

Free Concurrency Project1

Prepared by: Nadir ÇİFTÇİ
August, 2018





Concepts

- Thread Synchronization
- Critical Region
- User/Daemon Threads
- Volatile/Memory Barrier
- Unraveled Threads

SampleMain1.java

```
public class SampleMain1 {  
  
    public static void main(String[] args) {  
        Thread thread1 = new Thread(new Runnable() {  
            public void run() {  
                try {  
                    Thread.sleep(5000);  
                    System.out.println("Thread1 end: " + new Date());  
                } catch (InterruptedException e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
        System.out.println("Thread1 start: " + new Date());  
        thread1.start();  
    }  
}
```

Console Output

@ Javadoc  Problems  Declaration  Console  Task List

<terminated> SampleMain1 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (23 Aug 2018 14:40:28)
Thread1 start: Thu Aug 23 14:40:29 EEST 2018
Thread1 end: Thu Aug 23 14:40:34 EEST 2018

SampleMain2.java

```
public class SampleMain2 {

    private static volatile boolean isFinished = false;

    public static void main(String[] args) {
        Thread[] threads = new Thread[5];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread(new Runnable(){
                public void run() {
                    System.out.println(Thread.currentThread().getName() + " start: " + new Date());
                    int count=1;
                    while (!isFinished){
                        try {
                            System.out.println(Thread.currentThread().getName() + " sleep ... " + count++
                                + " : " + new Date());
                            Thread.sleep(5000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                    System.out.println(Thread.currentThread().getName() + " ended: " + new Date());
                }
            });
        }

        for (Thread thread : threads) {
            thread.start();
        }
    }
}
```

SampleMain2.java(continued)

```
    try {
        Thread.sleep(13000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    isFinished = true;
    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Main thread end:" + new Date());
}
}
```

Console Output

@ Javadoc Problems Declaration Console Task List

<terminated> SampleMain2 (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (23 Ağu 2018 14:58:27)

```
Thread-0 start: Thu Aug 23 14:58:28 EEST 2018
Thread-2 start: Thu Aug 23 14:58:28 EEST 2018
Thread-0 sleep ... 1 : Thu Aug 23 14:58:28 EEST 2018
Thread-4 start: Thu Aug 23 14:58:28 EEST 2018
Thread-1 start: Thu Aug 23 14:58:28 EEST 2018
Thread-3 start: Thu Aug 23 14:58:28 EEST 2018
Thread-1 sleep ... 1 : Thu Aug 23 14:58:28 EEST 2018
Thread-4 sleep ... 1 : Thu Aug 23 14:58:28 EEST 2018
Thread-2 sleep ... 1 : Thu Aug 23 14:58:28 EEST 2018
Thread-3 sleep ... 1 : Thu Aug 23 14:58:28 EEST 2018
Thread-0 sleep ... 2 : Thu Aug 23 14:58:33 EEST 2018
Thread-4 sleep ... 2 : Thu Aug 23 14:58:33 EEST 2018
Thread-2 sleep ... 2 : Thu Aug 23 14:58:33 EEST 2018
Thread-3 sleep ... 2 : Thu Aug 23 14:58:33 EEST 2018
Thread-1 sleep ... 2 : Thu Aug 23 14:58:33 EEST 2018
Thread-0 sleep ... 3 : Thu Aug 23 14:58:38 EEST 2018
Thread-2 sleep ... 3 : Thu Aug 23 14:58:38 EEST 2018
Thread-3 sleep ... 3 : Thu Aug 23 14:58:38 EEST 2018
Thread-1 sleep ... 3 : Thu Aug 23 14:58:38 EEST 2018
Thread-4 sleep ... 3 : Thu Aug 23 14:58:38 EEST 2018
Thread-0 ended: Thu Aug 23 14:58:43 EEST 2018
Thread-4 ended: Thu Aug 23 14:58:43 EEST 2018
Thread-2 ended: Thu Aug 23 14:58:43 EEST 2018
Thread-1 ended: Thu Aug 23 14:58:43 EEST 2018
Thread-3 ended: Thu Aug 23 14:58:43 EEST 2018
Main thread end: Thu Aug 23 14:58:46 EEST 2018
```


SampleMain3.java

```
public class SampleMain3 {  
  
    public static void main(String[] args) {  
        final SampleModel3 sampleModel3 = new SampleModel3();  
        Thread[] threads = new Thread[10];  
        for (int i = 0; i < threads.length; i++) {  
            threads[i] = new Thread(new Runnable() {  
                public void run() {  
                    sampleModel3.incrementCounter();  
                }  
            });  
        }  
        for (Thread thread : threads) {  
            thread.start();  
        }  
        try {  
            TimeUnit.SECONDS.sleep(3);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
        System.out.println(sampleModel3.getCounter());  
    }  
}
```


SampleModel3.java

```
public class SampleModel3 {  
  
    private int counter = 0;  
  
    public synchronized void incrementCounter() {  
        counter = counter + 1;  
    }  
  
    public int getCounter() {  
        return counter;  
    }  
}
```

Console output

@ Javadoc  Problems  Declaration  Console  Task List

<terminated> SampleMain3 (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Aug 2018 13:00:58)

10

SampleMain4.java

```
public class SampleMain4 {  
  
    public static void main(String[] args) {  
        Lock lock = new ReentrantLock();  
  
        Thread[] threads = new Thread[5];  
        for (int i = 0; i < threads.length; i++) {  
            threads[i] = new SampleThread4(lock);  
        }  
        for (Thread thread : threads) {  
            thread.start();  
        }  
    }  
}
```

SampleThread4.java

```
public class SampleThread4 extends Thread{

    private Lock lock;

    public SampleThread4(Lock lock) {
        this.lock = lock;
    }

    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " start: " + new Date());
        try {
            while( !lock.tryLock(3, TimeUnit.SECONDS) ){
                System.out.println(Thread.currentThread().getName() + " retry acquiring lock... "
                    + new Date());
            }
            System.out.println(Thread.currentThread().getName() + " acquired lock! " + new Date());
        } catch (InterruptedException e1) {
            e1.printStackTrace();
            return;
        }
        try{
            long sleepSeconds = (long)(Math.random() * 4) ;
            System.out.println(Thread.currentThread().getName() + " sleep for " + sleepSeconds
                + " seconds...");
            TimeUnit.SECONDS.sleep( sleepSeconds);
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

SampleThread4.java(continued)

```
    }finally{
        System.out.println(Thread.currentThread().getName() + " release lock! " +new Date());
        lock.unlock();
    }
    System.out.println(Thread.currentThread().getName() + " end: " + new Date());
}
}
```

Console Output

@ Javadoc Problems Declaration Console Task List

<terminated> SampleMain4 (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Aug 2018 13:04:03)

```
Thread-2 start: Thu Aug 30 13:04:03 EEST 2018
Thread-3 start: Thu Aug 30 13:04:03 EEST 2018
Thread-0 start: Thu Aug 30 13:04:03 EEST 2018
Thread-4 start: Thu Aug 30 13:04:03 EEST 2018
Thread-1 start: Thu Aug 30 13:04:03 EEST 2018
Thread-2 acquired lock! Thu Aug 30 13:04:03 EEST 2018
Thread-2 sleep for 2 seconds...
Thread-2 release lock! Thu Aug 30 13:04:05 EEST 2018
Thread-2 end: Thu Aug 30 13:04:05 EEST 2018
Thread-3 acquired lock! Thu Aug 30 13:04:05 EEST 2018
Thread-3 sleep for 2 seconds...
Thread-4 retry acquiring lock... Thu Aug 30 13:04:06 EEST 2018
Thread-0 retry acquiring lock... Thu Aug 30 13:04:06 EEST 2018
Thread-1 retry acquiring lock... Thu Aug 30 13:04:06 EEST 2018
Thread-3 release lock! Thu Aug 30 13:04:07 EEST 2018
Thread-3 end: Thu Aug 30 13:04:07 EEST 2018
Thread-4 acquired lock! Thu Aug 30 13:04:07 EEST 2018
Thread-4 sleep for 1 seconds...
Thread-4 release lock! Thu Aug 30 13:04:08 EEST 2018
Thread-4 end: Thu Aug 30 13:04:08 EEST 2018
Thread-0 acquired lock! Thu Aug 30 13:04:08 EEST 2018
Thread-0 sleep for 1 seconds...
Thread-1 retry acquiring lock... Thu Aug 30 13:04:09 EEST 2018
Thread-0 release lock! Thu Aug 30 13:04:09 EEST 2018
Thread-0 end: Thu Aug 30 13:04:09 EEST 2018
Thread-1 acquired lock! Thu Aug 30 13:04:09 EEST 2018
Thread-1 sleep for 3 seconds...
Thread-1 release lock! Thu Aug 30 13:04:12 EEST 2018
Thread-1 end: Thu Aug 30 13:04:12 EEST 2018
```

SampleMain5.java

```
public class SampleMain5 {  
  
    public static void main(String[] args) {  
        ThreadPoolExecutor executor = (ThreadPoolExecutor) Executors.newFixedThreadPool(5);  
        for (int i=0; i<10 ; i++ ) {  
            SampleTask5 sampleTask5 = new SampleTask5(i+1);  
            executor.execute(sampleTask5);  
        }  
        executor.shutdown();  
        try {  
            executor.awaitTermination(60, TimeUnit.SECONDS);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
        System.out.println(Thread.currentThread().getName() + " end: " + new Date());  
    }  
}
```

SampleTask5.java

```
public class SampleTask5 implements Runnable {

    private int taskNo;

    public SampleTask5(int taskNo) {
        this.taskNo = taskNo;
    }

    public void run() {
        System.out.println(Thread.currentThread().getName() + " - task no: " + taskNo + " start: "
            + new Date());
        try{
            long sleepSeconds = (long)(Math.random() * 4) ;
            System.out.println(Thread.currentThread().getName() + " - task no: " + taskNo + " sleep for "
                + sleepSeconds + " seconds...");
            TimeUnit.SECONDS.sleep( sleepSeconds);
        }catch(Exception e){
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " - task no: " + taskNo + " end: "
            + new Date());
    }
}
```


Console Output

@ Javadoc Problems Declaration Console Task List

<terminated> SampleMain5 (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Aug 2018 13:06:28)

```
pool-1-thread-1 - task no: 1 start: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-1 - task no: 1 sleep for 3 seconds...
pool-1-thread-2 - task no: 2 start: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-2 - task no: 2 sleep for 1 seconds...
pool-1-thread-5 - task no: 5 start: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-5 - task no: 5 sleep for 3 seconds...
pool-1-thread-4 - task no: 4 start: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-4 - task no: 4 sleep for 3 seconds...
pool-1-thread-3 - task no: 3 start: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-3 - task no: 3 sleep for 0 seconds...
pool-1-thread-3 - task no: 3 end: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-3 - task no: 6 start: Thu Aug 30 13:06:28 EEST 2018
pool-1-thread-3 - task no: 6 sleep for 3 seconds...
pool-1-thread-2 - task no: 2 end: Thu Aug 30 13:06:29 EEST 2018
pool-1-thread-2 - task no: 7 start: Thu Aug 30 13:06:29 EEST 2018
pool-1-thread-2 - task no: 7 sleep for 3 seconds...
pool-1-thread-1 - task no: 1 end: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-1 - task no: 8 start: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-1 - task no: 8 sleep for 1 seconds...
pool-1-thread-4 - task no: 4 end: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-5 - task no: 5 end: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-4 - task no: 9 start: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-5 - task no: 10 start: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-5 - task no: 10 sleep for 2 seconds...
pool-1-thread-4 - task no: 9 sleep for 0 seconds...
pool-1-thread-4 - task no: 9 end: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-3 - task no: 6 end: Thu Aug 30 13:06:31 EEST 2018
pool-1-thread-1 - task no: 8 end: Thu Aug 30 13:06:32 EEST 2018
pool-1-thread-2 - task no: 7 end: Thu Aug 30 13:06:32 EEST 2018
pool-1-thread-5 - task no: 10 end: Thu Aug 30 13:06:33 EEST 2018
main end: Thu Aug 30 13:06:33 EEST 2018
```

SampleMain6.java

```
public class SampleMain6 {

    public static void main(String[] args) {
        ThreadPoolExecutor executor = (ThreadPoolExecutor) Executors.newFixedThreadPool(5);
        ArrayList<Future<Long>> results = new ArrayList<>();
        for (int i=0; i<10 ; i++ ) {
            SampleTask6 sampleTask6 = new SampleTask6(i+1);
            results.add(executor.submit(sampleTask6));
        }
        for (int i = 0; i < results.size(); i++) {
            Future<Long> future = results.get(i);
            try {
                System.out.println("Task no: " + (i+1) + " slept for " + future.get(10, TimeUnit.SECONDS)
                    + " seconds.");
            } catch (InterruptedException | ExecutionException | TimeoutException e) {
                e.printStackTrace();
            }
        }
        executor.shutdown();
        try {
            executor.awaitTermination(60, TimeUnit.SECONDS);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " end: " + new Date());
    }
}
```

SampleTask6.java

```
public class SampleTask6 implements Callable<Long> {

    private int taskNo;

    public SampleTask6(int taskNo) {
        this.taskNo = taskNo;
    }

    @Override
    public Long call() throws Exception {
        System.out.println(Thread.currentThread().getName() + " - task no: " + taskNo + " start: "
            + new Date());
        long sleepSeconds = (long) (Math.random() * 4) ;
        try{
            System.out.println(Thread.currentThread().getName() + " - task no: " + taskNo + " sleep for "
                + sleepSeconds + " seconds...");
            TimeUnit.SECONDS.sleep( sleepSeconds);
        } catch(Exception e){
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " - task no: " + taskNo + " end: "
            + new Date());
        return sleepSeconds;
    }

}
```

Console Output

@ Javadoc Problems Declaration Console Task List

<terminated> SampleMain6 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Ağ 2018 13:09:35)

```
pool-1-thread-3 - task no: 3 start: Thu Aug 30 13:09:35 EEST 2018
pool-1-thread-2 - task no: 2 start: Thu Aug 30 13:09:35 EEST 2018
pool-1-thread-1 - task no: 1 start: Thu Aug 30 13:09:35 EEST 2018
pool-1-thread-5 - task no: 5 start: Thu Aug 30 13:09:35 EEST 2018
pool-1-thread-4 - task no: 4 start: Thu Aug 30 13:09:35 EEST 2018
pool-1-thread-3 - task no: 3 sleep for 1 seconds...
pool-1-thread-5 - task no: 5 sleep for 3 seconds...
pool-1-thread-1 - task no: 1 sleep for 0 seconds...
pool-1-thread-4 - task no: 4 sleep for 3 seconds...
pool-1-thread-2 - task no: 2 sleep for 0 seconds...
pool-1-thread-2 - task no: 2 end: Thu Aug 30 13:09:36 EEST 2018
pool-1-thread-1 - task no: 1 end: Thu Aug 30 13:09:36 EEST 2018
pool-1-thread-2 - task no: 6 start: Thu Aug 30 13:09:36 EEST 2018
pool-1-thread-2 - task no: 6 sleep for 2 seconds...
pool-1-thread-1 - task no: 7 start: Thu Aug 30 13:09:36 EEST 2018
pool-1-thread-1 - task no: 7 sleep for 0 seconds...
pool-1-thread-1 - task no: 7 end: Thu Aug 30 13:09:36 EEST 2018
pool-1-thread-1 - task no: 8 start: Thu Aug 30 13:09:36 EEST 2018
pool-1-thread-1 - task no: 8 sleep for 3 seconds...
Task no: 1 slept for 0 seconds.
Task no: 2 slept for 0 seconds.
pool-1-thread-3 - task no: 3 end: Thu Aug 30 13:09:37 EEST 2018
Task no: 3 slept for 1 seconds.
pool-1-thread-3 - task no: 9 start: Thu Aug 30 13:09:37 EEST 2018
pool-1-thread-3 - task no: 9 sleep for 2 seconds...
pool-1-thread-2 - task no: 6 end: Thu Aug 30 13:09:38 EEST 2018
pool-1-thread-2 - task no: 10 start: Thu Aug 30 13:09:38 EEST 2018
pool-1-thread-2 - task no: 10 sleep for 3 seconds...
pool-1-thread-5 - task no: 5 end: Thu Aug 30 13:09:39 EEST 2018
pool-1-thread-4 - task no: 4 end: Thu Aug 30 13:09:39 EEST 2018
Task no: 4 slept for 3 seconds.
Task no: 5 slept for 3 seconds.
Task no: 6 slept for 2 seconds.
Task no: 7 slept for 0 seconds.
pool-1-thread-1 - task no: 8 end: Thu Aug 30 13:09:39 EEST 2018
pool-1-thread-3 - task no: 9 end: Thu Aug 30 13:09:39 EEST 2018
Task no: 8 slept for 3 seconds.
Task no: 9 slept for 2 seconds.
pool-1-thread-2 - task no: 10 end: Thu Aug 30 13:09:41 EEST 2018
Task no: 10 slept for 3 seconds.
main end: Thu Aug 30 13:09:41 EEST 2018
```

SampleMain7.java

```
public class SampleMain7 {

    public static void main(String[] args) {
        ScheduledExecutorService executor = Executors.newScheduledThreadPool(1);
        SampleTask7 sampleTask7 = new SampleTask7();
        ScheduledFuture<?> result=executor.scheduleAtFixedRate(sampleTask7, 1, 5, TimeUnit.SECONDS);
        try {
            TimeUnit.SECONDS.sleep(30);
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        System.out.println("Next execution of task starts in " + result.getDelay(TimeUnit.MILLISECONDS)
            + " milliseconds.");
        try {
            TimeUnit.MILLISECONDS.sleep(result.getDelay(TimeUnit.MILLISECONDS) - 10);
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        executor.shutdown();
        try {
            executor.awaitTermination(60, TimeUnit.SECONDS);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " end: " + new Date());
    }

}
```

SampleTask7.java

```
public class SampleTask7 implements Runnable {

    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " start: " + new Date());
        long sleepSeconds = (long) (Math.random() * 4) ;
        try{
            System.out.println(Thread.currentThread().getName() + " sleep for " + sleepSeconds
                               + " seconds...");
            TimeUnit.SECONDS.sleep( sleepSeconds);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " end: " + new Date());
    }
}
```

Console Output

@ Javadoc  Problems  Declaration  Console  Task List

<terminated> SampleMain7 (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Aug 2018 13:11:58)

```
pool-1-thread-1 start: Thu Aug 30 13:11:59 EEST 2018
pool-1-thread-1 sleep for 2 seconds...
pool-1-thread-1 end: Thu Aug 30 13:12:01 EEST 2018
pool-1-thread-1 start: Thu Aug 30 13:12:04 EEST 2018
pool-1-thread-1 sleep for 3 seconds...
pool-1-thread-1 end: Thu Aug 30 13:12:07 EEST 2018
pool-1-thread-1 start: Thu Aug 30 13:12:09 EEST 2018
pool-1-thread-1 sleep for 0 seconds...
pool-1-thread-1 end: Thu Aug 30 13:12:09 EEST 2018
pool-1-thread-1 start: Thu Aug 30 13:12:14 EEST 2018
pool-1-thread-1 sleep for 0 seconds...
pool-1-thread-1 end: Thu Aug 30 13:12:14 EEST 2018
pool-1-thread-1 start: Thu Aug 30 13:12:19 EEST 2018
pool-1-thread-1 sleep for 0 seconds...
pool-1-thread-1 end: Thu Aug 30 13:12:19 EEST 2018
pool-1-thread-1 start: Thu Aug 30 13:12:24 EEST 2018
pool-1-thread-1 sleep for 1 seconds...
pool-1-thread-1 end: Thu Aug 30 13:12:25 EEST 2018
Next execution of task starts in 999 milliseconds.
main end: Thu Aug 30 13:12:29 EEST 2018
```

SampleMain8.java

```
public class SampleMain8 {

    public static void main(String[] args) {
        long numberOfItemsToProcess = 104;
        int numberOfParallelTasks = 5;
        int chunkSize = 5; //should be greater than the number of parallel tasks
        SampleTask8 sampleTask8 = new SampleTask8(numberOfItemsToProcess, numberOfParallelTasks
            , chunkSize);
        ForkJoinPool joinForkPool = new ForkJoinPool();
        joinForkPool.execute(sampleTask8);
        while (!sampleTask8.isDone()) {
            try {
                TimeUnit.MILLISECONDS.sleep(50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println(Thread.currentThread().getName() + " Steal Count: "
            + joinForkPool.getStealCount());
        joinForkPool.shutdown();
        System.out.println(Thread.currentThread().getName() + " end: " + new Date());
    }
}
```


SampleTask8.java

```
public class SampleTask8 extends RecursiveAction {

    private static final long serialVersionUID = 1852883230181599349L;
    private long numberOfItemsToProcess;
    private int numberOfParallelTasks;
    private int chunkSize;

    public SampleTask8(long numberOfItemsToProcess, int numberOfParallelTasks, int chunkSize) {
        this.numberOfItemsToProcess = numberOfItemsToProcess;
        this.numberOfParallelTasks = numberOfParallelTasks;
        this.chunkSize = chunkSize;
    }

    @Override
    protected void compute() {
        if ( numberOfItemsToProcess <= chunkSize) {
            System.out.println(this + " will compute itself " + numberOfItemsToProcess + " items...");
            long sleepSeconds = (long) Math.random() * chunkSize;
            try {
                TimeUnit.SECONDS.sleep(sleepSeconds);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println(this + " will divide tasks for " + numberOfItemsToProcess + " items...");
            long numberOfItemsPerTask = numberOfItemsToProcess / numberOfParallelTasks;
            long remainingItemsToProcess = numberOfItemsToProcess;
            SampleTask8 sampleTask8 = null;
```

SampleTask8.java(continued)

```
ArrayList<SampleTask8> sampleTask8List = new ArrayList<>();
for (int i=0; i< numberOfParallelTasks; i++){
    if (i < numberOfParallelTasks -1 ){
        sampleTask8 = new SampleTask8(numberOfItemsPerTask, numberOfParallelTasks, chunkSize);
        remainingItemsToProcess -= numberOfItemsPerTask;
        sampleTask8List.add(sampleTask8);
    }else{
        sampleTask8 = new SampleTask8(remainingItemsToProcess, numberOfParallelTasks
            , chunkSize);
        sampleTask8List.add(sampleTask8);
        break;
    }
}
invokeAll (sampleTask8List);
}
}
```

Console Output

@ Javadoc Problems Declaration Console Task List

```
<terminated> SampleMain8 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Ağu 2018 13:14:45)
com.ndr.free.concurrency.model.SampleTask8@67d479cf will divide tasks for 104 items...
com.ndr.free.concurrency.model.SampleTask8@20e90906 will divide tasks for 20 items...
com.ndr.free.concurrency.model.SampleTask8@36c51089 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@43c0ae76 will divide tasks for 20 items...
com.ndr.free.concurrency.model.SampleTask8@1efde7ba will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@2f78743b will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@5a4b4b50 will divide tasks for 24 items...
com.ndr.free.concurrency.model.SampleTask8@53d9f80 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@d16e5d6 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@3ad6a0e0 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@60dbf04d will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@77d80e9 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@409a44d6 will divide tasks for 8 items...
com.ndr.free.concurrency.model.SampleTask8@10b28f30 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@3836b1bb will compute itself 1 items...
com.ndr.free.concurrency.model.SampleTask8@19789a96 will compute itself 1 items...
com.ndr.free.concurrency.model.SampleTask8@6a4d37e5 will compute itself 1 items...
com.ndr.free.concurrency.model.SampleTask8@7390d1e8 will compute itself 1 items...
com.ndr.free.concurrency.model.SampleTask8@501d5ebc will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@28c5119e will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@2207b0fb will divide tasks for 20 items...
com.ndr.free.concurrency.model.SampleTask8@3cecfaea will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@ece88d2 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@a16b7c will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@65979a36 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@440d8355 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@40914272 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@427b7b5d will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@6da264f1 will divide tasks for 20 items...
com.ndr.free.concurrency.model.SampleTask8@36422510 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@308f5944 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@132d9844 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@1667a232 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@26f44031 will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@5329645a will compute itself 4 items...
com.ndr.free.concurrency.model.SampleTask8@6dc98c1b will compute itself 4 items...
main Steal Count: 13
main end: Thu Aug 30 13:14:45 EEST 2018
```

SampleMain9.java

```
public class SampleMain9 {

    public static void main(String[] args) {
        ExecutorService threadPool = new ScheduledThreadPoolExecutor(5);
        AsynchronousFileChannel fileChannel = null;
        try {
            HashSet<StandardOpenOption> openOptionSet = new HashSet<>();
            openOptionSet.add(StandardOpenOption.READ);
            fileChannel = AsynchronousFileChannel.open(Paths.get(Paths.get("").toAbsolutePath().toString()
                + "\\src\\main\\resources\\SampleFile.txt"), openOptionSet, threadPool);
        } catch (IOException e) {
            e.printStackTrace();
        }

        final int handlerCount = 10;
        SampleCompletionHandler[] handlers = new SampleCompletionHandler[handlerCount];
        for (int i = 0; i < handlers.length; i++) {
            handlers[i] = new SampleCompletionHandler();
        }

        final int bufferCount = 10;
        ByteBuffer buffers[] = new ByteBuffer[bufferCount];
        for (int i = 0; i < bufferCount; i++) {
            buffers[i] = ByteBuffer.allocate(10);
            fileChannel.read(buffers[i], i * 10, null, handlers[i % 5] );
        }

        try {
            threadPool.awaitTermination(3, TimeUnit.SECONDS);
        }
```

SampleMain9.java(continued)

```
    } catch (InterruptedException e1) {
        e1.printStackTrace();
    }

    threadPool.shutdown();

    for (ByteBuffer byteBuffer : buffers) {
        for (int i = 0; i < byteBuffer.limit(); i++) {
            System.out.print((char) byteBuffer.get(i));
        }
        System.out.println("");
    }
}
```

SampleCompletionHandler.Java

```
public class SampleCompletionHandler implements CompletionHandler<Integer, ByteBuffer> {

    @Override
    public void completed(Integer result, ByteBuffer attachment) {
        System.out.println(Thread.currentThread().getName() + " completed..." + result
            + " bytes read!");
    }




    @Override
    public void failed(Throwable e, ByteBuffer byteBuffer) {
        e.printStackTrace();
    }

}
```

SampleFile.txt

0123456789123456789023456789013456789012456789012356789012346789012345789012345689012345679012345678

Console Output

@ Javadoc  Problems  Declaration  Console  Task List

<terminated> SampleMain9 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30 Aug 2018 13:16:52)

```
pool-1-thread-1 completed...10 bytes read!  
pool-1-thread-2 completed...10 bytes read!  
pool-1-thread-2 completed...10 bytes read!  
pool-1-thread-1 completed...10 bytes read!  
pool-1-thread-2 completed...10 bytes read!  
pool-1-thread-3 completed...10 bytes read!  
pool-1-thread-3 completed...10 bytes read!  
pool-1-thread-3 completed...10 bytes read!  
pool-1-thread-3 completed...10 bytes read!  
pool-1-thread-4 completed...10 bytes read!  
0123456789  
1234567890  
2345678901  
3456789012  
4567890123  
5678901234  
6789012345  
7890123456  
8901234567  
9012345678
```