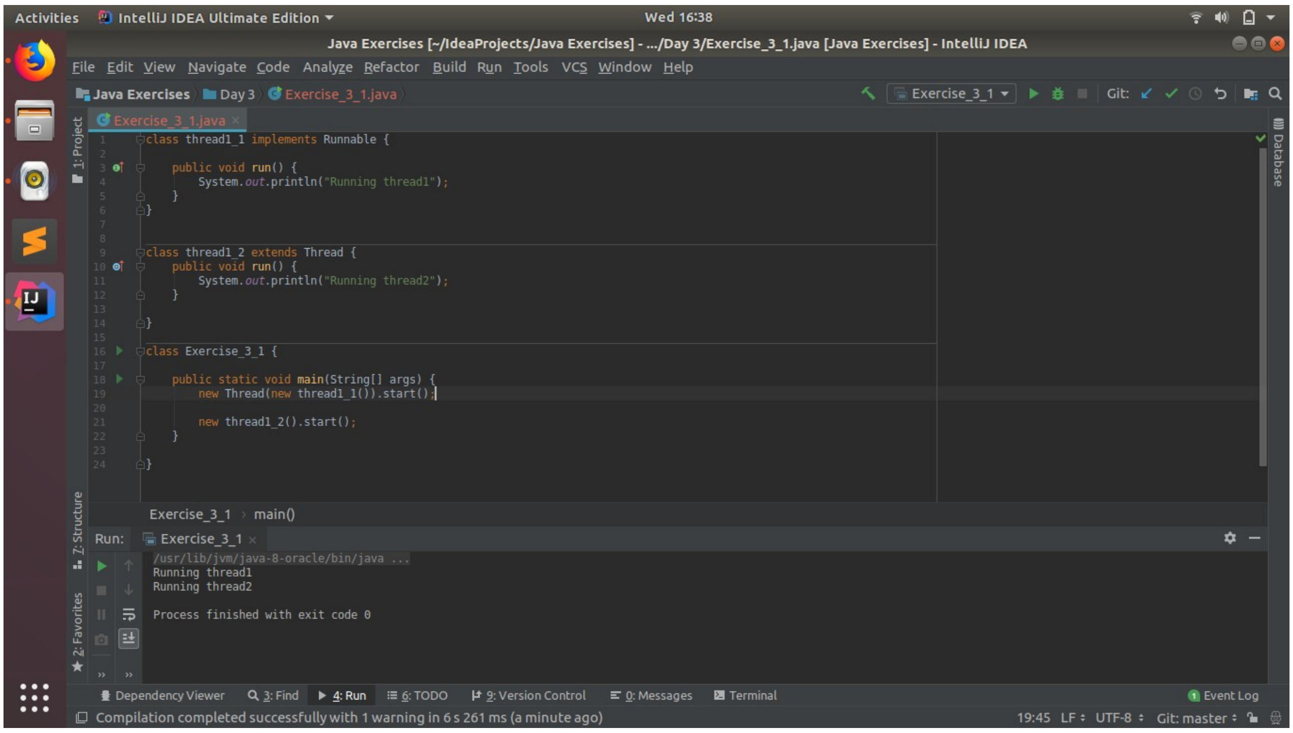
Q1. Create and Run a Thread using Runnable Interface and Thread class.
A1.



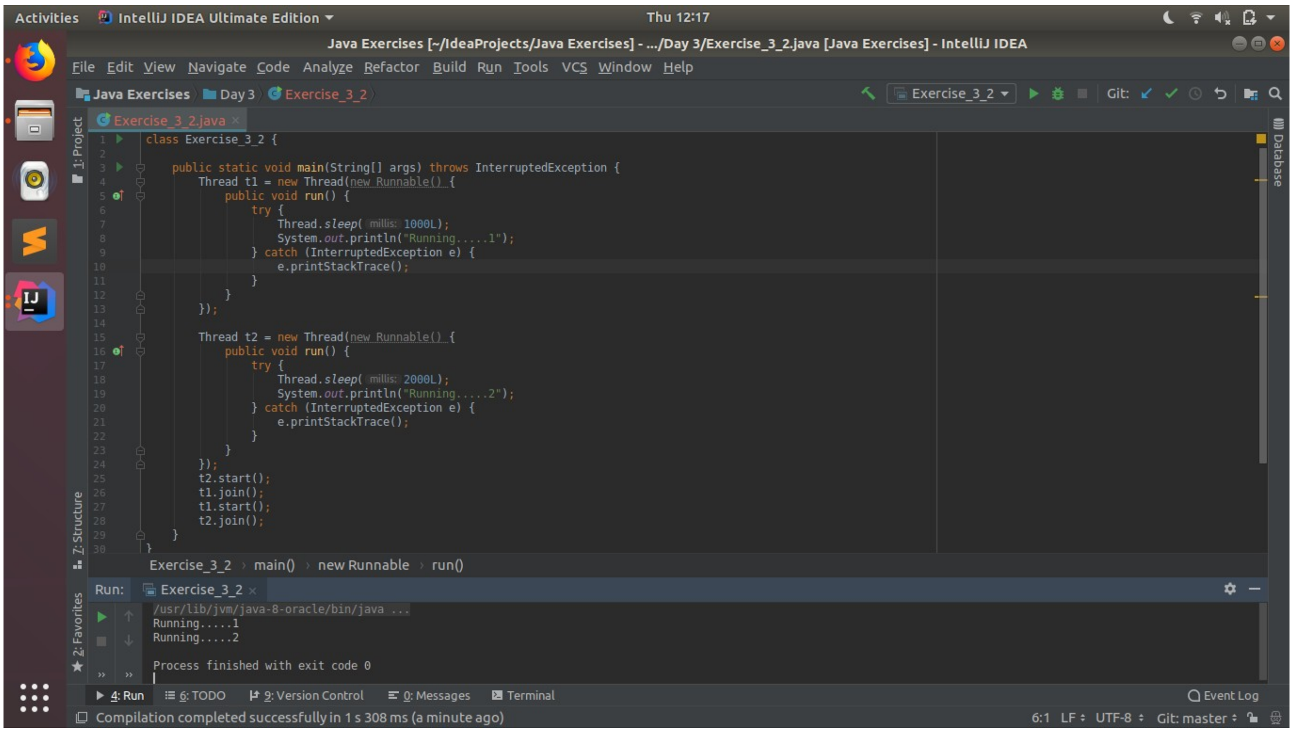
Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_1.java [Java Exercises] - IntelliJ IDEA

```java
class thread1_1 implements Runnable {

    public void run() {
        System.out.println("Running thread1");
    }
}


class thread1_2 extends Thread {
    public void run() {
        System.out.println("Running thread2");
    }

}

class Exercise_3_1 {

    public static void main(String[] args) {
        new Thread(new thread1_1()).start();

        new thread1_2().start();
    }

}
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Running thread1
Running thread2

Process finished with exit code 0
```

Compilation completed successfully with 1 warning in 6 s 261 ms (a minute ago)

Q2. Use sleep and join methods with thread.
A2.



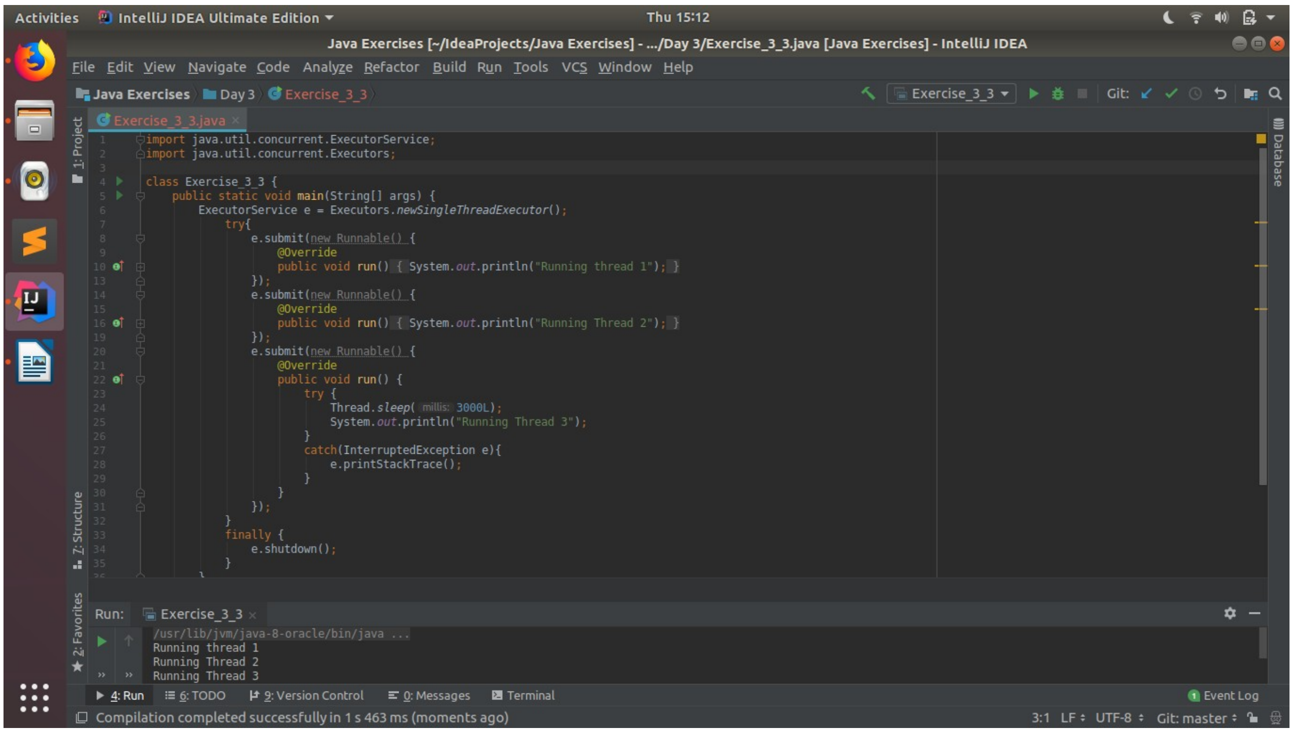
Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_2.java [Java Exercises] - IntelliJ IDEA

```java
class Exercise_3_2 {

    public static void main(String[] args) throws InterruptedException {
        Thread t1 = new Thread(new Runnable() {
            public void run() {
                try {
                    Thread.sleep( millis: 1000L);
                    System.out.println("Running.....1");
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
        });

        Thread t2 = new Thread(new Runnable() {
            public void run() {
                try {
                    Thread.sleep( millis: 2000L);
                    System.out.println("Running.....2");
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
        });
        t2.start();
        t1.join();
        t1.start();
        t2.join();
    }
}
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Running.....1
Running.....2

Process finished with exit code 0
```

Compilation completed successfully in 1 s 308 ms (a minute ago)

Q3. Use a singleThreadExecutor to submit multiple threads.
A3.

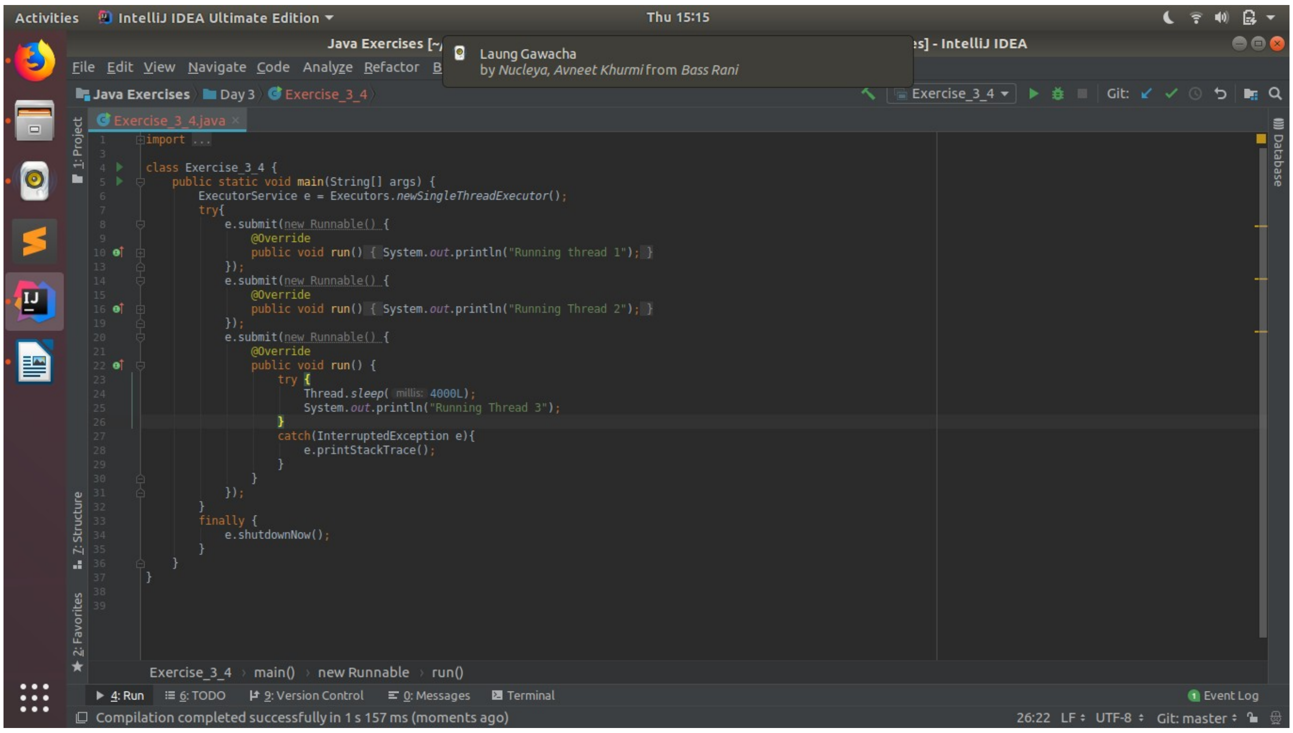


```java
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Exercise_3_3 {
    public static void main(String[] args) {
        ExecutorService e = Executors.newSingleThreadExecutor();
            try{
                e.submit(new Runnable() {
                    @Override
                    public void run() { System.out.println("Running thread 1"); }
                });
                e.submit(new Runnable() {
                    @Override
                    public void run() { System.out.println("Running Thread 2"); }
                });
                e.submit(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            Thread.sleep( millis: 3000L);
                            System.out.println("Running Thread 3");
                        }
                        catch(InterruptedException e){
                            e.printStackTrace();
                        }
                    }
                });
            }
            finally {
                e.shutdown();
            }
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Running thread 1
Running Thread 2
Running Thread 3
```

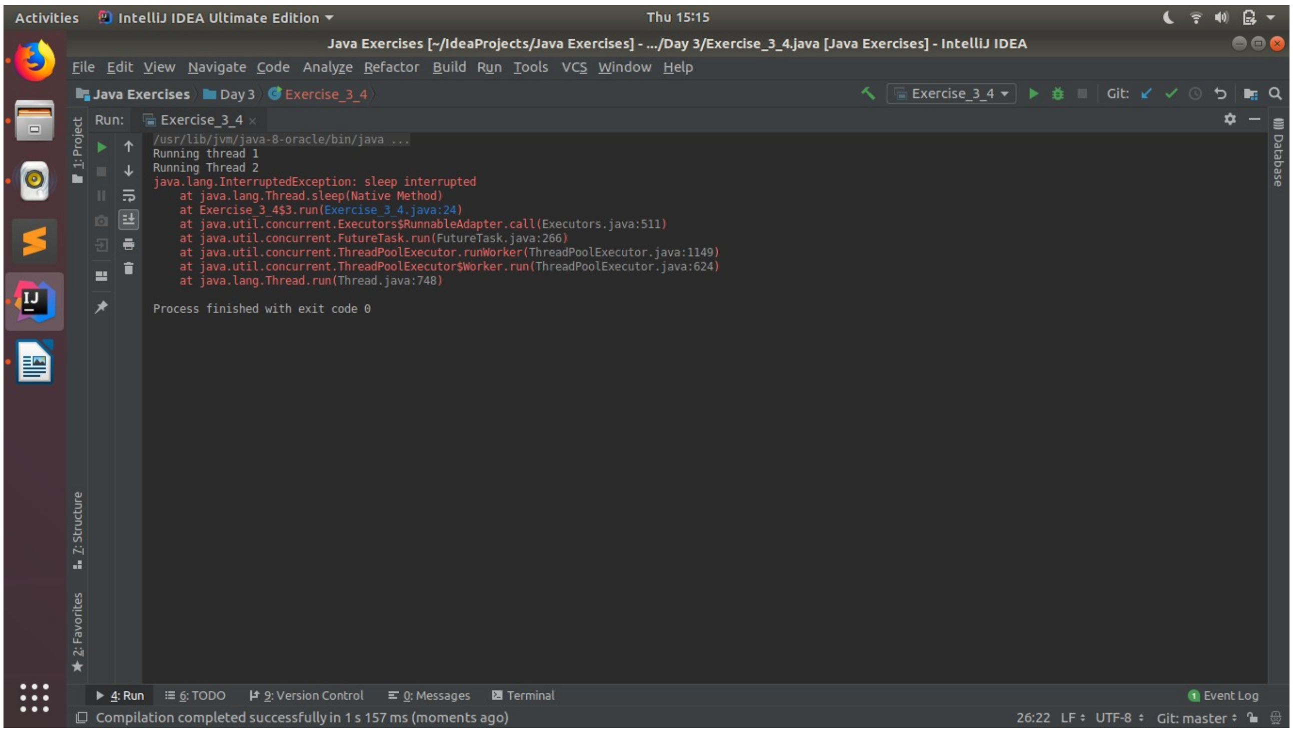
Compilation completed successfully in 1 s 463 ms (moments ago)

Q4. Try shutdown() and shutdownNow() and observe the difference.
A4.



```java
import ...

class Exercise_3_4 {
    public static void main(String[] args) {
        ExecutorService e = Executors.newSingleThreadExecutor();
        try{
            e.submit(new Runnable() {
                @Override
                public void run() { System.out.println("Running thread 1"); }
            });
            e.submit(new Runnable() {
                @Override
                public void run() { System.out.println("Running Thread 2"); }
            });
            e.submit(new Runnable() {
                @Override
                public void run() {
                    try {
                        Thread.sleep( millis: 4000L);
                        System.out.println("Running Thread 3");
                    }
                    catch(InterruptedException e){
                        e.printStackTrace();
                    }
                }
            });
        }
        finally {
            e.shutdownNow();
        }
    }
}
```

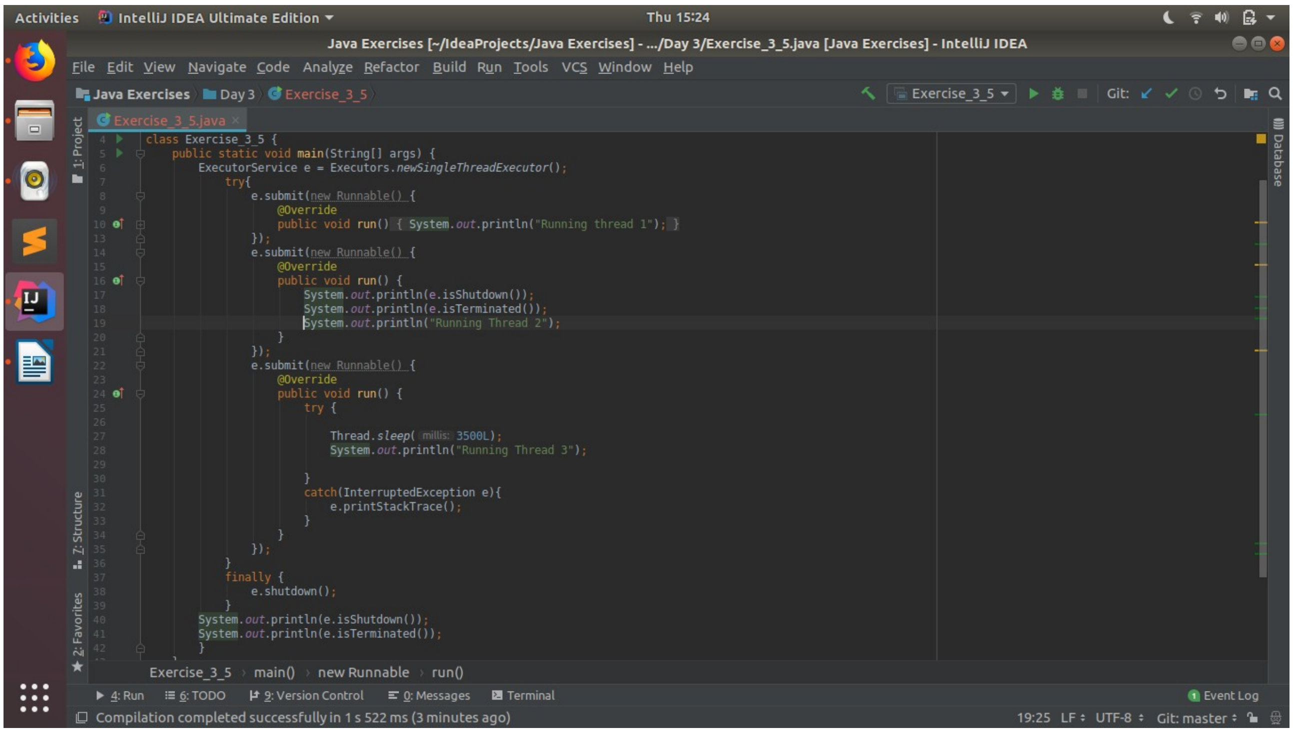
Compilation completed successfully in 1 s 157 ms (moments ago)


Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_4.java [Java Exercises] - IntelliJ IDEA

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Running thread 1
Running Thread 2
java.lang.InterruptedException: sleep interrupted
	at java.lang.Thread.sleep(Native Method)
	at Exercise_3_4$3.run(Exercise_3_4.java:24)
	at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
	at java.util.concurrent.FutureTask.run(FutureTask.java:266)
	at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
	at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
	at java.lang.Thread.run(Thread.java:748)

Process finished with exit code 0
```

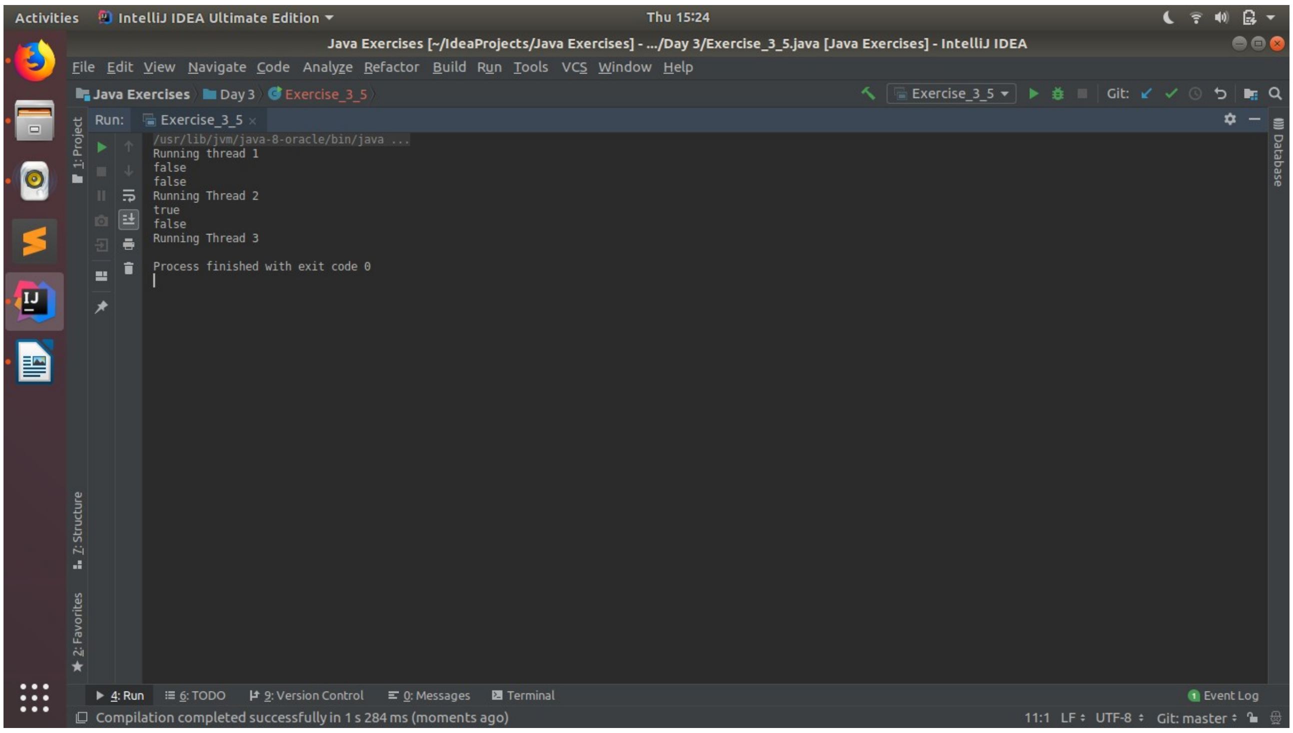
Compilation completed successfully in 1 s 157 ms (moments ago)

Q5. Use isShutDown() and isTerminated() with ExecutorService.
A5.


Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_5.java [Java Exercises] - IntelliJ IDEA

```java
class Exercise_3_5 {
    public static void main(String[] args) {
        ExecutorService e = Executors.newSingleThreadExecutor();
        try{
            e.submit(new Runnable() {
                @Override
                public void run() { System.out.println("Running thread 1"); }
            });
            e.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println(e.isShutdown());
                    System.out.println(e.isTerminated());
                    System.out.println("Running Thread 2");
                }
            });
            e.submit(new Runnable() {
                @Override
                public void run() {
                    try {

                        Thread.sleep( millis: 3500L);
                        System.out.println("Running Thread 3");

                    }
                    catch(InterruptedException e){
                        e.printStackTrace();
                    }
                }
            });
        }
        finally {
            e.shutdown();
        }
        System.out.println(e.isShutdown());
        System.out.println(e.isTerminated());
    }
```

Compilation completed successfully in 1 s 522 ms (3 minutes ago)

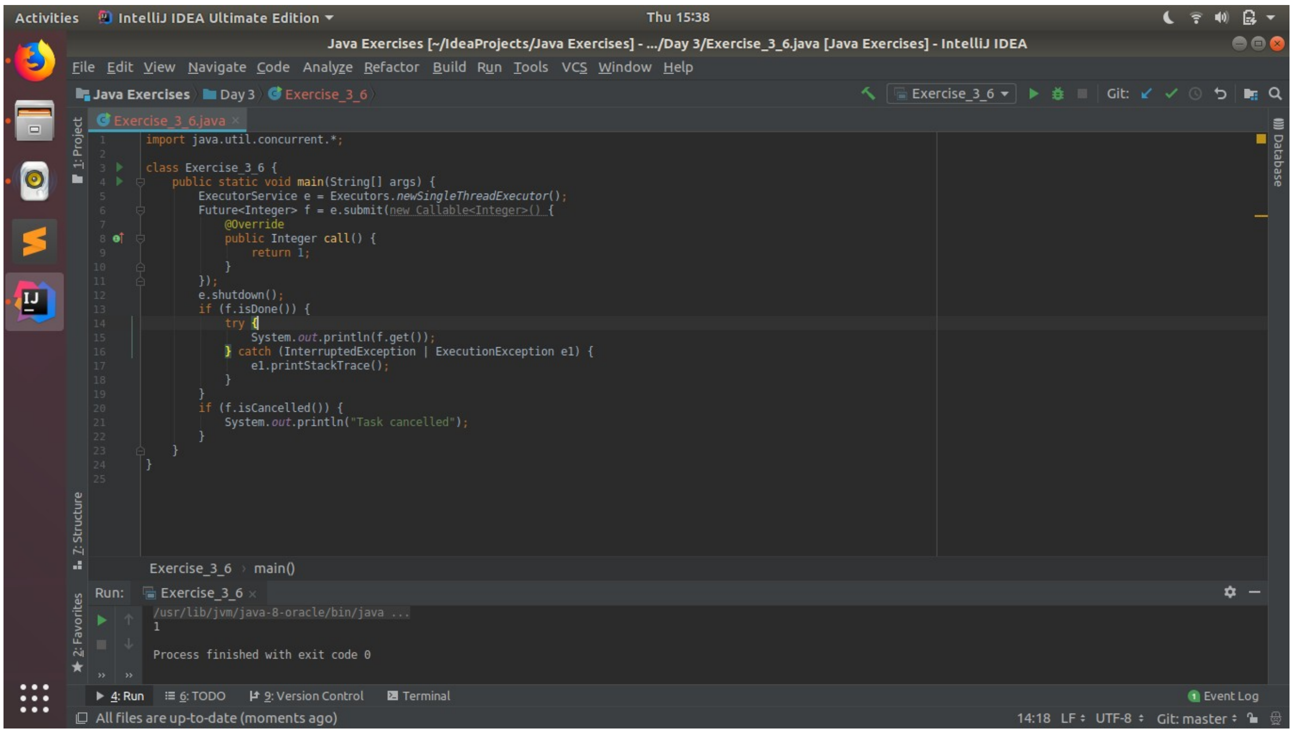


```
/usr/lib/jvm/java-8-oracle/bin/java ...
Running thread 1
false
false
Running Thread 2
true
false
Running Thread 3

Process finished with exit code 0
```

Compilation completed successfully in 1 s 284 ms (moments ago)

Q6. Return a Future from ExecutorService by using callable and use get(), isDone(), isCancelled() with the Future object to know the status of task submitted.

A6.

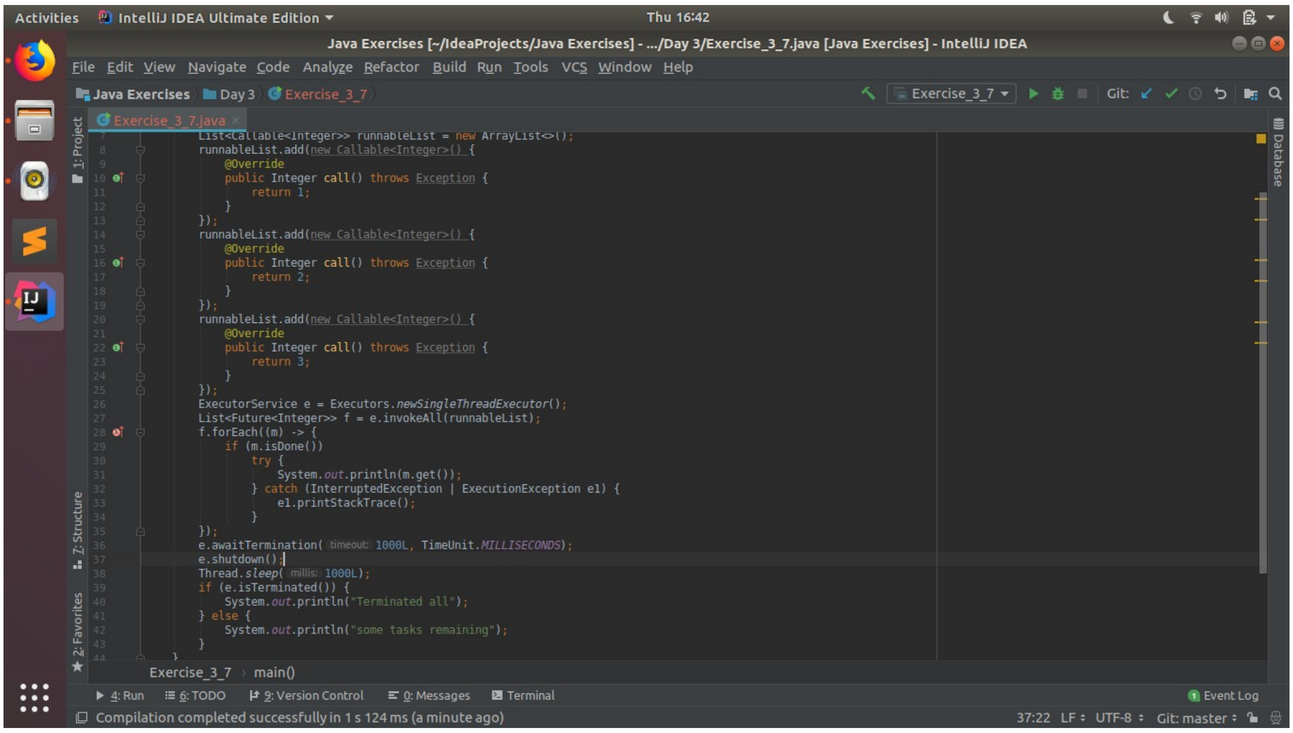


```java
import java.util.concurrent.*;

class Exercise_3_6 {
    public static void main(String[] args) {
        ExecutorService e = Executors.newSingleThreadExecutor();
        Future<Integer> f = e.submit(new Callable<Integer>() {
            @Override
            public Integer call() {
                return 1;
            }
        });
        e.shutdown();
        if (f.isDone()) {
            try {
                System.out.println(f.get());
            } catch (InterruptedException | ExecutionException e1) {
                e1.printStackTrace();
            }
        }
        if (f.isCancelled()) {
            System.out.println("Task cancelled");
        }
    }
}
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
1

Process finished with exit code 0
```

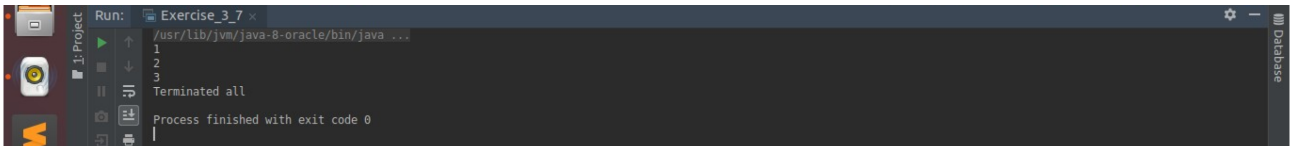
All files are up-to-date (moments ago)

Q7. Submit List of tasks to ExecutorService and wait for the completion of all the tasks.
A7.



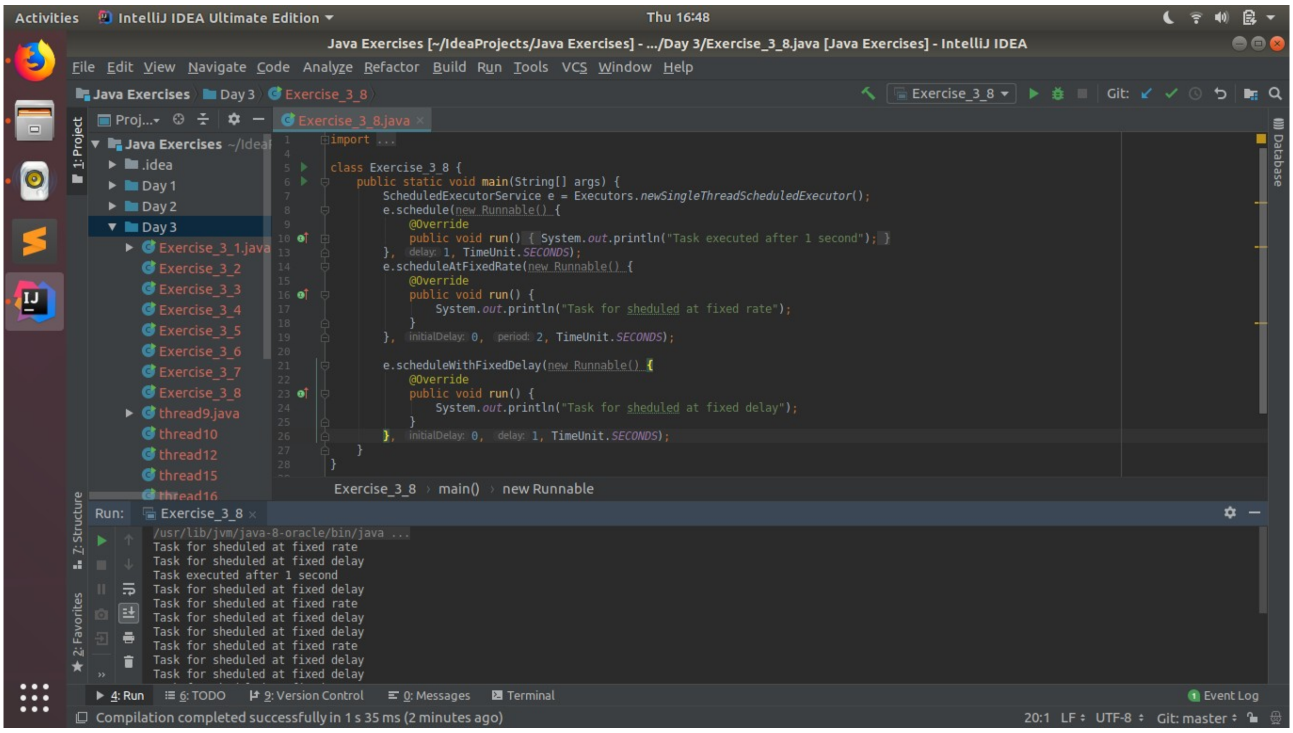
```java
        List<Callable<Integer>> runnableList = new ArrayList<>();
        runnableList.add(new Callable<Integer>() {
            @Override
            public Integer call() throws Exception {
                return 1;
            }
        });
        runnableList.add(new Callable<Integer>() {
            @Override
            public Integer call() throws Exception {
                return 2;
            }
        });
        runnableList.add(new Callable<Integer>() {
            @Override
            public Integer call() throws Exception {
                return 3;
            }
        });
        ExecutorService e = Executors.newSingleThreadExecutor();
        List<Future<Integer>> f = e.invokeAll(runnableList);
        f.forEach((m) -> {
            if (m.isDone())
                try {
                    System.out.println(m.get());
                } catch (InterruptedException | ExecutionException e1) {
                    e1.printStackTrace();
                }
        });
        e.awaitTermination( timeout: 1000L, TimeUnit.MILLISECONDS);
        e.shutdown();
        Thread.sleep( millis: 1000L);
        if (e.isTerminated()) {
            System.out.println("Terminated all");
        } else {
            System.out.println("some tasks remaining");
        }
    }
```

Compilation completed successfully in 1 s 124 ms (a minute ago)



```
/usr/lib/jvm/java-8-oracle/bin/java ...
1
2
3
Terminated all

Process finished with exit code 0
```

Q8. Schedule task using schedule(), scheduleAtFixedRate() and scheduleAtFixedDelay()
A8.



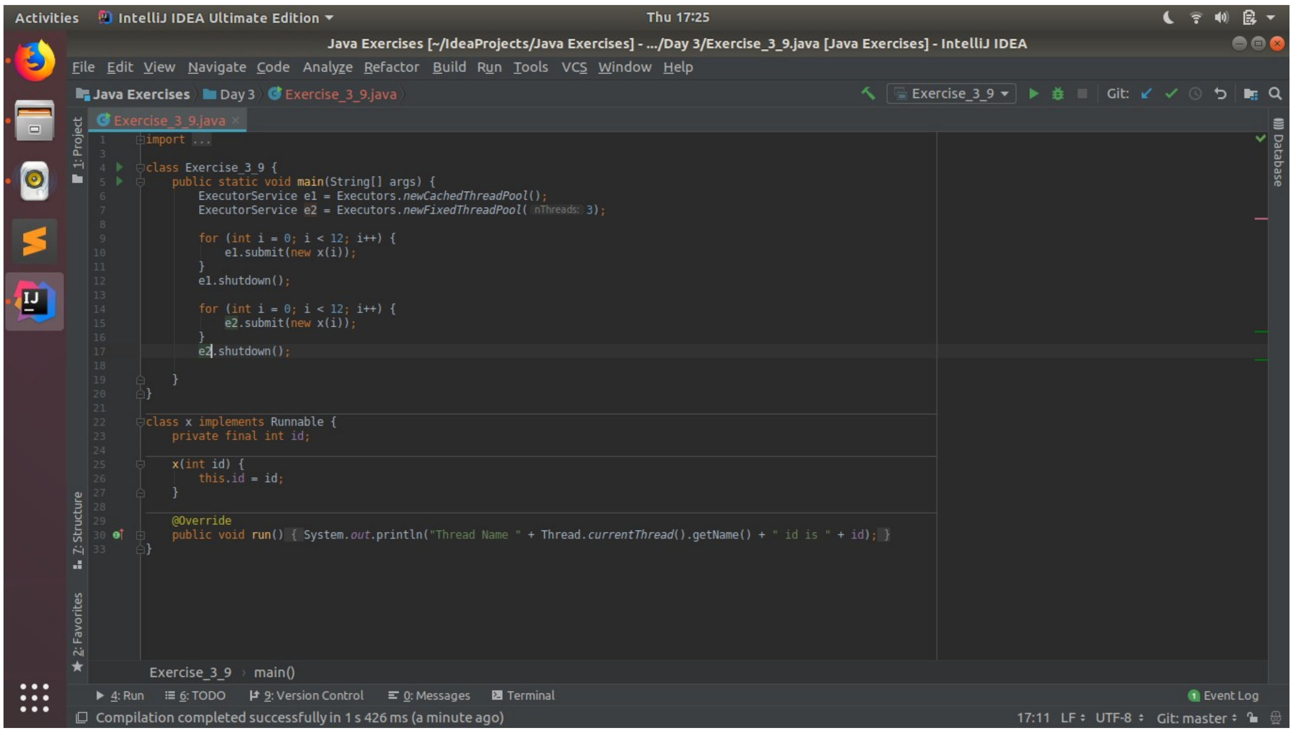
```java
import ...

class Exercise_3_8 {
    public static void main(String[] args) {
        ScheduledExecutorService e = Executors.newSingleThreadScheduledExecutor();
        e.schedule(new Runnable() {
            @Override
            public void run() { System.out.println("Task executed after 1 second"); }
        }, delay: 1, TimeUnit.SECONDS);
        e.scheduleAtFixedRate(new Runnable() {
            @Override
            public void run() {
                System.out.println("Task for sheduled at fixed rate");
            }
        }, initialDelay: 0, period: 2, TimeUnit.SECONDS);

        e.scheduleWithFixedDelay(new Runnable() {
            @Override
            public void run() {
                System.out.println("Task for sheduled at fixed delay");
            }
        }, initialDelay: 0, delay: 1, TimeUnit.SECONDS);
    }
}
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Task for sheduled at fixed rate
Task for sheduled at fixed delay
Task executed after 1 second
Task for sheduled at fixed delay
Task for sheduled at fixed rate
Task for sheduled at fixed delay
Task for sheduled at fixed delay
Task for sheduled at fixed rate
Task for sheduled at fixed delay
Task for sheduled at fixed delay
```

Compilation completed successfully in 1 s 35 ms (2 minutes ago)

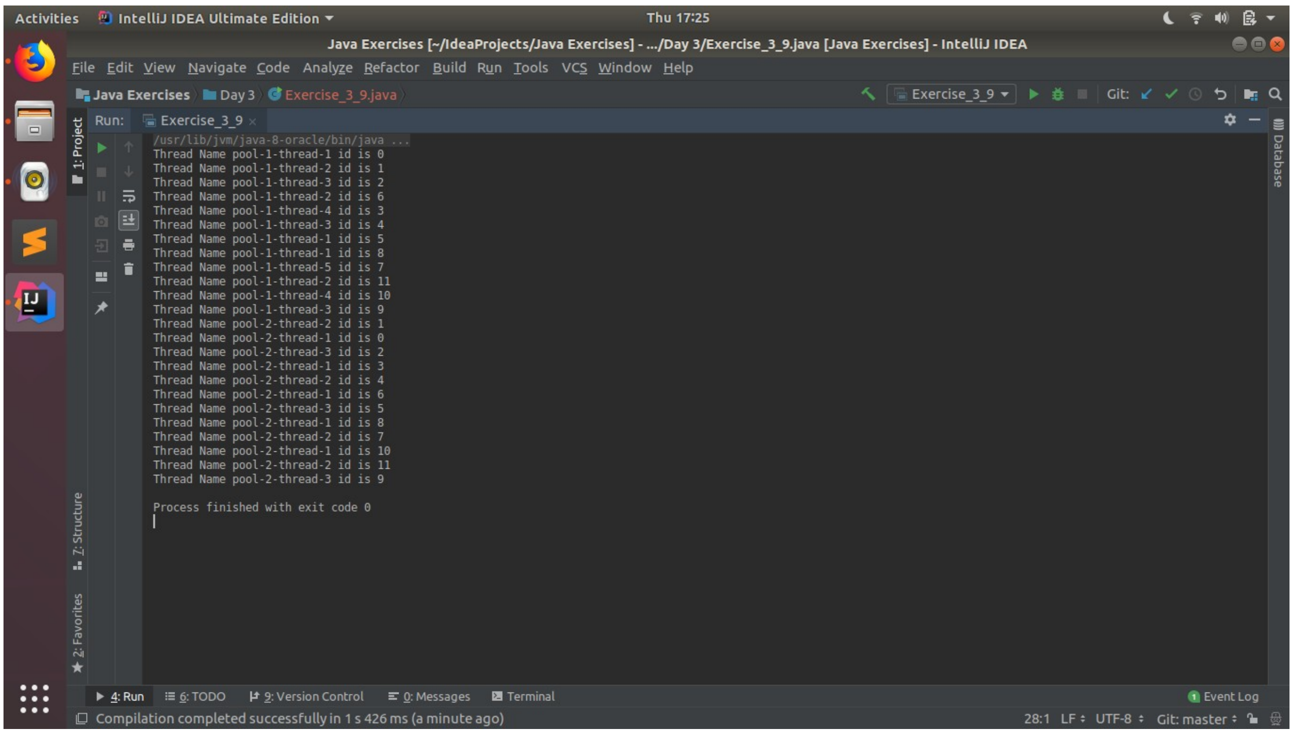
Q9. Increase concurrency with Thread pools using newCachedThreadPool() and newFixedThreadPool().

A9.



```java
import ...

class Exercise_3_9 {
    public static void main(String[] args) {
        ExecutorService e1 = Executors.newCachedThreadPool();
        ExecutorService e2 = Executors.newFixedThreadPool( nThreads: 3);

        for (int i = 0; i < 12; i++) {
            e1.submit(new x(i));
        }
        e1.shutdown();

        for (int i = 0; i < 12; i++) {
            e2.submit(new x(i));
        }
        e2.shutdown();

    }
}

class x implements Runnable {
    private final int id;

    x(int id) {
        this.id = id;
    }

    @Override
    public void run() { System.out.println("Thread Name " + Thread.currentThread().getName() + " id is " + id); }
}
```

Compilation completed successfully in 1 s 426 ms (a minute ago)

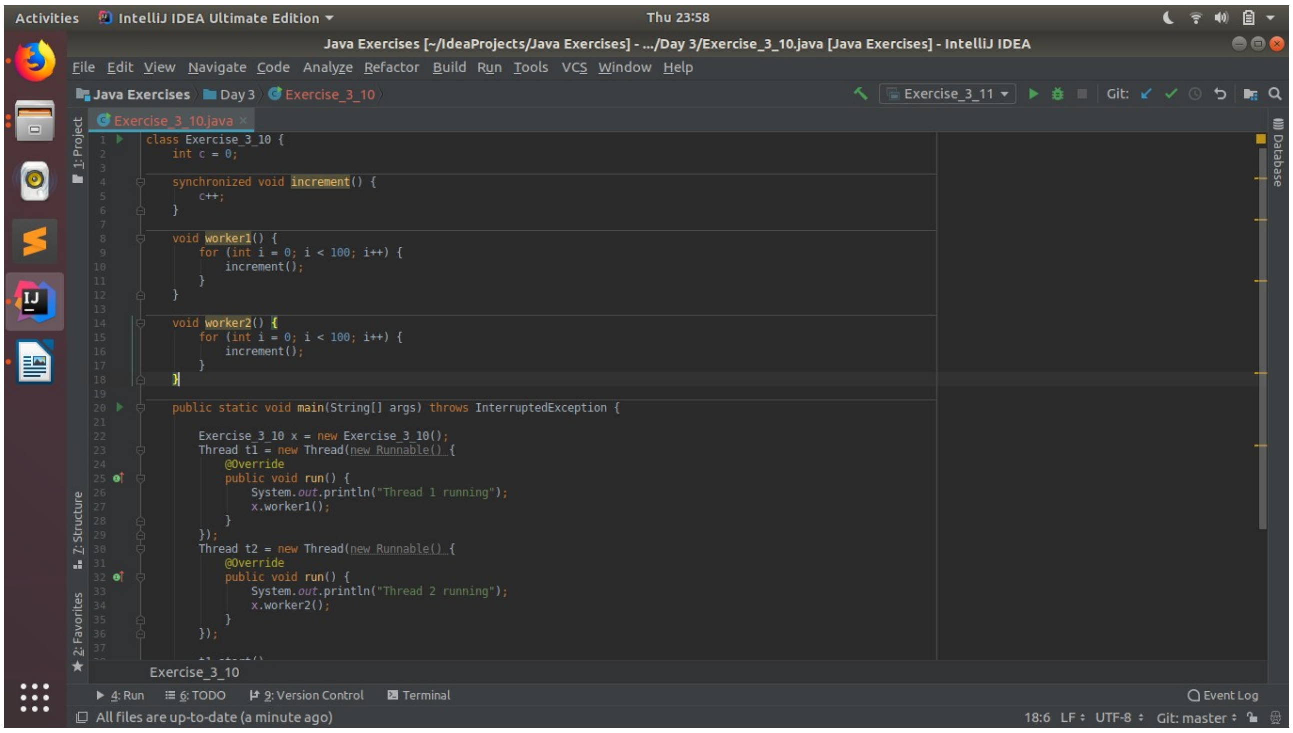


```
/usr/lib/jvm/java-8-oracle/bin/java ...
Thread Name pool-1-thread-1 id is 0
Thread Name pool-1-thread-2 id is 1
Thread Name pool-1-thread-3 id is 2
Thread Name pool-1-thread-2 id is 6
Thread Name pool-1-thread-4 id is 3
Thread Name pool-1-thread-3 id is 4
Thread Name pool-1-thread-1 id is 5
Thread Name pool-1-thread-1 id is 8
Thread Name pool-1-thread-5 id is 7
Thread Name pool-1-thread-2 id is 11
Thread Name pool-1-thread-4 id is 10
Thread Name pool-1-thread-3 id is 9
Thread Name pool-2-thread-2 id is 1
Thread Name pool-2-thread-1 id is 0
Thread Name pool-2-thread-3 id is 2
Thread Name pool-2-thread-1 id is 3
Thread Name pool-2-thread-2 id is 4
Thread Name pool-2-thread-1 id is 6
Thread Name pool-2-thread-3 id is 5
Thread Name pool-2-thread-1 id is 8
Thread Name pool-2-thread-2 id is 7
Thread Name pool-2-thread-1 id is 10
Thread Name pool-2-thread-2 id is 11
Thread Name pool-2-thread-3 id is 9

Process finished with exit code 0
```

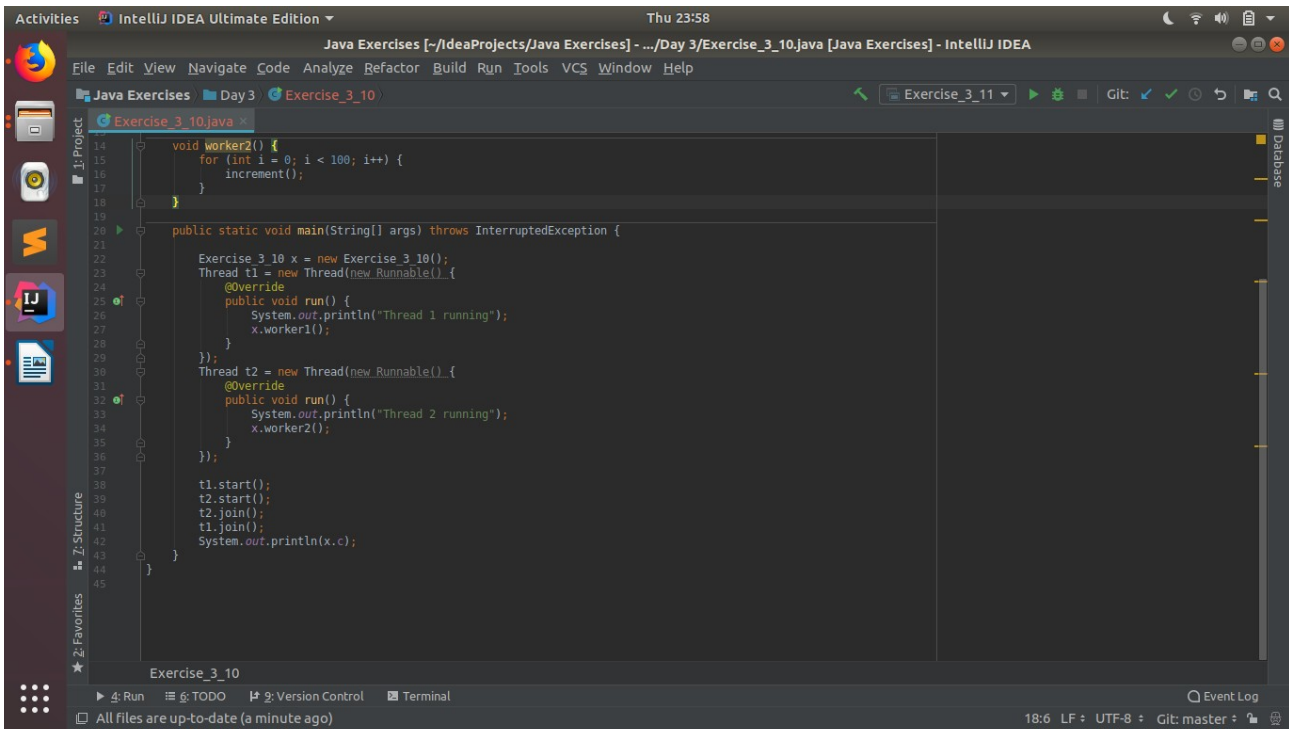
Compilation completed successfully in 1 s 426 ms (a minute ago)

Q10. Use Synchronize method to enable synchronization between multiple threads trying to access method at same time.

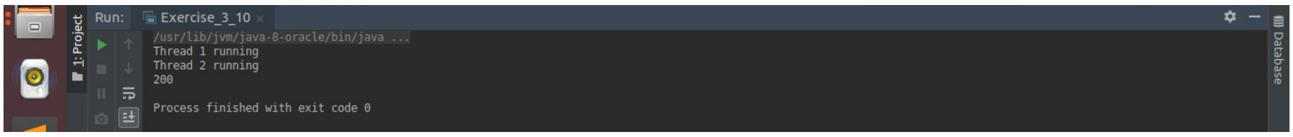
A10.


Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_10.java [Java Exercises] - IntelliJ IDEA
Exercise_3_10.java
All files are up-to-date (a minute ago)

```java
class Exercise_3_10 {
    int c = 0;

    synchronized void increment() {
        c++;
    }

    void worker1() {
        for (int i = 0; i < 100; i++) {
            increment();
        }
    }

    void worker2() {
        for (int i = 0; i < 100; i++) {
            increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {

        Exercise_3_10 x = new Exercise_3_10();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1 running");
                x.worker1();
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2 running");
                x.worker2();
            }
        });
```

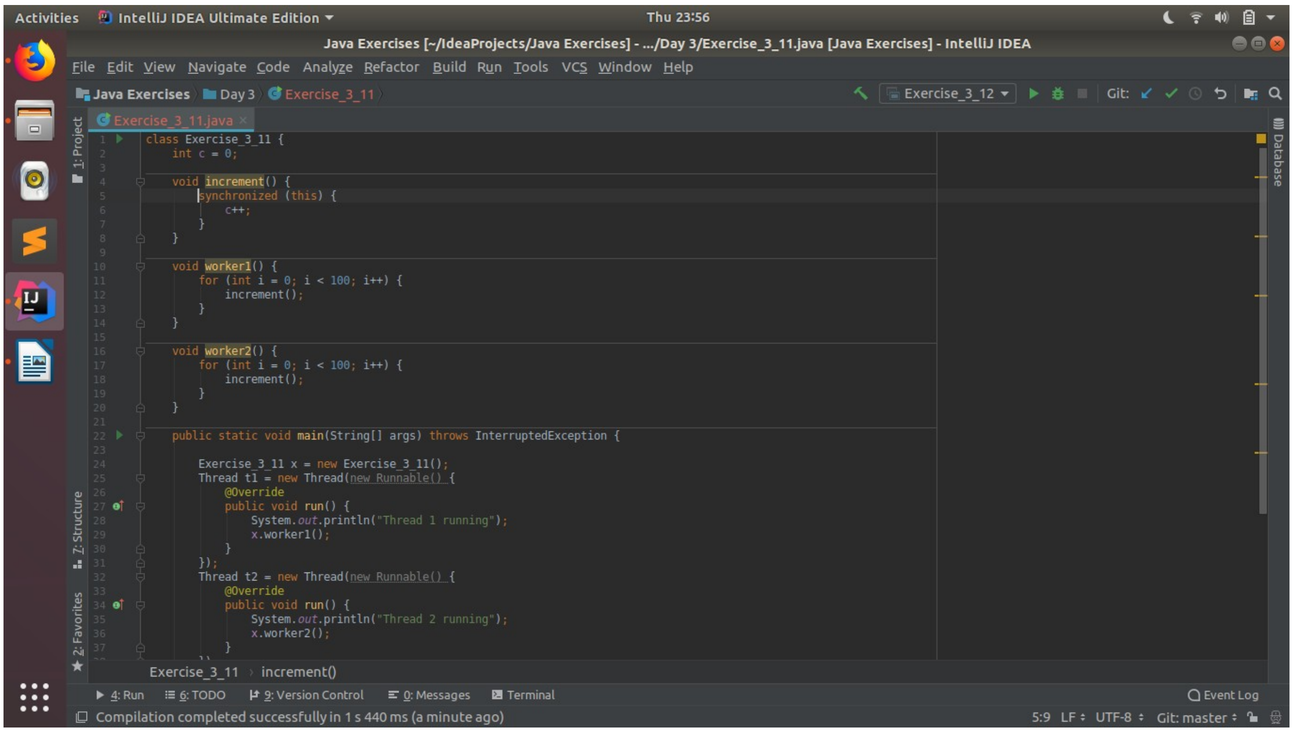

Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_10.java [Java Exercises] - IntelliJ IDEA
Exercise_3_10.java
All files are up-to-date (a minute ago)

```java
    void worker2() {
        for (int i = 0; i < 100; i++) {
            increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {

        Exercise_3_10 x = new Exercise_3_10();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1 running");
                x.worker1();
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2 running");
                x.worker2();
            }
        });

        t1.start();
        t2.start();
        t2.join();
        t1.join();
        System.out.println(x.c);
    }
}
```



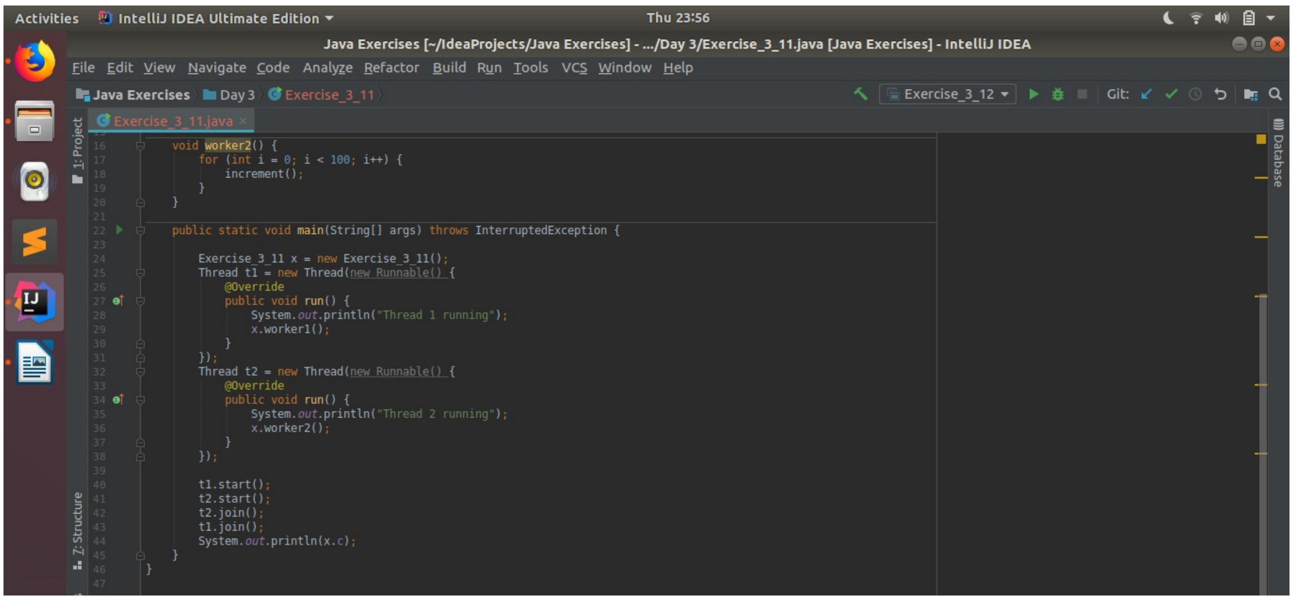
```
/usr/lib/jvm/java-8-oracle/bin/java ...
Thread 1 running
Thread 2 running
200

Process finished with exit code 0
```

Q11. Use Synchronize block to enable synchronization between multiple threads trying to access method at same time.
A11.

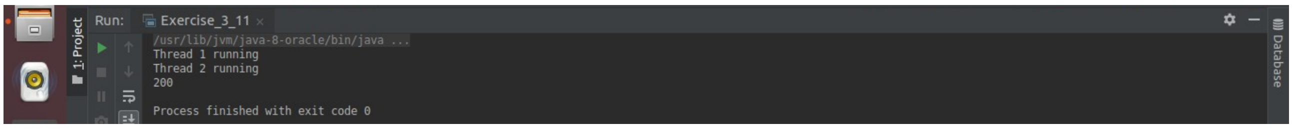


```java
class Exercise_3_11 {
    int c = 0;

    void increment() {
        synchronized (this) {
            c++;
        }
    }

    void worker1() {
        for (int i = 0; i < 100; i++) {
            increment();
        }
    }

    void worker2() {
        for (int i = 0; i < 100; i++) {
            increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {

        Exercise_3_11 x = new Exercise_3_11();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1 running");
                x.worker1();
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2 running");
                x.worker2();
            }
```

Compilation completed successfully in 1 s 440 ms (a minute ago)

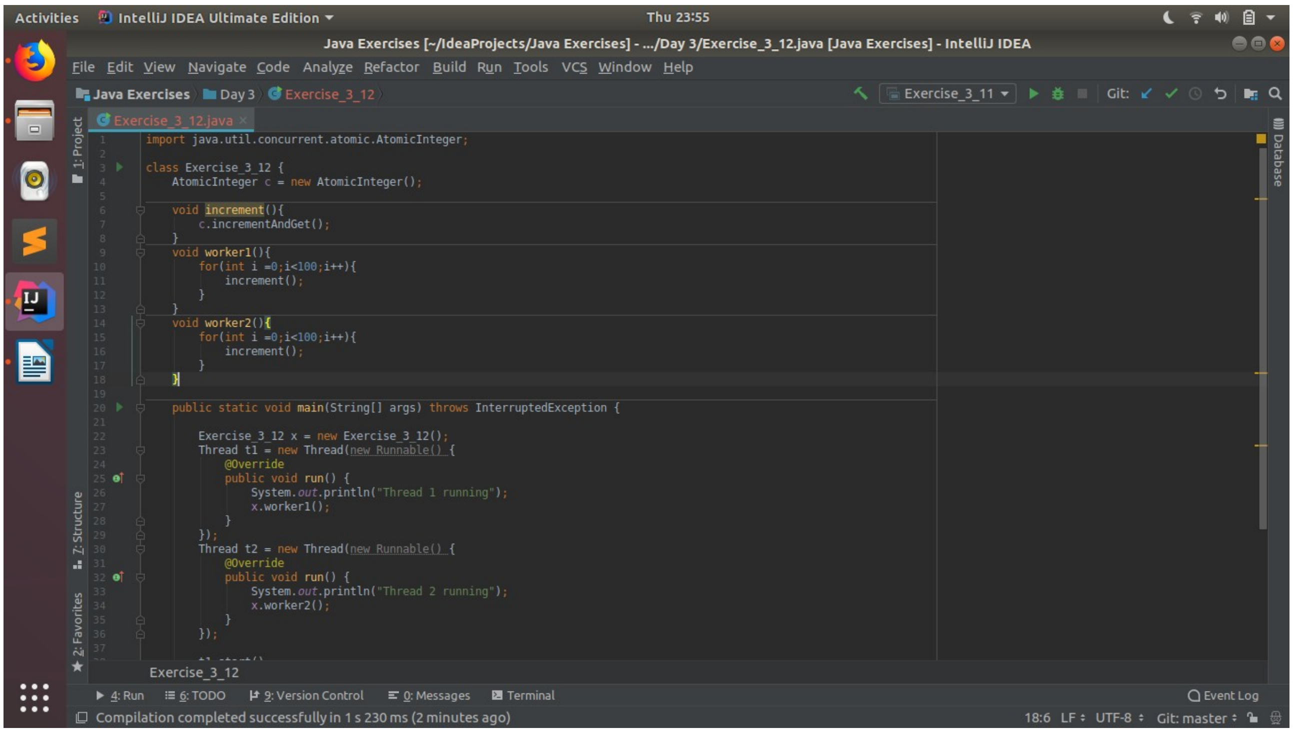


```java
    void worker2() {
        for (int i = 0; i < 100; i++) {
            increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {

        Exercise_3_11 x = new Exercise_3_11();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1 running");
                x.worker1();
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2 running");
                x.worker2();
            }
        });

        t1.start();
        t2.start();
        t2.join();
        t1.join();
        System.out.println(x.c);
    }
}
```



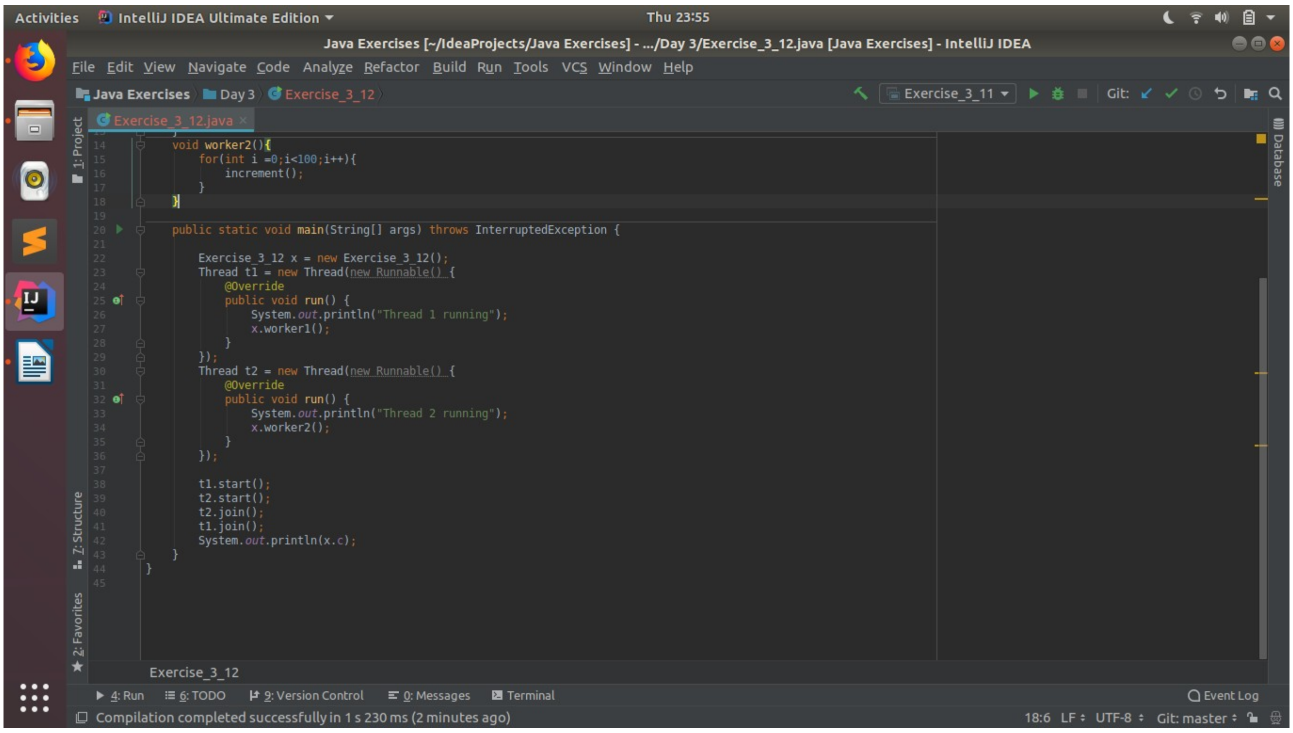
```
/usr/lib/jvm/java-8-oracle/bin/java ...
Thread 1 running
Thread 2 running
200

Process finished with exit code 0
```

Q12. Use Atomic Classes instead of Synchronize method and blocks.
A12.



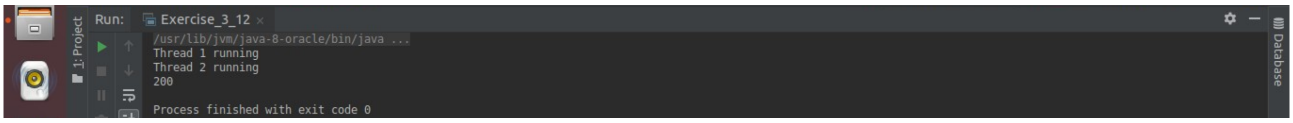
```java
import java.util.concurrent.atomic.AtomicInteger;

class Exercise_3_12 {
    AtomicInteger c = new AtomicInteger();

    void increment(){
        c.incrementAndGet();
    }
    void worker1(){
        for(int i =0;i<100;i++){
            increment();
        }
    }
    void worker2(){
        for(int i =0;i<100;i++){
            increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {

        Exercise_3_12 x = new Exercise_3_12();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1 running");
                x.worker1();
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2 running");
                x.worker2();
            }
        });

```

Compilation completed successfully in 1 s 230 ms (2 minutes ago)



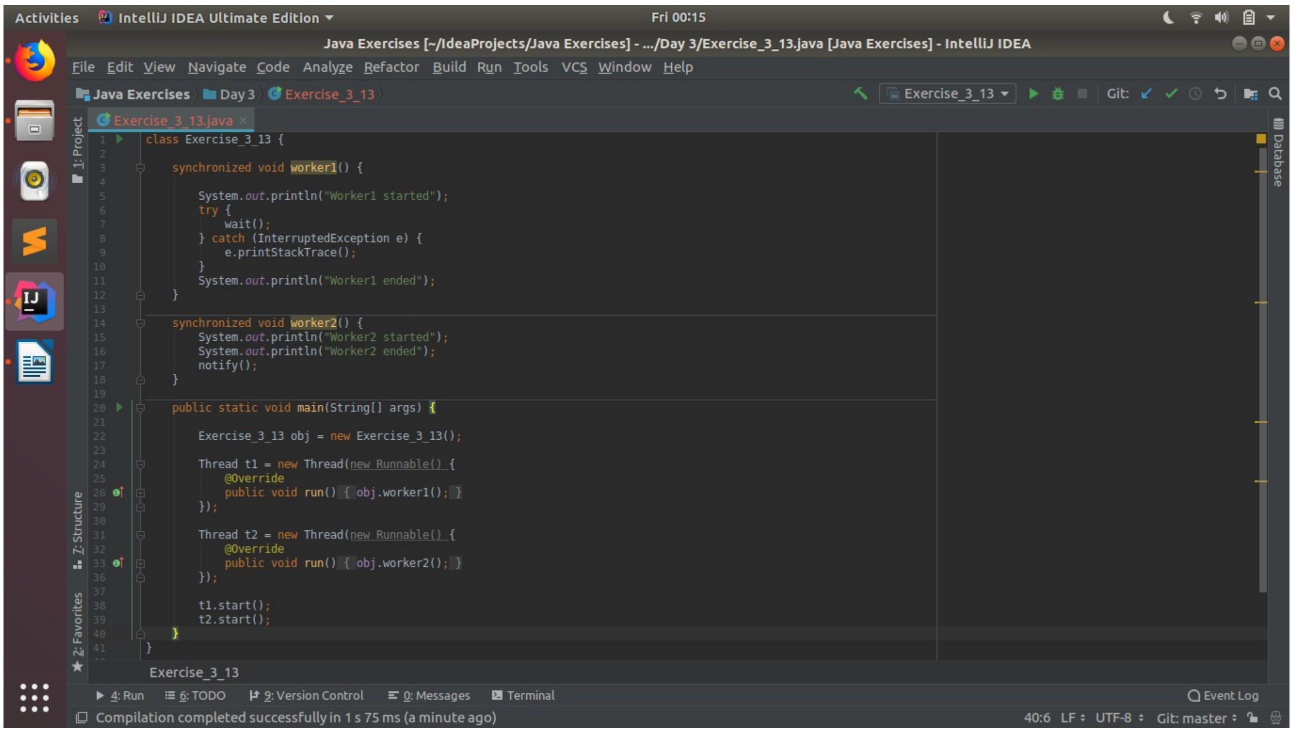
```java
    void worker2(){
        for(int i =0;i<100;i++){
            increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {

        Exercise_3_12 x = new Exercise_3_12();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1 running");
                x.worker1();
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2 running");
                x.worker2();
            }
        });

        t1.start();
        t2.start();
        t2.join();
        t1.join();
        System.out.println(x.c);
    }
}
```

Compilation completed successfully in 1 s 230 ms (2 minutes ago)



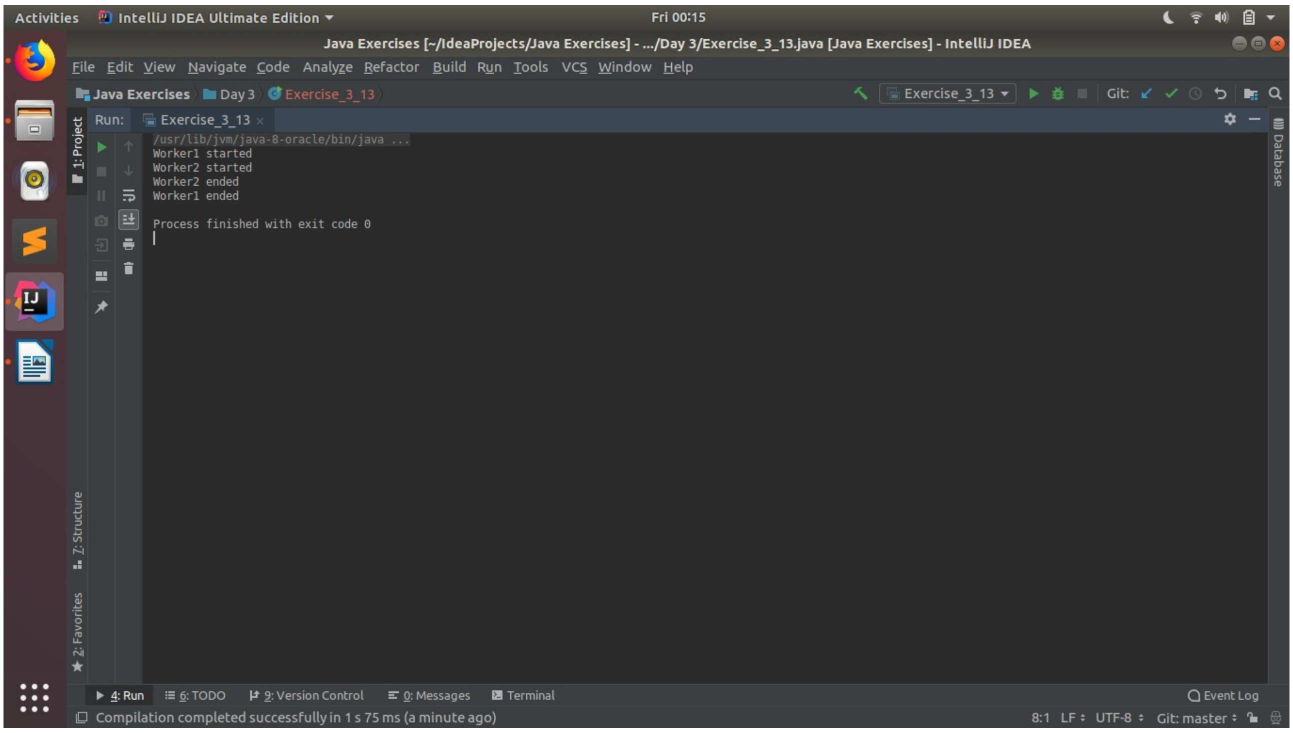
```
/usr/lib/jvm/java-8-oracle/bin/java ...
Thread 1 running
Thread 2 running
200

Process finished with exit code 0
```

Q13. Coordinate 2 threads using wait() and notify().
A13.


Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_13.java [Java Exercises] - IntelliJ IDEA

```java
class Exercise_3_13 {

    synchronized void worker1() {

        System.out.println("Worker1 started");
        try {
            wait();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Worker1 ended");
    }

    synchronized void worker2() {
        System.out.println("Worker2 started");
        System.out.println("Worker2 ended");
        notify();
    }

    public static void main(String[] args) {

        Exercise_3_13 obj = new Exercise_3_13();

        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() { obj.worker1(); }
        });

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() { obj.worker2(); }
        });

        t1.start();
        t2.start();
    }
}
```

Compilation completed successfully in 1 s 75 ms (a minute ago)

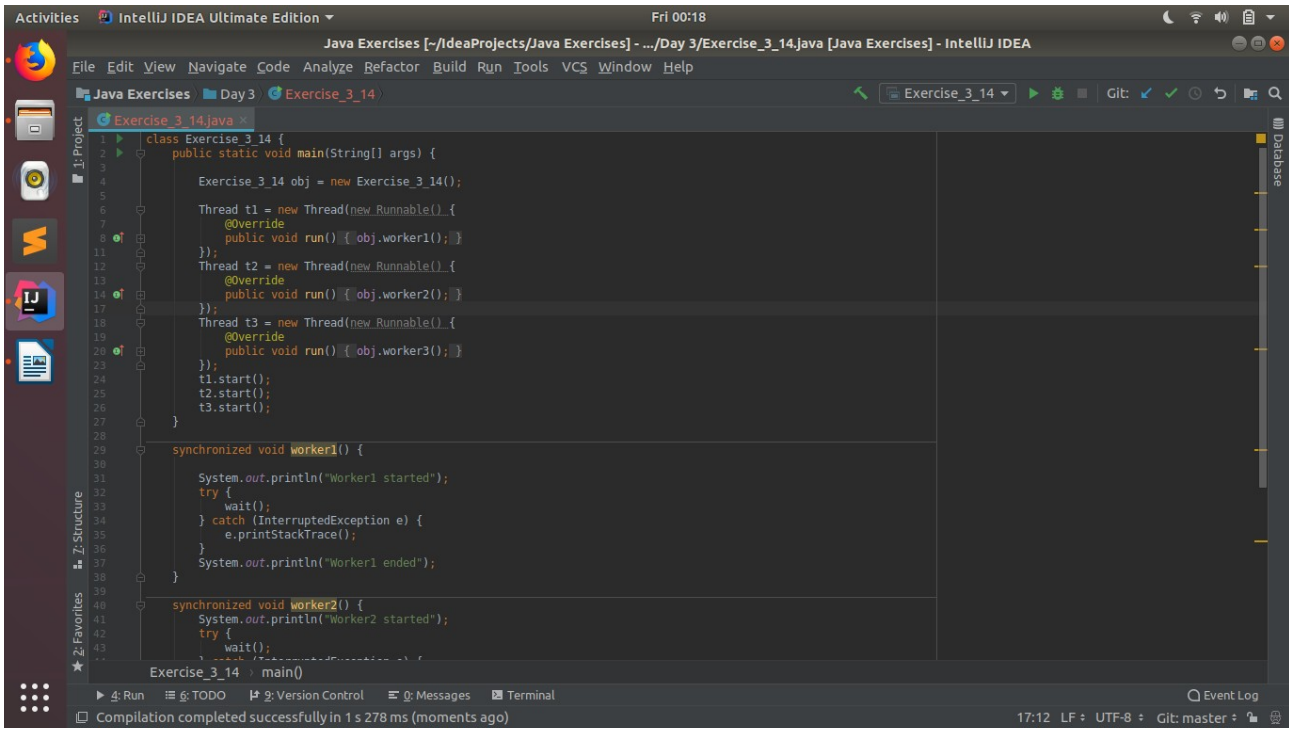

Run: Exercise_3_13

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Worker1 started
Worker2 started
Worker2 ended
Worker1 ended

Process finished with exit code 0
```

Compilation completed successfully in 1 s 75 ms (a minute ago)

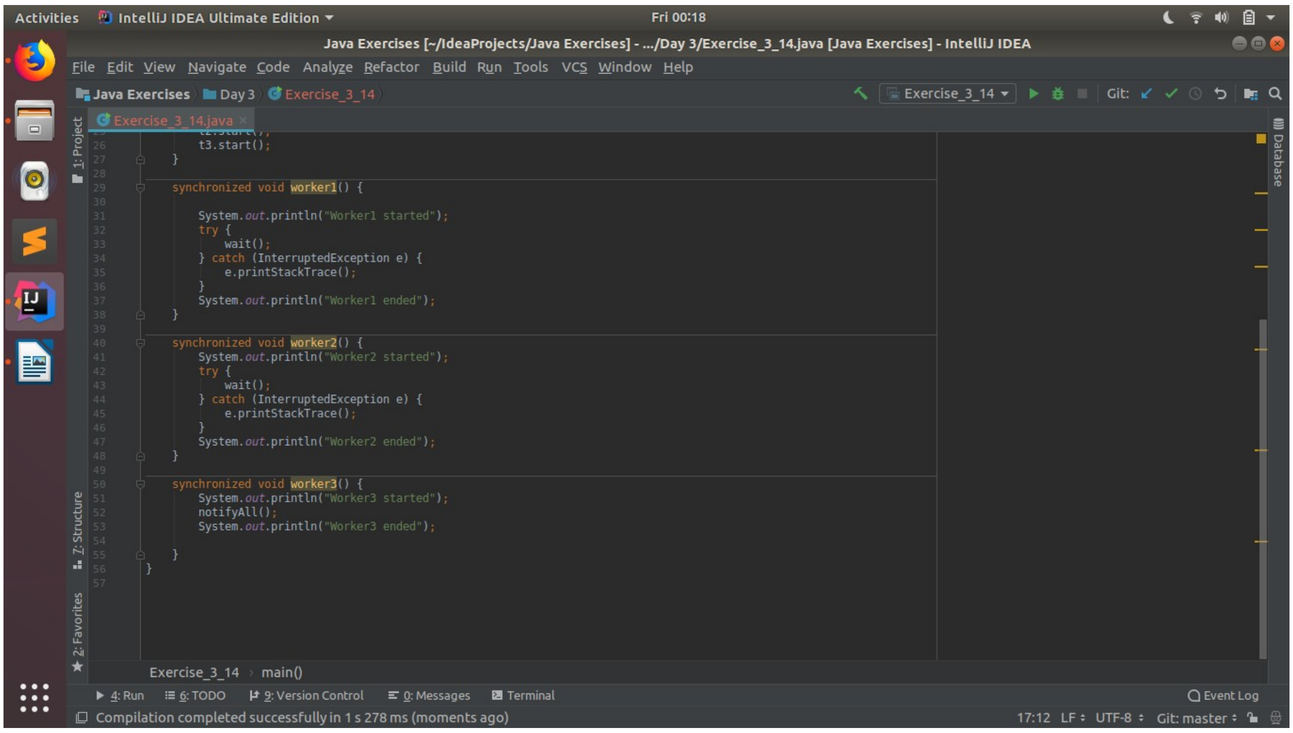
Q14 Coordinate mulitple threads using wait() and notifyAll()

A14.


Activities
IntelliJ IDEA Ultimate Edition
Fri 00:18
Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_14.java [Java Exercises] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Java Exercises
Day 3
Exercise_3_14
Exercise_3_14.java
Exercise_3_14 main()
4: Run
6: TODO
9: Version Control
0: Messages
Terminal
Event Log
17:12 LF UTF-8 Git: master

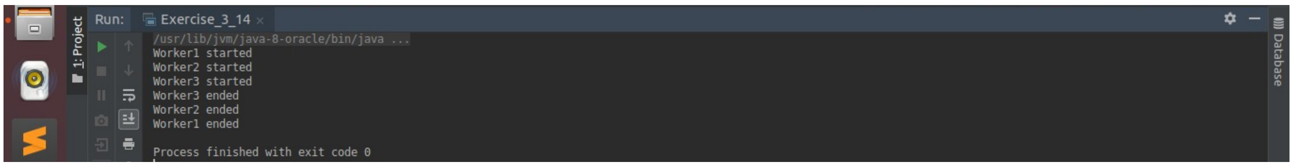
```java
class Exercise_3_14 {
    public static void main(String[] args) {

        Exercise_3_14 obj = new Exercise_3_14();

        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() { obj.worker1(); }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() { obj.worker2(); }
        });
        Thread t3 = new Thread(new Runnable() {
            @Override
            public void run() { obj.worker3(); }
        });
        t1.start();
        t2.start();
        t3.start();
    }

    synchronized void worker1() {

        System.out.println("Worker1 started");
        try {
            wait();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Worker1 ended");
    }

    synchronized void worker2() {
        System.out.println("Worker2 started");
        try {
            wait();
```

Compilation completed successfully in 1 s 278 ms (moments ago)


Activities
IntelliJ IDEA Ultimate Edition
Fri 00:18
Java Exercises [~/IdeaProjects/Java Exercises] - .../Day 3/Exercise_3_14.java [Java Exercises] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Java Exercises
Day 3
Exercise_3_14
Exercise_3_14.java
Exercise_3_14 main()
4: Run
6: TODO
9: Version Control
0: Messages
Terminal
Event Log
17:12 LF UTF-8 Git: master

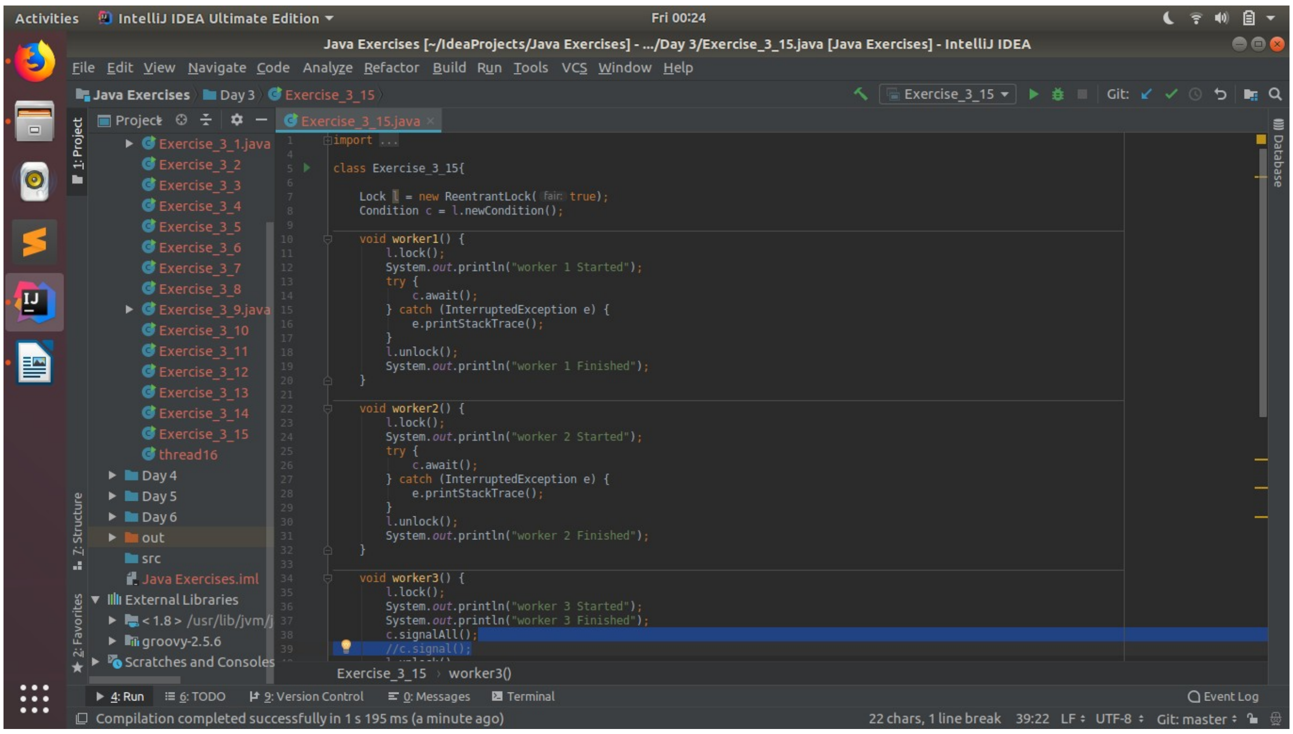
```java
        t3.start();
    }

    synchronized void worker1() {

        System.out.println("Worker1 started");
        try {
            wait();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Worker1 ended");
    }

    synchronized void worker2() {
        System.out.println("Worker2 started");
        try {
            wait();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Worker2 ended");
    }

    synchronized void worker3() {
        System.out.println("Worker3 started");
        notifyAll();
        System.out.println("Worker3 ended");

    }
}
```

Compilation completed successfully in 1 s 278 ms (moments ago)



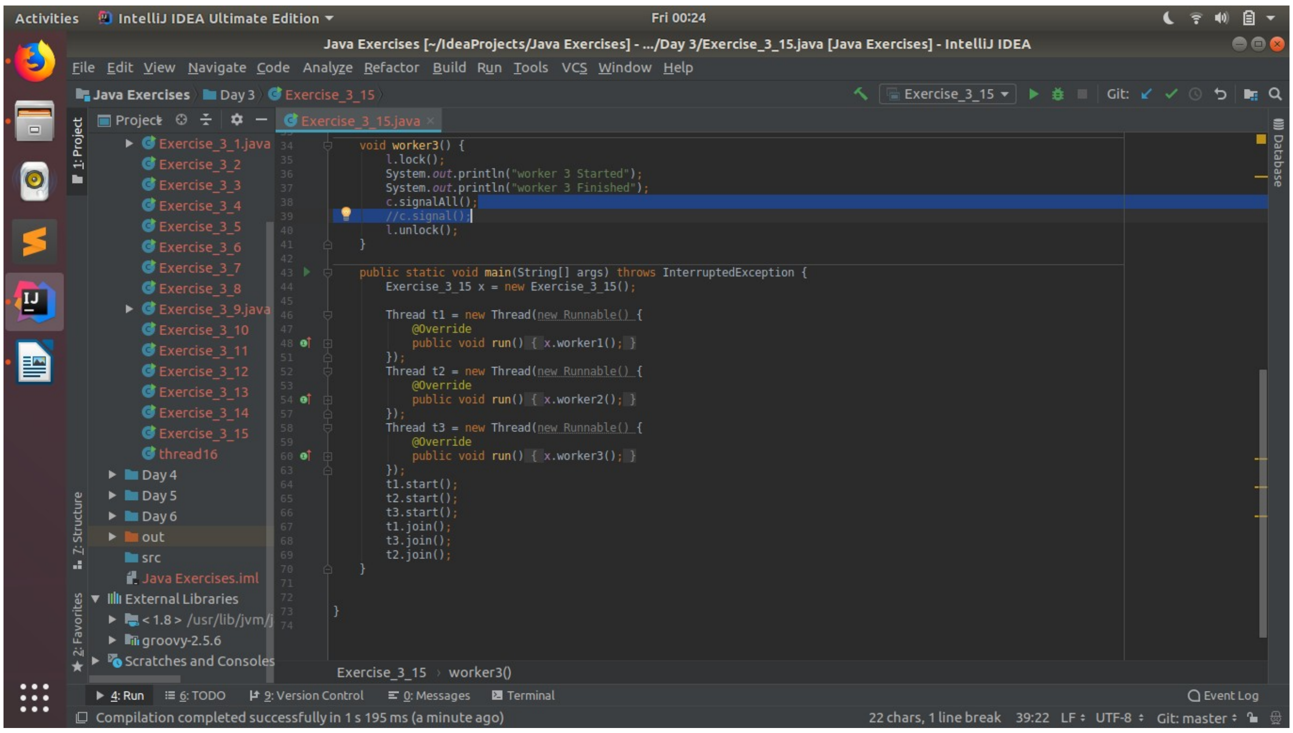
```
/usr/lib/jvm/java-8-oracle/bin/java ...
Worker1 started
Worker2 started
Worker3 started
Worker3 ended
Worker2 ended
Worker1 ended

Process finished with exit code 0
```

Q15. Use Reentract lock for coordinating 2 threads with signal(), signalAll() and wait().
A15.



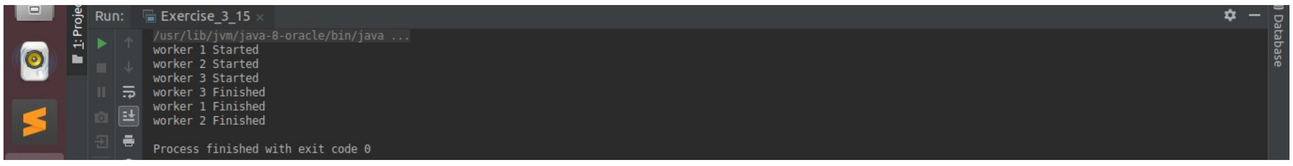
```java
import ...

class Exercise_3_15{

    Lock l = new ReentrantLock( fair: true);
    Condition c = l.newCondition();

    void worker1() {
        l.lock();
        System.out.println("worker 1 Started");
        try {
            c.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        l.unlock();
        System.out.println("worker 1 Finished");
    }

    void worker2() {
        l.lock();
        System.out.println("worker 2 Started");
        try {
            c.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        l.unlock();
        System.out.println("worker 2 Finished");
    }

    void worker3() {
        l.lock();
        System.out.println("worker 3 Started");
        System.out.println("worker 3 Finished");
        c.signalAll();
        //c.signal();
```

Compilation completed successfully in 1 s 195 ms (a minute ago)



```java
    void worker3() {
        l.lock();
        System.out.println("worker 3 Started");
        System.out.println("worker 3 Finished");
        c.signalAll();
        //c.signal();
        l.unlock();
    }

    public static void main(String[] args) throws InterruptedException {
        Exercise_3_15 x = new Exercise_3_15();

        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() { x.worker1(); }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() { x.worker2(); }
        });
        Thread t3 = new Thread(new Runnable() {
            @Override
            public void run() { x.worker3(); }
        });
        t1.start();
        t2.start();
        t3.start();
        t1.join();
        t3.join();
        t2.join();
    }


}
```

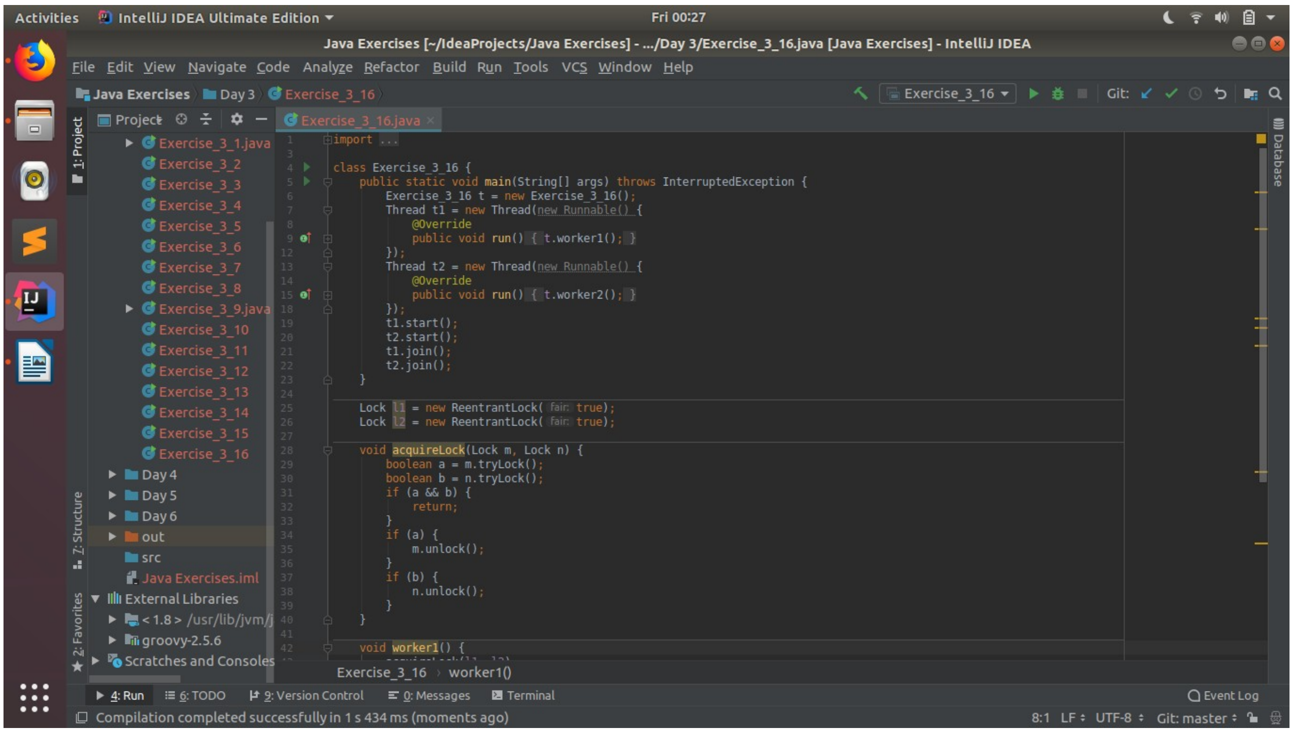
Compilation completed successfully in 1 s 195 ms (a minute ago)



```
/usr/lib/jvm/java-8-oracle/bin/java ...
worker 1 Started
worker 2 Started
worker 3 Started
worker 3 Finished
worker 1 Finished
worker 2 Finished

Process finished with exit code 0
```

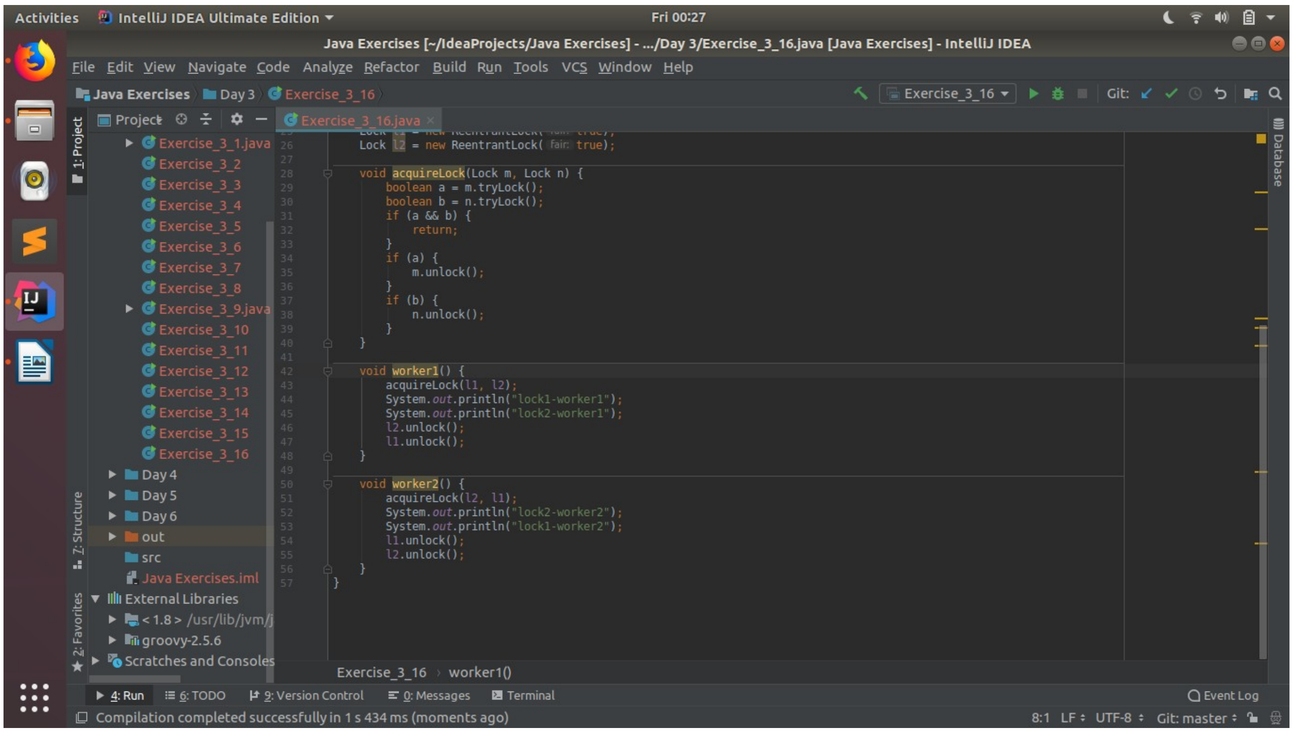
Q16. Create a deadlock and Resolve it using tryLock().

A16.



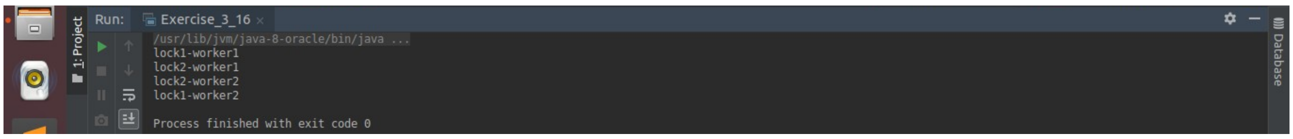
```java
import ...

class Exercise_3_16 {
    public static void main(String[] args) throws InterruptedException {
        Exercise_3_16 t = new Exercise_3_16();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() { t.worker1(); }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() { t.worker2(); }
        });
        t1.start();
        t2.start();
        t1.join();
        t2.join();
    }

    Lock l1 = new ReentrantLock( fair: true);
    Lock l2 = new ReentrantLock( fair: true);

    void acquireLock(Lock m, Lock n) {
        boolean a = m.tryLock();
        boolean b = n.tryLock();
        if (a && b) {
            return;
        }
        if (a) {
            m.unlock();
        }
        if (b) {
            n.unlock();
        }
    }

    void worker1() {
```

Compilation completed successfully in 1 s 434 ms (moments ago)



```java
    Lock l2 = new ReentrantLock( fair: true);

    void acquireLock(Lock m, Lock n) {
        boolean a = m.tryLock();
        boolean b = n.tryLock();
        if (a && b) {
            return;
        }
        if (a) {
            m.unlock();
        }
        if (b) {
            n.unlock();
        }
    }

    void worker1() {
        acquireLock(l1, l2);
        System.out.println("lock1-worker1");
        System.out.println("lock2-worker1");
        l2.unlock();
        l1.unlock();
    }

    void worker2() {
        acquireLock(l2, l1);
        System.out.println("lock2-worker2");
        System.out.println("lock1-worker2");
        l1.unlock();
        l2.unlock();
    }
}
```

Compilation completed successfully in 1 s 434 ms (moments ago)



```
/usr/lib/jvm/java-8-oracle/bin/java ...
lock1-worker1
lock2-worker1
lock2-worker2
lock1-worker2

Process finished with exit code 0
```