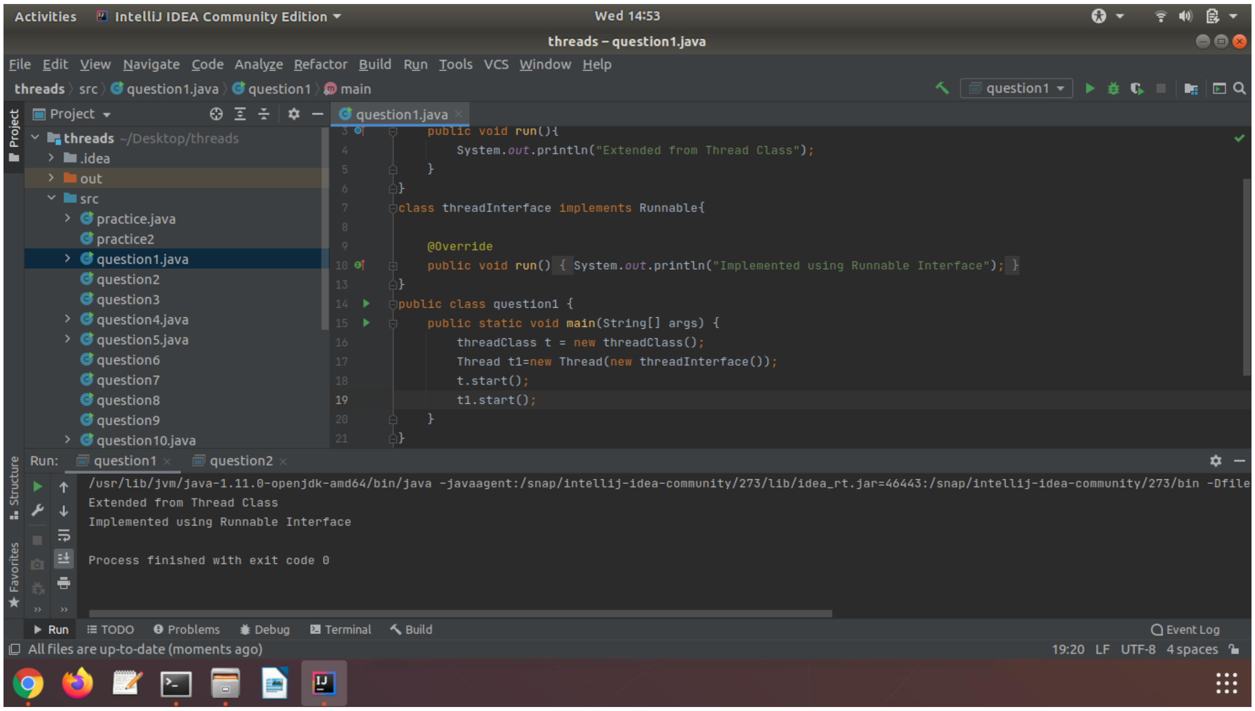
1. Create and Run a Thread using Runnable Interface and Thread class.



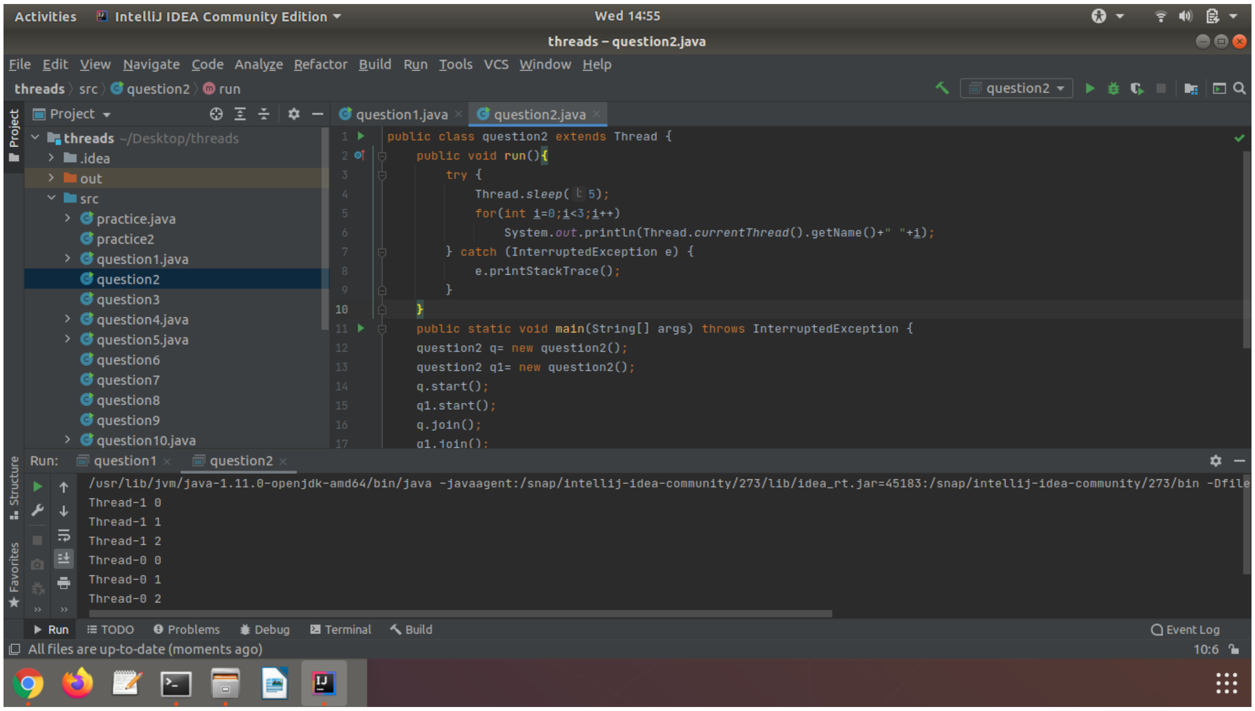
```java
    public void run(){
        System.out.println("Extended from Thread Class");
    }
}
class threadInterface implements Runnable{

    @Override
    public void run() { System.out.println("Implemented using Runnable Interface"); }
}
public class question1 {
    public static void main(String[] args) {
        threadClass t = new threadClass();
        Thread t1=new Thread(new threadInterface());
        t.start();
        t1.start();
    }
}
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=46443:/snap/intellij-idea-community/273/bin -Dfile
Extended from Thread Class
Implemented using Runnable Interface

Process finished with exit code 0
```

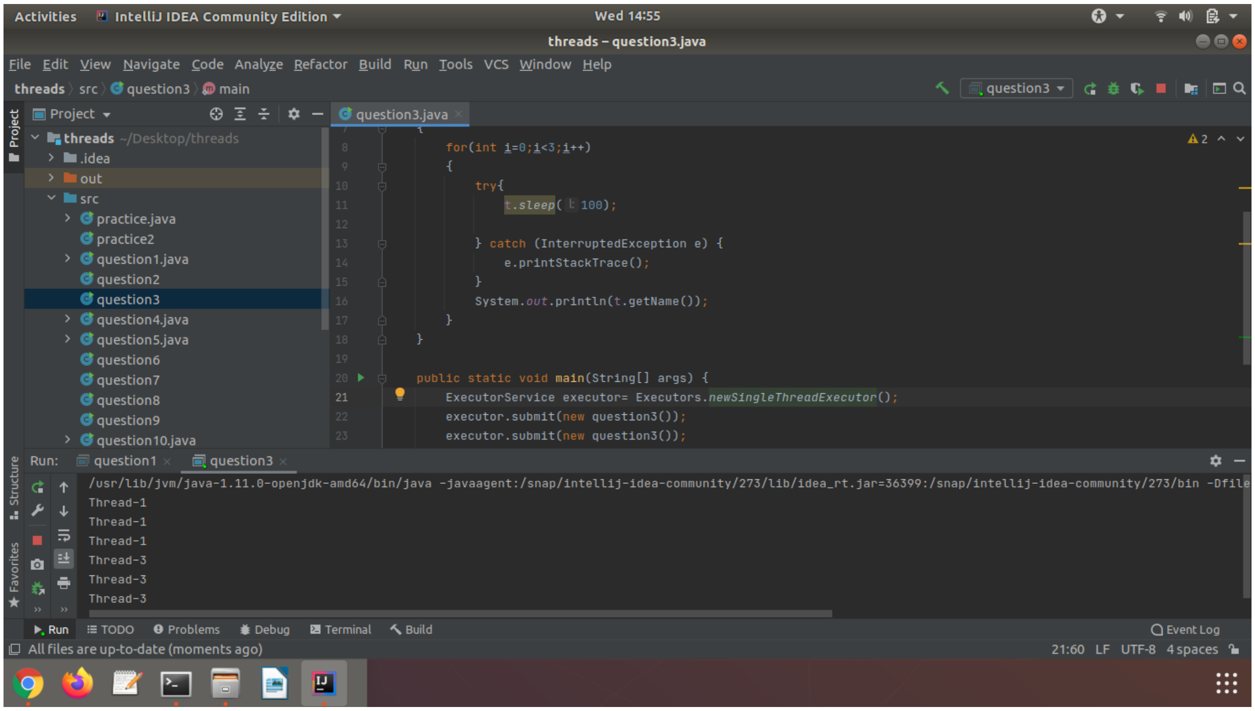
2. Use sleep and join methods with thread.



```java
public class question2 extends Thread {
    public void run(){
        try {
            Thread.sleep( l: 5);
            for(int i=0;i<3;i++)
                System.out.println(Thread.currentThread().getName()+" "+i);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) throws InterruptedException {
    question2 q= new question2();
    question2 q1= new question2();
    q.start();
    q1.start();
    q.join();
    q1.join();
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=45183:/snap/intellij-idea-community/273/bin -Dfile
Thread-1 0
Thread-1 1
Thread-1 2
Thread-0 0
Thread-0 1
Thread-0 2
```

3. Use a singleThreadExecutor to submit multiple threads.



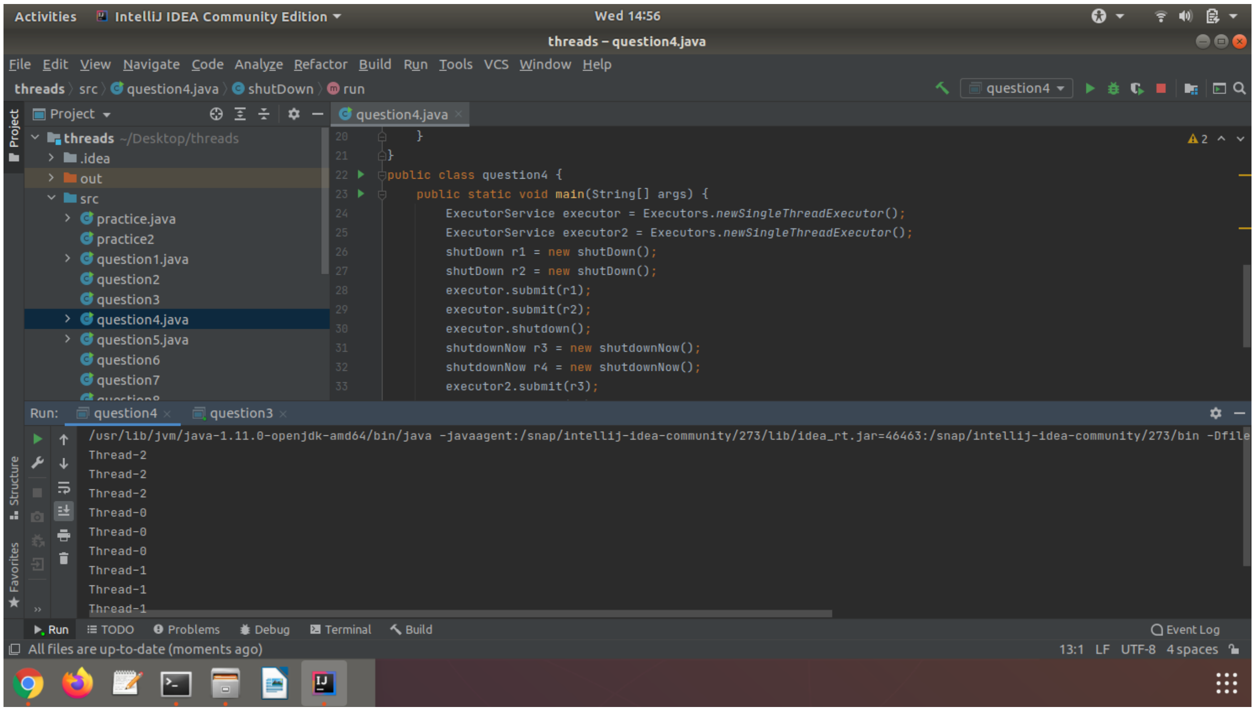
```java
        for(int i=0;i<3;i++)
        {
            try{
                t.sleep(100);

            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println(t.getName());
        }
    }

    public static void main(String[] args) {
        ExecutorService executor= Executors.newSingleThreadExecutor();
        executor.submit(new question3());
        executor.submit(new question3());
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=36399:/snap/intellij-idea-community/273/bin -Dfile
Thread-1
Thread-1
Thread-1
Thread-3
Thread-3
Thread-3
```

4. Try shutdown() and shutdownNow() and observe the difference.

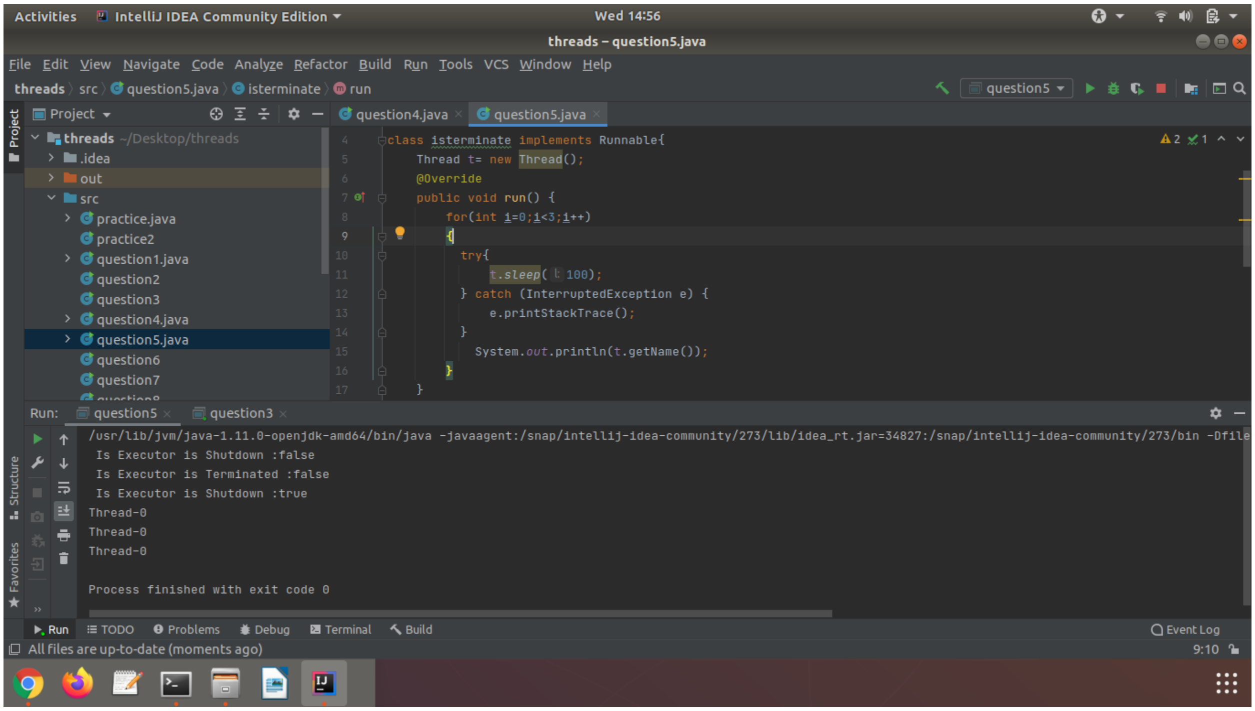


Activities
IntelliJ IDEA Community Edition
Wed 14:56
threads – question4.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
threads src question4.java shutDown run
question4
Project
threads ~/Desktop/threads
.idea
out
src
practice.java
practice2
question1.java
question2
question3
question4.java
question5.java
question6
question7
question4.java
Run: question4 question3
Run TODO Problems Debug Terminal Build Event Log
All files are up-to-date (moments ago)
13:1 LF UTF-8 4 spaces

```java
    }
}
public class question4 {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newSingleThreadExecutor();
        ExecutorService executor2 = Executors.newSingleThreadExecutor();
        shutDown r1 = new shutDown();
        shutDown r2 = new shutDown();
        executor.submit(r1);
        executor.submit(r2);
        executor.shutdown();
        shutdownNow r3 = new shutdownNow();
        shutdownNow r4 = new shutdownNow();
        executor2.submit(r3);
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=46463:/snap/intellij-idea-community/273/bin -Dfile
Thread-2
Thread-2
Thread-2
Thread-0
Thread-0
Thread-0
Thread-1
Thread-1
Thread-1
```

5. Use isShutDown() and isTerminated() with ExecutorService.



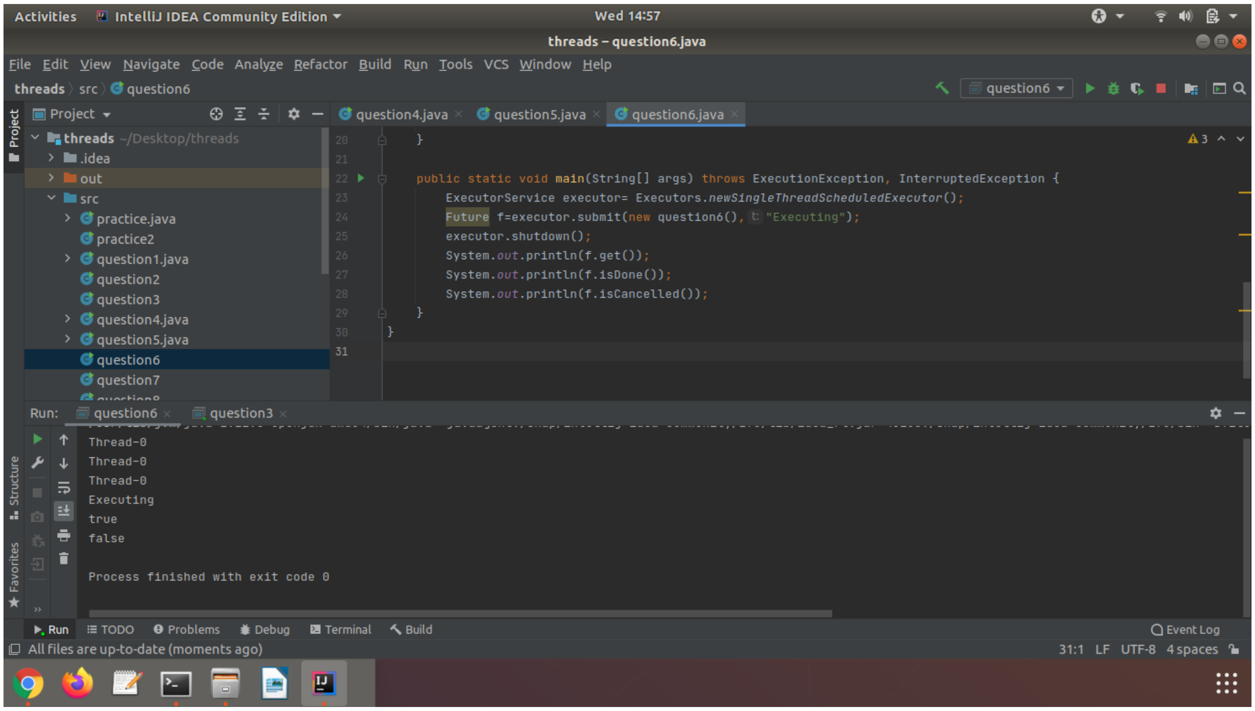
```
class isterminate implements Runnable{
    Thread t= new Thread();
    @Override
    public void run() {
        for(int i=0;i<3;i++)
        {
          try{
              t.sleep( 100);
          } catch (InterruptedException e) {
              e.printStackTrace();
          }
            System.out.println(t.getName());
        }
    }
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=34827:/snap/intellij-idea-community/273/bin -Dfile
 Is Executor is Shutdown :false
 Is Executor is Terminated :false
 Is Executor is Shutdown :true
Thread-0
Thread-0
Thread-0

Process finished with exit code 0
```

Activities
IntelliJ IDEA Community Edition
Wed 14:56
threads – question5.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
threads src question5.java isterminate run
question5
Project
threads ~/Desktop/threads
.idea
out
src
practice.java
practice2
question1.java
question2
question3
question4.java
question5.java
question6
question7
question4.java
question5.java
Run: question5 question3
Run TODO Problems Debug Terminal Build
Event Log
All files are up-to-date (moments ago)
9:10

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



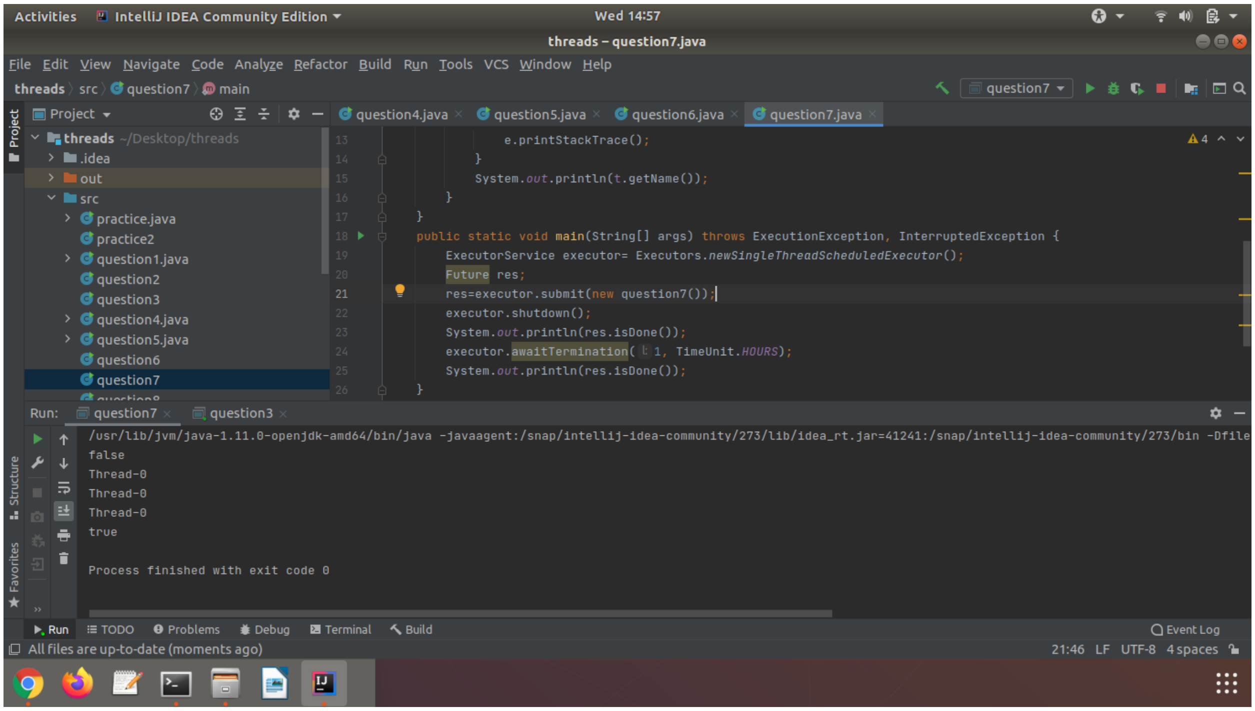
```
    }

    public static void main(String[] args) throws ExecutionException, InterruptedException {
        ExecutorService executor= Executors.newSingleThreadScheduledExecutor();
        Future f=executor.submit(new question6(), "Executing");
        executor.shutdown();
        System.out.println(f.get());
        System.out.println(f.isDone());
        System.out.println(f.isCancelled());
    }
}
```

```
Thread-0
Thread-0
Thread-0
Executing
true
false

Process finished with exit code 0
```

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

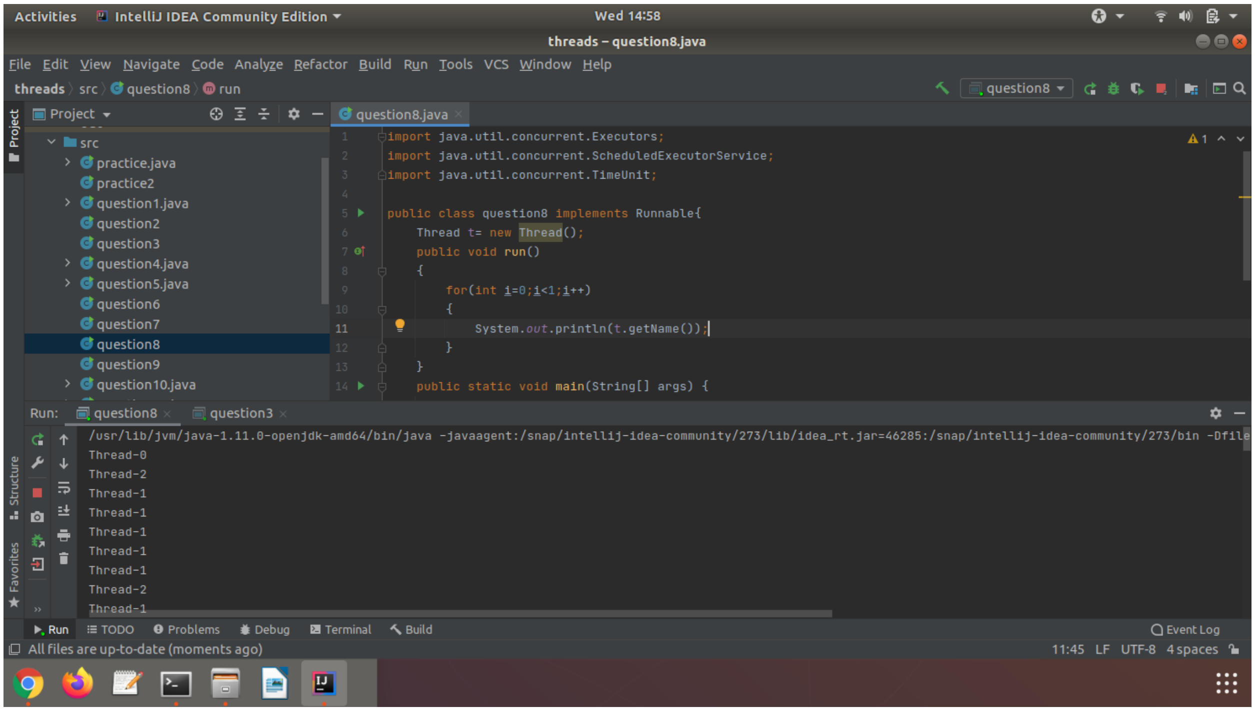


```java
                e.printStackTrace();
            }
            System.out.println(t.getName());
        }
    }
    public static void main(String[] args) throws ExecutionException, InterruptedException {
        ExecutorService executor= Executors.newSingleThreadScheduledExecutor();
        Future res;
        res=executor.submit(new question7());
        executor.shutdown();
        System.out.println(res.isDone());
        executor.awaitTermination( 1, TimeUnit.HOURS);
        System.out.println(res.isDone());
    }
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=41241:/snap/intellij-idea-community/273/bin -Dfile
false
Thread-0
Thread-0
Thread-0
true

Process finished with exit code 0
```

8. Schedule task using schedule(), scheduleAtFixedRate() and scheduleAtFixedDelay()

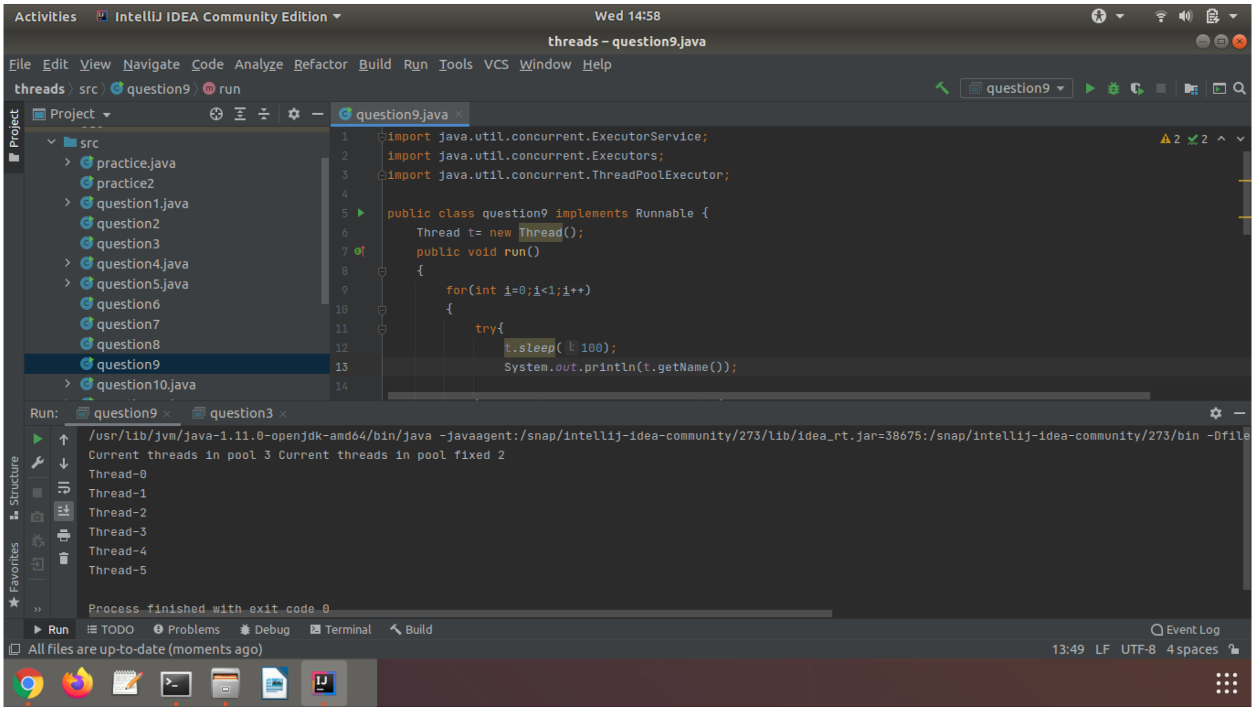


```java
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class question8 implements Runnable{
    Thread t= new Thread();
    public void run()
    {
        for(int i=0;i<1;i++)
        {
            System.out.println(t.getName());
        }
    }
    public static void main(String[] args) {
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=46285:/snap/intellij-idea-community/273/bin -Dfile
Thread-0
Thread-2
Thread-1
Thread-1
Thread-1
Thread-1
Thread-1
Thread-2
Thread-1
```

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



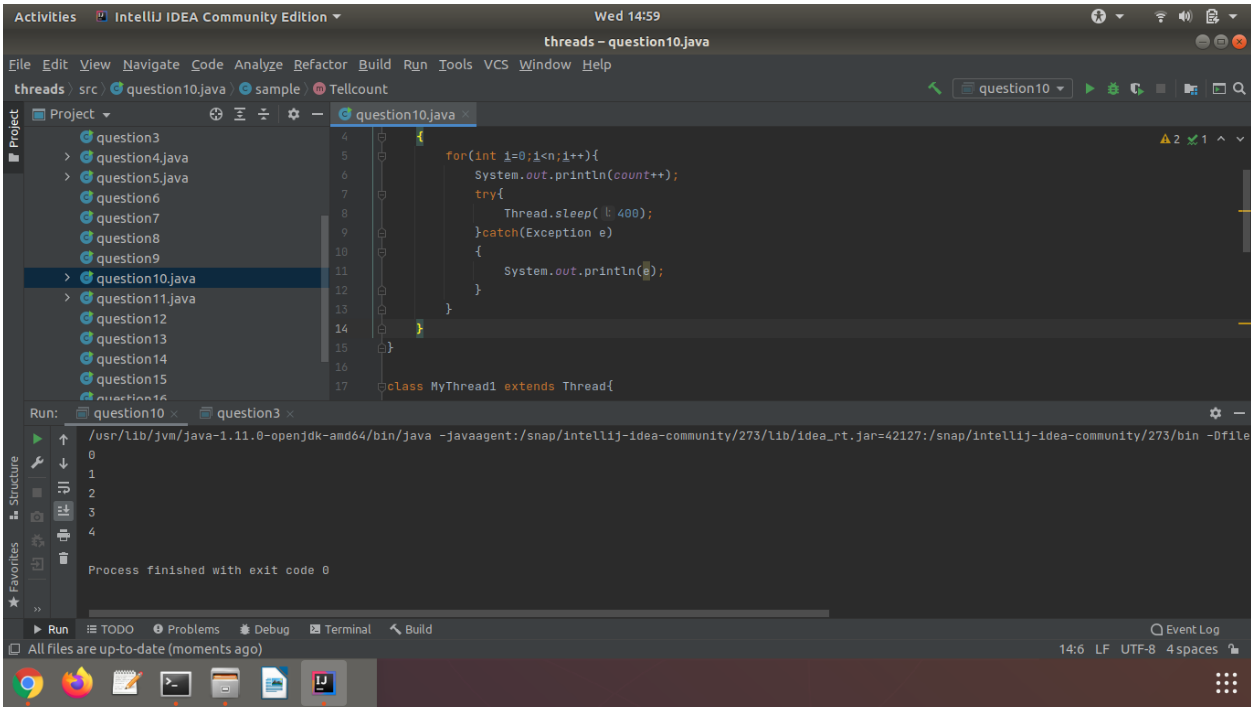
```java
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.ThreadPoolExecutor;

public class question9 implements Runnable {
    Thread t= new Thread();
    public void run()
    {
        for(int i=0;i<1;i++)
        {
            try{
                t.sleep( 100);
                System.out.println(t.getName());
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=38675:/snap/intellij-idea-community/273/bin -Dfile
Current threads in pool 3 Current threads in pool fixed 2
Thread-0
Thread-1
Thread-2
Thread-3
Thread-4
Thread-5

Process finished with exit code 0
```

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



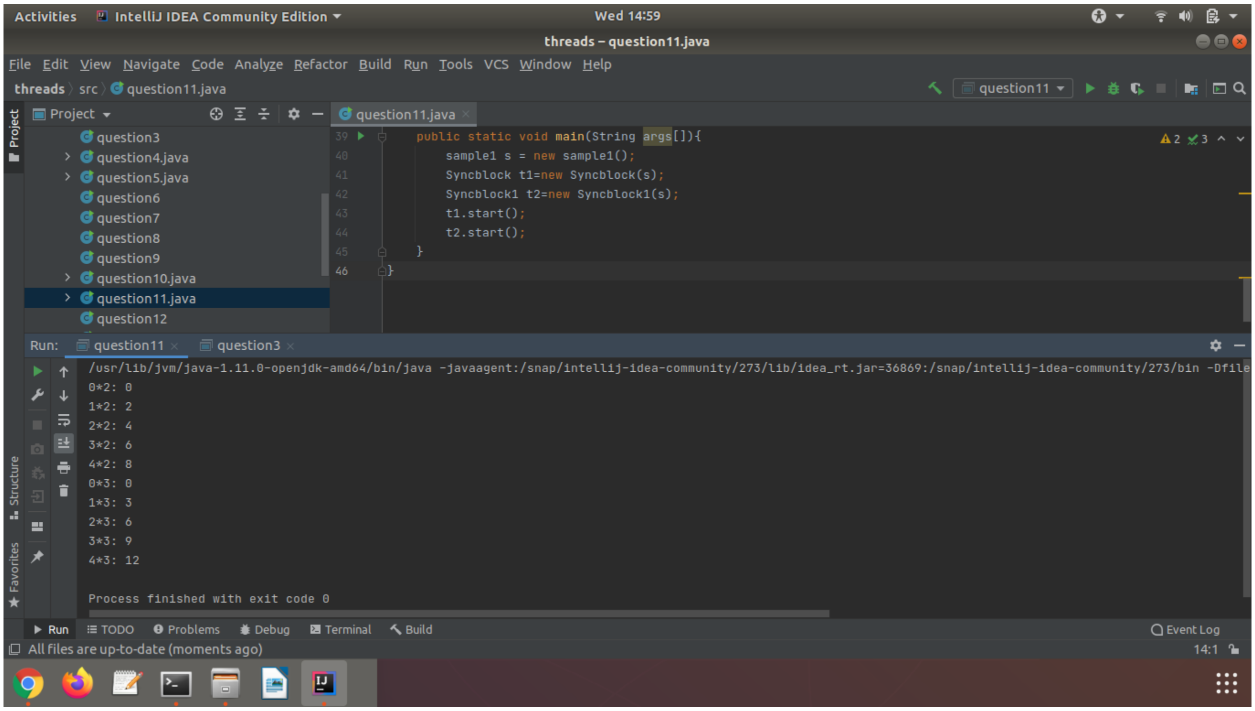
```java
    {
        for(int i=0;i<n;i++){
            System.out.println(count++);
            try{
                Thread.sleep( 400);
            }catch(Exception e)
            {
                System.out.println(e);
            }
        }
    }
}

class MyThread1 extends Thread{
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=42127:/snap/intellij-idea-community/273/bin -Dfile
0
1
2
3
4

Process finished with exit code 0
```

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

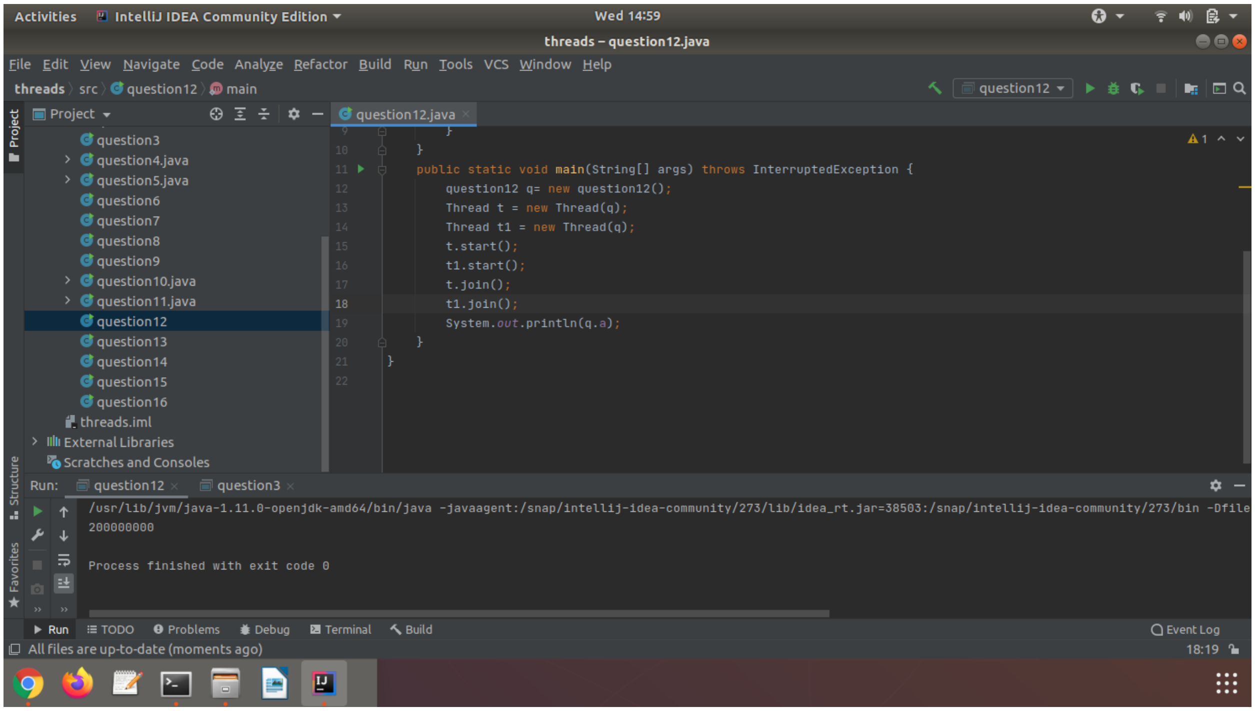


```
    public static void main(String args[]){
        sample1 s = new sample1();
        Syncblock t1=new Syncblock(s);
        Syncblock1 t2=new Syncblock1(s);
        t1.start();
        t2.start();
    }
}
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=36869:/snap/intellij-idea-community/273/bin -Dfile
0*2: 0
1*2: 2
2*2: 4
3*2: 6
4*2: 8
0*3: 0
1*3: 3
2*3: 6
3*3: 9
4*3: 12

Process finished with exit code 0
```

12. Use Atomic Classes instead of Synchronize method and blocks.

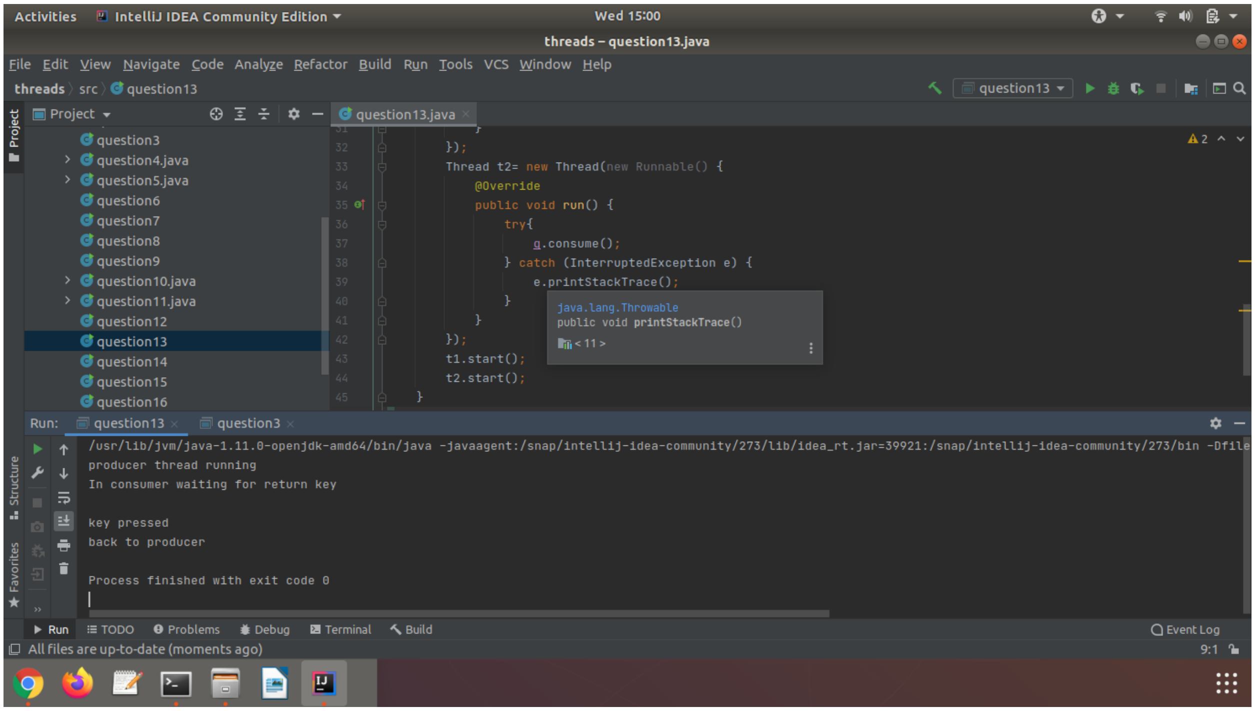


```
        }
    }
    public static void main(String[] args) throws InterruptedException {
        question12 q= new question12();
        Thread t = new Thread(q);
        Thread t1 = new Thread(q);
        t.start();
        t1.start();
        t.join();
        t1.join();
        System.out.println(q.a);
    }
}
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=38503:/snap/intellij-idea-community/273/bin -Dfile
200000000

Process finished with exit code 0
```

13. Coordinate 2 threads using wait() and notify().


Activities
IntelliJ IDEA Community Edition
Wed 15:00
threads – question13.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
threads src question13
question13
Project
question3
question4.java
question5.java
question6
question7
question8
question9
question10.java
question11.java
question12
question13
question14
question15
question16
question13.java
Run: question13 question3
Run TODO Problems Debug Terminal Build Event Log
All files are up-to-date (moments ago)
9:1

```java
            }
        });
        Thread t2= new Thread(new Runnable() {
            @Override
            public void run() {
                try{
                    q.consume();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        t1.start();
        t2.start();
    }
```

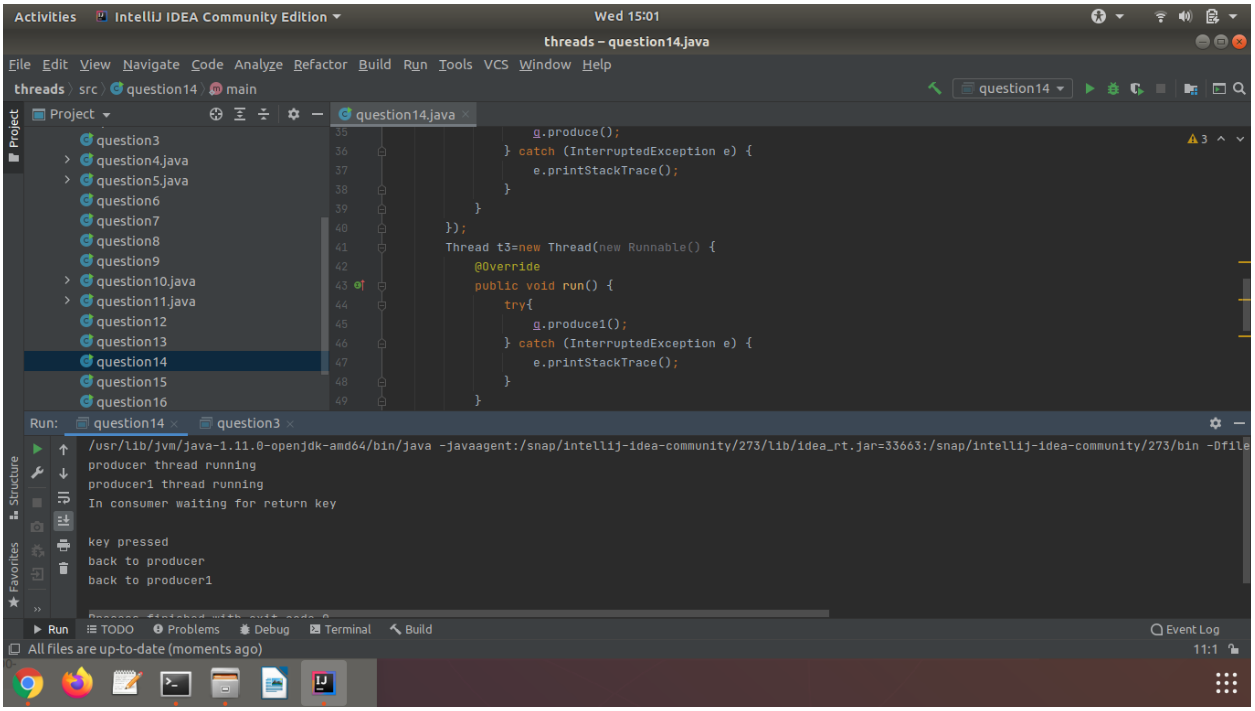
```
java.lang.Throwable
public void printStackTrace()
< 11 >
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=39921:/snap/intellij-idea-community/273/bin -Dfile
producer thread running
In consumer waiting for return key

key pressed
back to producer

Process finished with exit code 0
```

14. Coordinate mulitple threads using wait() and notifyAll()


Activities
IntelliJ IDEA Community Edition
Wed 15:01
threads – question14.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
threads src question14 main
question14
Project
question3
question4.java
question5.java
question6
question7
question8
question9
question10.java
question11.java
question12
question13
question14
question15
question16
question14.java
Run: question14 question3
Run TODO Problems Debug Terminal Build Event Log
All files are up-to-date (moments ago)
11:1

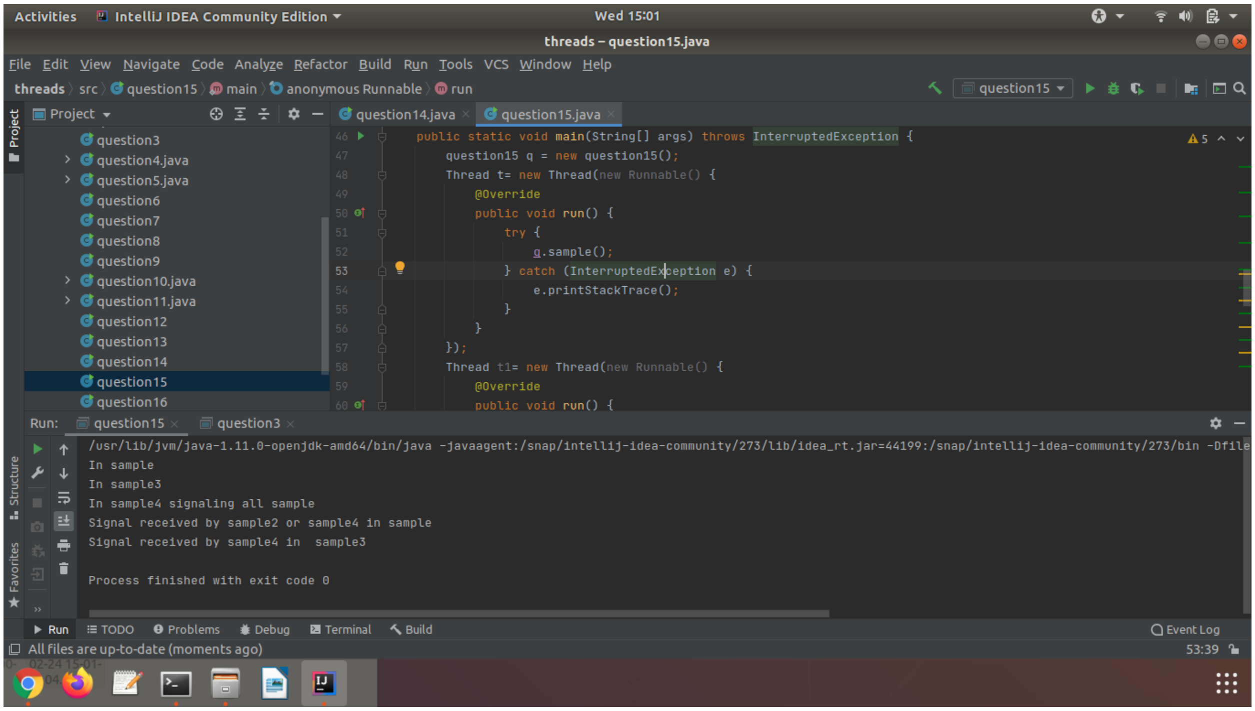
```java
                    q.produce();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        Thread t3=new Thread(new Runnable() {
            @Override
            public void run() {
                try{
                    q.produce1();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=33663:/snap/intellij-idea-community/273/bin -Dfile
producer thread running
producer1 thread running
In consumer waiting for return key

key pressed
back to producer
back to producer1

Process finished with exit code 0
```

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

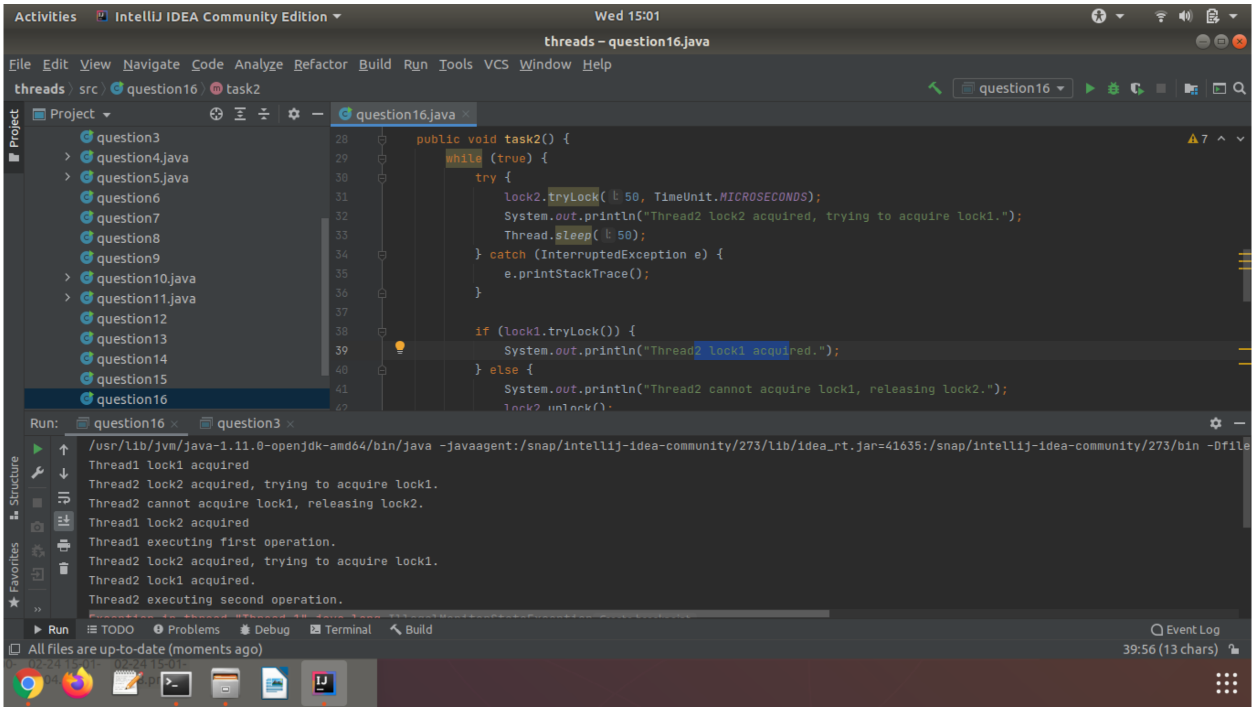


```java
    public static void main(String[] args) throws InterruptedException {
        question15 q = new question15();
        Thread t= new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    q.sample();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        Thread t1= new Thread(new Runnable() {
            @Override
            public void run() {
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=44199:/snap/intellij-idea-community/273/bin -Dfile
In sample
In sample3
In sample4 signaling all sample
Signal received by sample2 or sample4 in sample
Signal received by sample4 in  sample3

Process finished with exit code 0
```

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



```java
    public void task2() {
        while (true) {
            try {
                lock2.tryLock( 50, TimeUnit.MICROSECONDS);
                System.out.println("Thread2 lock2 acquired, trying to acquire lock1.");
                Thread.sleep( 50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            if (lock1.tryLock()) {
                System.out.println("Thread2 lock1 acquired.");
            } else {
                System.out.println("Thread2 cannot acquire lock1, releasing lock2.");
                lock2.unlock();
```

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=41635:/snap/intellij-idea-community/273/bin -Dfile
Thread1 lock1 acquired
Thread2 lock2 acquired, trying to acquire lock1.
Thread2 cannot acquire lock1, releasing lock2.
Thread1 lock2 acquired
Thread1 executing first operation.
Thread2 lock2 acquired, trying to acquire lock1.
Thread2 lock1 acquired.
Thread2 executing second operation.
```