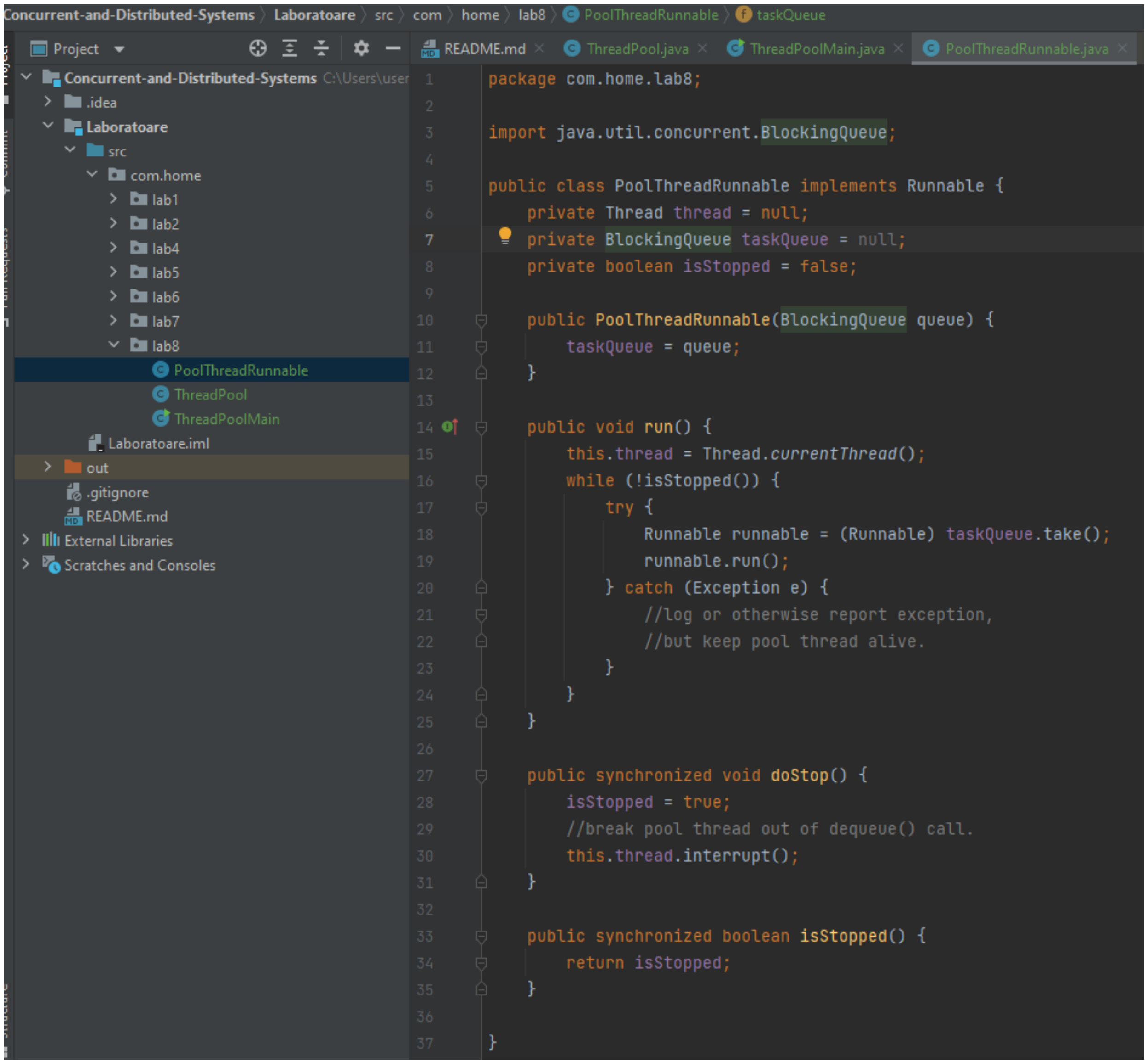
Draghici Andreea-Maria
CR 3.1B

Laborator 9

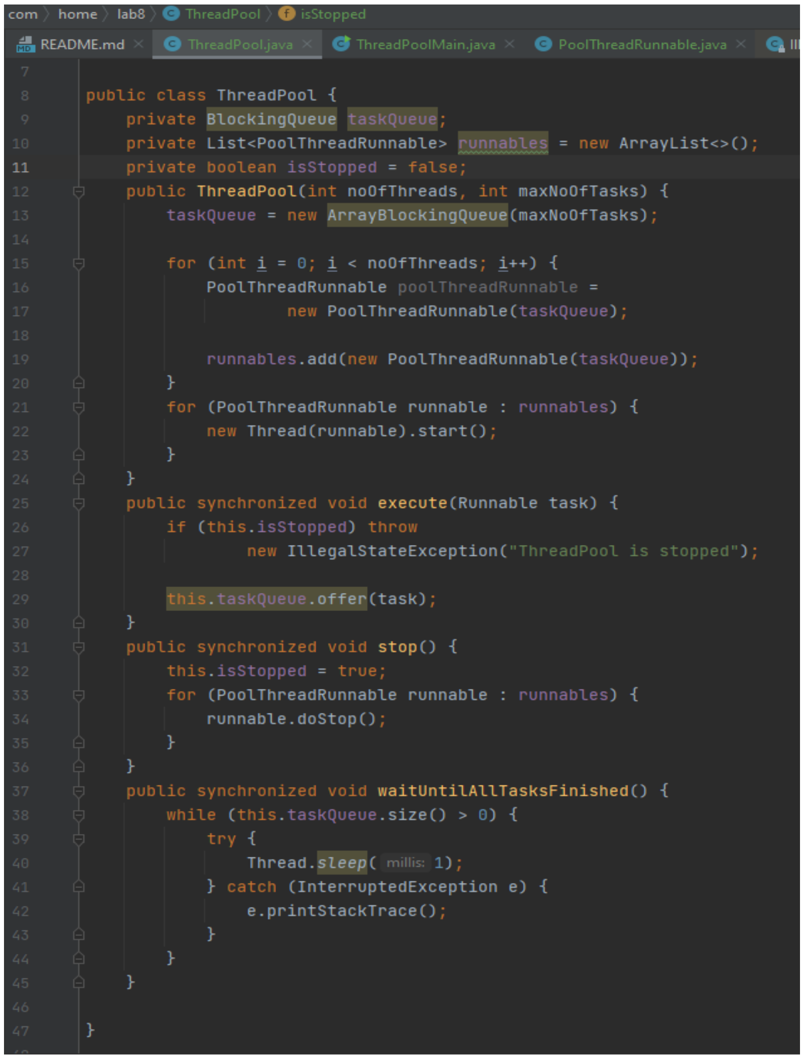
In acest laborator am creat un nou proiect si am implementat exemplul din documentul corespunzator platformei de laborator 9.

Atasez screenshot-uri cu implementarea si rezultatul obtinut.

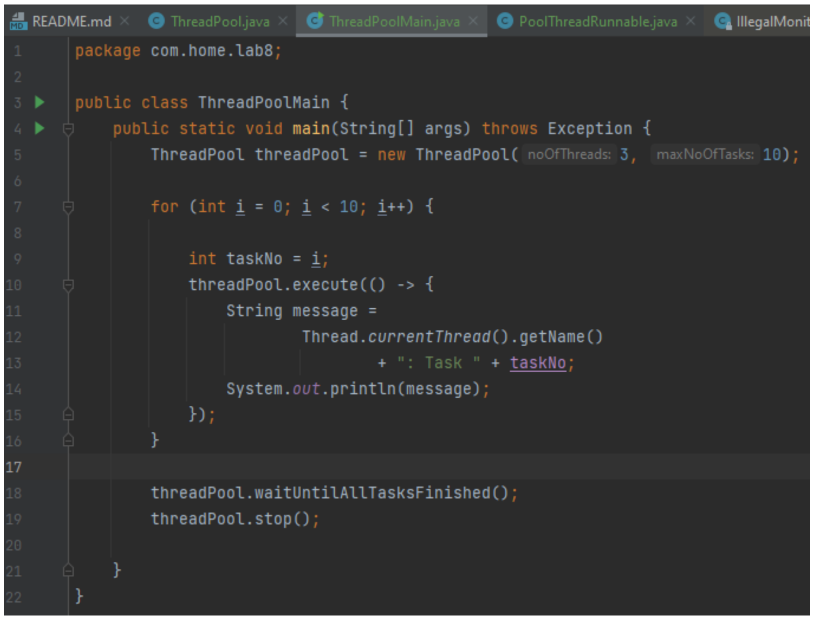


```java
package com.home.lab8;

import java.util.concurrent.BlockingQueue;

public class PoolThreadRunnable implements Runnable {
    private Thread thread = null;
    private BlockingQueue taskQueue = null;
    private boolean isStopped = false;

    public PoolThreadRunnable(BlockingQueue queue) {
        taskQueue = queue;
    }

    public void run() {
        this.thread = Thread.currentThread();
        while (!isStopped()) {
            try {
                Runnable runnable = (Runnable) taskQueue.take();
                runnable.run();
            } catch (Exception e) {
                //log or otherwise report exception,
                //but keep pool thread alive.
            }
        }
    }

    public synchronized void doStop() {
        isStopped = true;
        //break pool thread out of dequeue() call.
        this.thread.interrupt();
    }

    public synchronized boolean isStopped() {
        return isStopped;
    }

}
```

Draghici Andreea-Maria
CR 3.1B



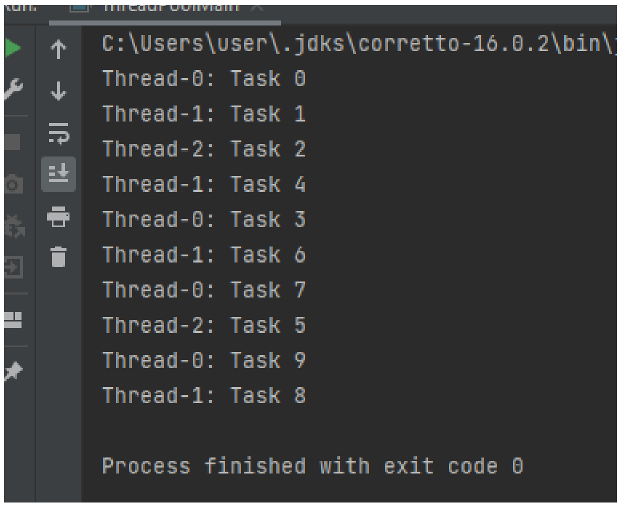
```
public class ThreadPool {
    private BlockingQueue taskQueue;
    private List<PoolThreadRunnable> runnables = new ArrayList<>();
    private boolean isStopped = false;
    public ThreadPool(int noOfThreads, int maxNoOfTasks) {
        taskQueue = new ArrayBlockingQueue(maxNoOfTasks);

        for (int i = 0; i < noOfThreads; i++) {
            PoolThreadRunnable poolThreadRunnable =
                    new PoolThreadRunnable(taskQueue);

            runnables.add(new PoolThreadRunnable(taskQueue));
        }
        for (PoolThreadRunnable runnable : runnables) {
            new Thread(runnable).start();
        }
    }
    public synchronized void execute(Runnable task) {
        if (this.isStopped) throw
                new IllegalStateException("ThreadPool is stopped");

        this.taskQueue.offer(task);
    }
    public synchronized void stop() {
        this.isStopped = true;
        for (PoolThreadRunnable runnable : runnables) {
            runnable.doStop();
        }
    }
    public synchronized void waitUntilAllTasksFinished() {
        while (this.taskQueue.size() > 0) {
            try {
                Thread.sleep( millis: 1);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

}
```



```
package com.home.lab8;

public class ThreadPoolMain {
    public static void main(String[] args) throws Exception {
        ThreadPool threadPool = new ThreadPool( noOfThreads: 3,  maxNoOfTasks: 10);

        for (int i = 0; i < 10; i++) {

            int taskNo = i;
            threadPool.execute(() -> {
                String message =
                        Thread.currentThread().getName()
                                + ": Task " + taskNo;
                System.out.println(message);
            });
        }

        threadPool.waitUntilAllTasksFinished();
        threadPool.stop();

    }
}
```

Draghici Andreea-Maria
CR 3.1B

Output:



```
C:\Users\user\.jdks\corretto-16.0.2\bin\
Thread-0: Task 0
Thread-1: Task 1
Thread-2: Task 2
Thread-1: Task 4
Thread-0: Task 3
Thread-1: Task 6
Thread-0: Task 7
Thread-2: Task 5
Thread-0: Task 9
Thread-1: Task 8

Process finished with exit code 0
```

Observatii finale:

Ordinea apelurilor pe fire este data de CPU/OS cand vine vorba de ordinea in care sunt apelate acestea.