Problema 1:

```
Main (1) x
C:\Users\user\.jdk\corretto-16.0.2
Final counter: 20000000
Process finished with exit code 0
```

### Problema 2

```
CountDownLatchDemo.java x Worker.java x
1 package com.home.lab5.task2;
2
3 import java.util.Arrays;
4 import java.util.concurrent.CountDownLatch;
5
6 public class CountDownLatchDemo {
7     public static void main(String args[]) throws InterruptedException {
8         CountDownLatch latch = new CountDownLatch(4);
9         Worker first = new Worker( delay: 1000, latch, name: "WORKER-1");
10        Worker second = new Worker( delay: 2000, latch, name: "WORKER-2");
11        Worker third = new Worker( delay: 3000, latch, name: "WORKER-3");
12        Worker fourth = new Worker( delay: 4000, latch, name: "WORKER-4");
13        for (Worker worker : Arrays.asList(first, second, third, fourth)) {
14            worker.start();
15        }
16        // The main task waits for four threads
17        latch.await();
18        // Main thread has started
19        System.out.println(Thread.currentThread().getName() + " has finished");
20    }
21 }
```

```
CountDownLatchDemo.java x Worker.java x
1 package com.home.lab5.task2;
2
3 import java.util.concurrent.CountDownLatch;
4
5 public class Worker extends Thread {
6     private int delay;
7     private CountDownLatch latch;
8
9     public Worker(int delay, CountDownLatch latch, String name) {
10         super(name);
11         this.delay = delay;
12         this.latch = latch;
13     }
14
15     public void run() {
16         try {
17             Thread.sleep(delay);
18             latch.countDown();
19             System.out.println("Current thread with name: " + Thread.currentThread().getName() + " was finished");
20         } catch (InterruptedException e) {
21             e.printStackTrace();
22         }
23     }
24 }
```

## Output Problema 2:

```
CountDownLatchDemo x
C:\Users\user\.jdk\corretto-16.0.2\bin\java.exe
Current thread with name: WORKER-1 was finished
Current thread with name: WORKER-2 was finished
Current thread with name: WORKER-3 was finished
Current thread with name: WORKER-4 was finished
main has finished

Process finished with exit code 0
```

## Problema 3

```
HRManager.java x TechLead.java x
2
3 import java.util.Arrays;
4 import java.util.concurrent.CyclicBarrier;
5
6 public class TechLead extends Thread{
7     CyclicBarrier cyclicBarrier;
8     public TechLead(CyclicBarrier cyclicBarrier, String name) {
9         super(name);
10        this.cyclicBarrier= cyclicBarrier;
11    }
12    public void run() {
13        try {
14            Thread.sleep( millis 3000);
15            for (String s : Arrays.asList(" recruited developer", " waiting for others to complete ...")) {
16                System.out.println(Thread.currentThread().getName()+ s);
17            }
18            cyclicBarrier.await();
19            System.out.println("All finished recruiting, " + Thread.currentThread().getName() + " gives offer letter to candidate");
20        } catch (Exception e) {
21            e.printStackTrace();
22        }
23    }
24 }
```

```
HRManager.java x TechLead.java x
1 package com.home.lab5.task3;
2
3 import java.util.Arrays;
4 import java.util.concurrent.CyclicBarrier;
5
6 public class HRManager {
7     public static void main(String[] args) {
8         CyclicBarrier cyclicBarrier = new CyclicBarrier( parties: 3);
9         TechLead techLead1 = new TechLead(cyclicBarrier, name: "John TL");
10        TechLead techLead2 = new TechLead(cyclicBarrier, name: "Doe TL");
11        TechLead techLead3 = new TechLead(cyclicBarrier, name: "Mark TL");
12        for (TechLead techLead : Arrays.asList(techLead1, techLead2, techLead3)) {
13            techLead.start();
14        }
15        System.out.println("No work");
16    }
17 }
18
```

### Output Problema 3:

```
HRManager x
C:\Users\User\.jdk\corretto-16.0.2\bin\java.exe "-javaagent:C:\P
No work
Mark TL recruited developer
Mark TL waiting for others to complete ...
Doe TL recruited developer
Doe TL waiting for others to complete ...
John TL recruited developer
John TL waiting for others to complete ...
All finished recruiting, John TL gives offer letter to candidate
All finished recruiting, Mark TL gives offer letter to candidate
All finished recruiting, Doe TL gives offer letter to candidate
Process finished with exit code 0
```

### Problema 4

a.

```
1 package com.home.lab5.task4.a;
2
3 import java.util.concurrent.Semaphore;
4
5 public class CountSum extends Thread {
6     static volatile int n = 0;
7     Semaphore s = new Semaphore( permits: 1);
8
9     public int getN() {
10        return n;
11    }
12
13    public void run() {
14        int temp;
15        for (int i = 0; i < 100000000; i++) {
16            try {
17                s.acquire();
18                temp = n;
19                n = temp + 1;
20            } catch (InterruptedException e) {
21                e.printStackTrace();
22            } finally {
23                s.release();
24            }
25        }
26    }
27 }
```

### Output Problema 4 a

```
Main (2) x
C:\Users\user\.jdk\corretto-16.0.2
Final counter: 10587172

Process finished with exit code 0
```

b.

```
CountSum.java x Main.java x README.md x
1 package com.home.lab5.task4.b;
2
3 import java.util.concurrent.Semaphore;
4
5 public class CountSum extends Thread {
6     volatile int n = 0;
7     Semaphore s = new Semaphore( permits: 1);
8
9     public int getN() { return n; }
10
11
12
13     public void run() {
14         int temp;
15         for (int i = 0; i < 10000000; i++) {
16             try {
17                 s.acquire();
18                 temp = n;
19                 n = temp + 1;
20             } catch (InterruptedException e) {
21                 e.printStackTrace();
22             } finally {
23                 s.release();
24             }
25         }
26     }
27 }
28
```

### Output Problema 4 b

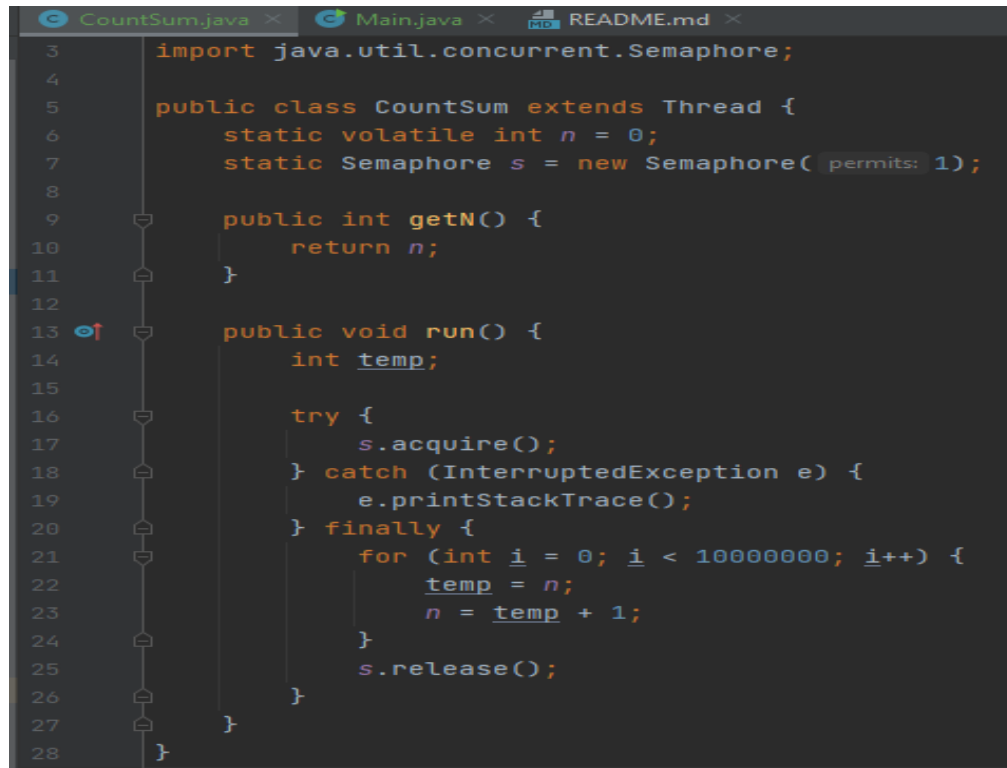
```
Main (3) x
C:\Users\user\.jdk\corretto-16.0.2
Final counter: 10000000

Process finished with exit code 0
```

### Observatii:

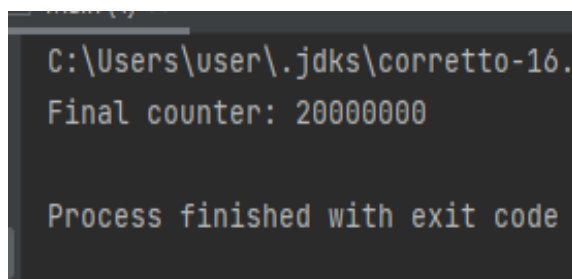
Rezultatul final va fi limitat la rezultatul unui fir, deoarece n=non-static rezulta ca rezultatul celor 2 fire este egal cu cel de la un singur fir.

c.



```
3      import java.util.concurrent.Semaphore;
4
5      public class CountSum extends Thread {
6          static volatile int n = 0;
7          static Semaphore s = new Semaphore( permits: 1);
8
9          public int getN() {
10             return n;
11         }
12
13         public void run() {
14             int temp;
15
16             try {
17                 s.acquire();
18             } catch (InterruptedException e) {
19                 e.printStackTrace();
20             } finally {
21                 for (int i = 0; i < 100000000; i++) {
22                     temp = n;
23                     n = temp + 1;
24                 }
25                 s.release();
26             }
27         }
28     }
```

#### Output Problema 4 c



```
C:\Users\user\.jdk\corretto-16.0.1
Final counter: 20000000

Process finished with exit code 0
```

#### Observatii:

Punand metoda `acquire()` inaintea buclei `for` si metoda `release()` dupa bucla rezulta ca codul va functiona bine.

d.

```
1 package com.home.lab5.task4.d;
2
3 import java.util.concurrent.Semaphore;
4
5 public class CountSum extends Thread {
6     static volatile int n = 0;
7     static Semaphore s = new Semaphore(permits: 1);
8
9     public int getN() {
10         return n;
11     }
12
13     public void run() {
14         int temp;
15         try {
16             s.acquire();
17         } catch (InterruptedException e) {
18             e.printStackTrace();
19         } finally {
20             for (int i = 0; i < 100000000; i++) {
21                 temp = n;
22                 n = temp + 1;
23             }
24         }
25     }
26 }
```

### Observatii:

In cazul de fata programul se blocheaza si nu primim rezultatul asteptat, deoarece al doilea thread nu poate obtine un permis, deoarece am eliminat declaratiile de eliberare din cod. Va continua sa astepte pana cand va putea primi un permis, prin urmare, blocand programul.