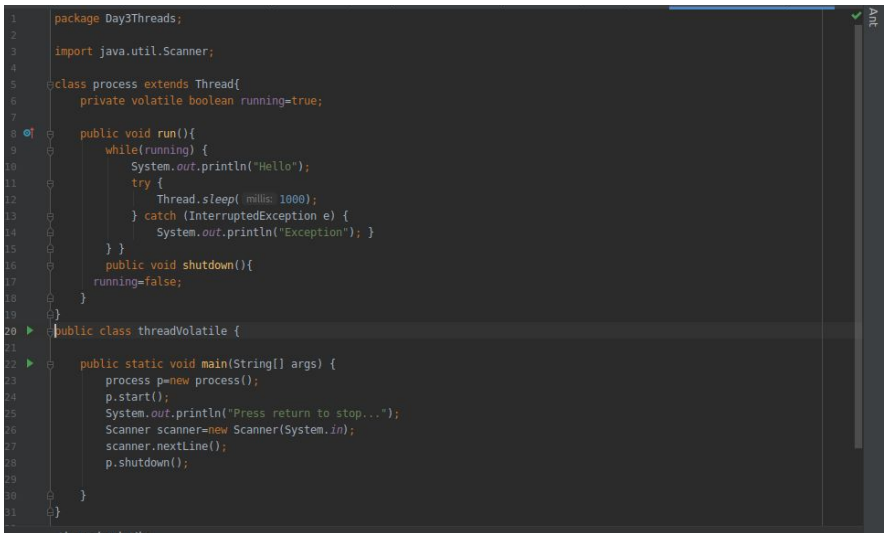
1. Write a program to demonstrate the use of volatile keyword.

Sol 1.



```
package Day3Threads;

import java.util.Scanner;

class process extends Thread{
    private volatile boolean running=true;

    public void run(){
        while(running) {
            System.out.println("Hello");
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                System.out.println("Exception"); }
        } }
        public void shutdown(){
      running=false;
    }
}
public class threadVolatile {

    public static void main(String[] args) {
        process p=new process();
        p.start();
        System.out.println("Press return to stop...");
        Scanner scanner=new Scanner(System.in);
        scanner.nextLine();
        p.shutdown();

    }
}
```

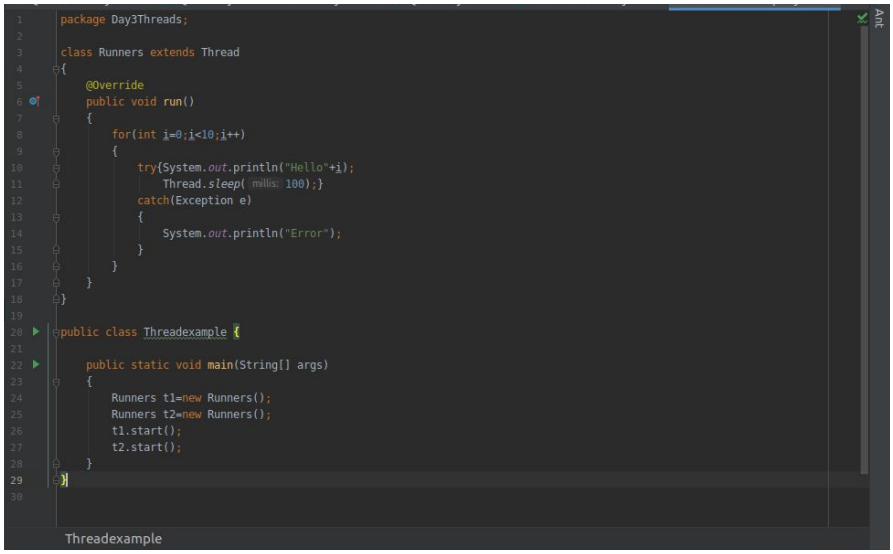
```
threadVolatile
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Hello
Press return to stop...
Hello
Hello
Hello
Hello


Process finished with exit code 0
```

2. Write a program to create a thread using Thread class and Runnable interface each.

Sol 2.

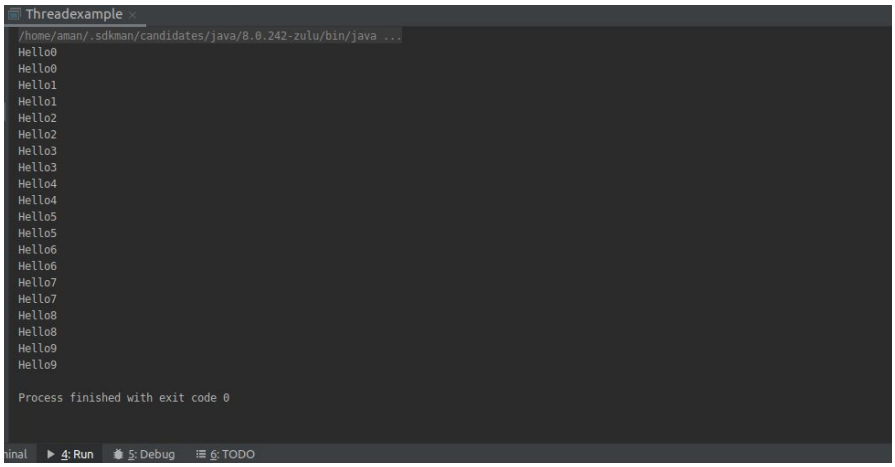
```
package Day3Threads;

class Runners implements Runnable
{
    @Override
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try{System.out.println("Hello"+i);
                Thread.sleep( millis: 100);}
            catch(Exception e)
            {
                System.out.println("Error");
            }
        }
    }
}

public class Threadexample {

    public static void main(String[] args)
    {
        Thread t1=new Thread(new Runners());
        Thread t2=new Thread(new Runners());
        t1.start();
        t2.start();
    }
}
```

```
Threadexample
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Hello0
Hello0
Hello1
Hello1
Hello2
Hello2
Hello3
Hello3
Hello4
Hello4
Hello5
Hello5
Hello6
Hello6
Hello7
Hello7
Hello8
Hello8
Hello9
Hello9

Process finished with exit code 0

inal  0: Messages  4: Run  5: Debug  6: TODO
```



```java
package Day3Threads;

class Runners extends Thread
{
    @Override
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try{System.out.println("Hello"+i);
                Thread.sleep( millis: 100);}
            catch(Exception e)
            {
                System.out.println("Error");
            }
        }
    }
}

public class Threadexample {

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

Threadexample



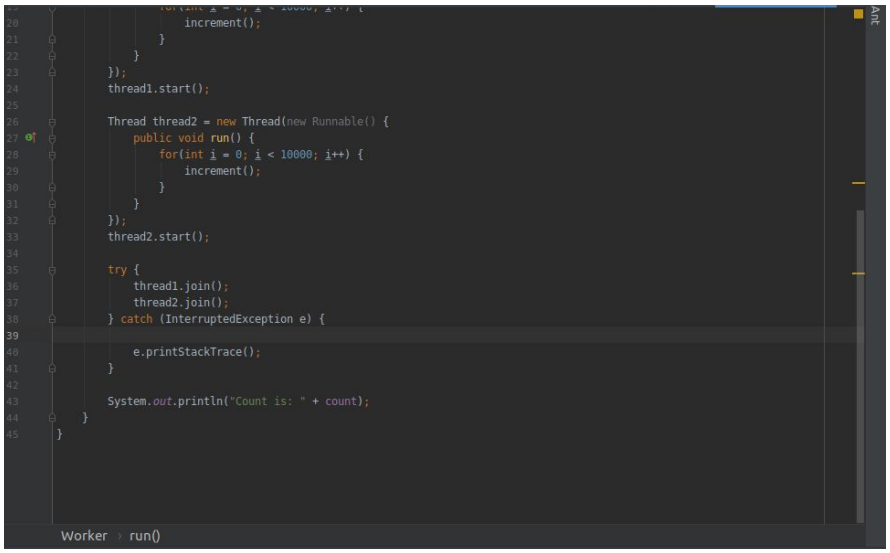
```
Threadexample
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Hello0
Hello0
Hello1
Hello1
Hello2
Hello2
Hello3
Hello3
Hello4
Hello4
Hello5
Hello5
Hello6
Hello6
Hello7
Hello7
Hello8
Hello8
Hello9
Hello9

Process finished with exit code 0

inal  ▶ 4: Run  5: Debug  6: TODO
```

3. Write a program using synchronization block and synchronization method

Sol 3.

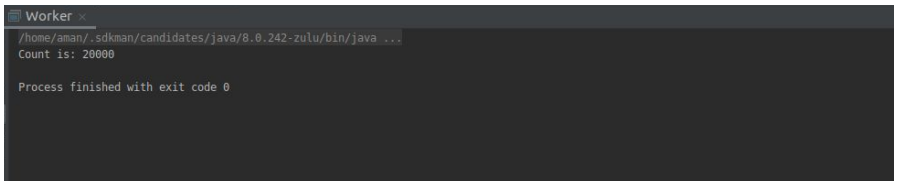
```
package Day3Threads;


public class Worker {
    private int count = 0;

    public synchronized void increment() {
        count++;
    }

    public static void main(String[] args)
    {
        Worker worker = new Worker();
        worker.run();
    }
    public void run() {
        Thread thread1 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10000; i++) {
                    increment();
                }
            }
        });
        thread1.start();

        Thread thread2 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10000; i++) {
                    increment();
                }
            }
```

Worker › run()



```
                    increment();
                }
            }
        });
        thread1.start();

        Thread thread2 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10000; i++) {
                    increment();
                }
            }
        });
        thread2.start();

        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {

            e.printStackTrace();
        }

        System.out.println("Count is: " + count);
    }
}
```

Worker › run()



```
Worker
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Count is: 20000

Process finished with exit code 0
```

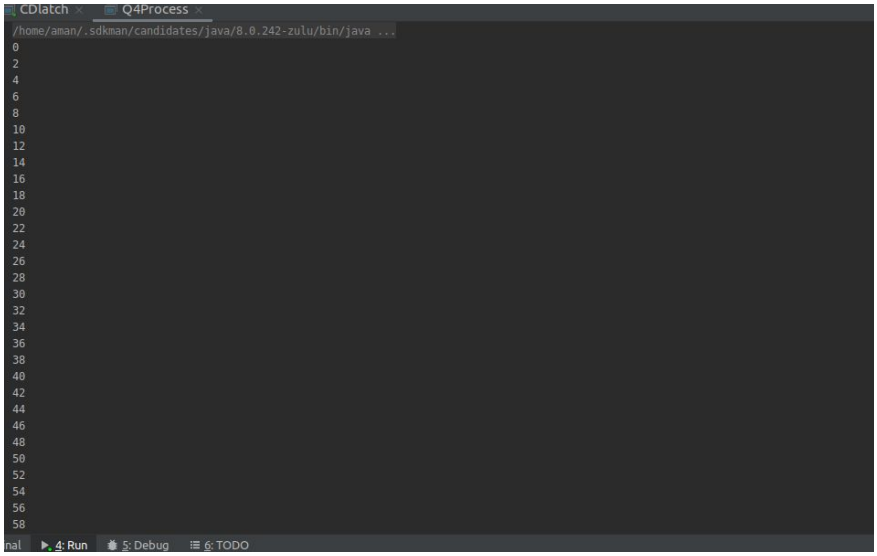
4. Write a program to create a Thread pool of 2 threads where one Thread will print even numbers and other will print odd numbers.

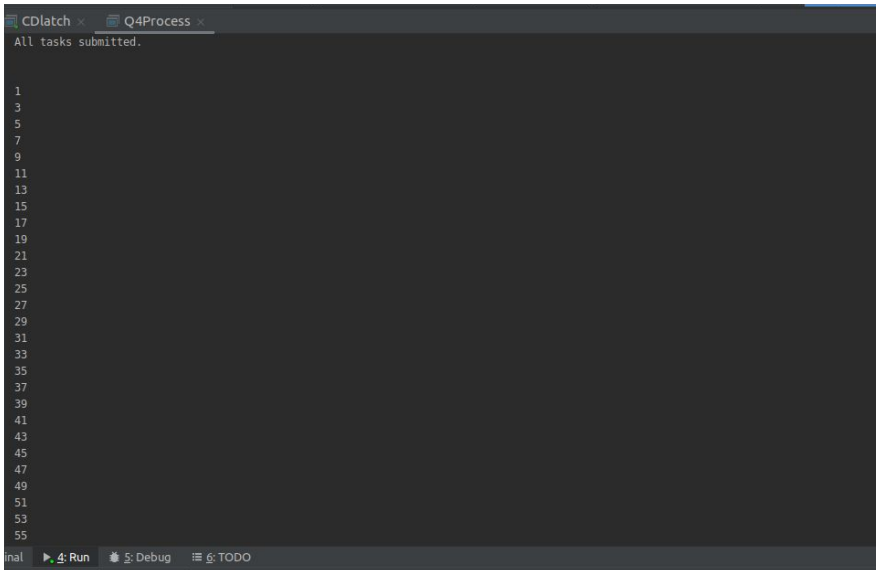
Sol 4.

```java
package Day3Threads;


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

class ThreadPoolDemo implements Runnable{

    public void run() {
        for (int i=0; i<100;i=i+2){
            System.out.println(i);
        }

        try {
            Thread.sleep( millis: 500);
        } catch (InterruptedException e) {
        }
        System.out.println("\n");

        for (int i=1;i<100;i=i+2){
            System.out.println(i);
        }
    }
}
```

```java
public class Q4Process {
    public static void main(String[] args) {

        ExecutorService executor = Executors.newFixedThreadPool( nThreads: 2);


        executor.submit(new ThreadPoolDemo());

        executor.shutdown();

        System.out.println("All tasks submitted.");

        try {
            executor.awaitTermination( timeout: 1, TimeUnit.DAYS);
        } catch (InterruptedException e) {
        }

        System.out.println("All tasks completed.");
    }
}
```

Q4Process › main()


CDlatch
Q4Process
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
4: Run
5: Debug
6: TODO


CDlatch
Q4Process
All tasks submitted.
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
4: Run
5: Debug
6: TODO

5. Write a program to demonstrate wait and notify methods.

Sol 5.

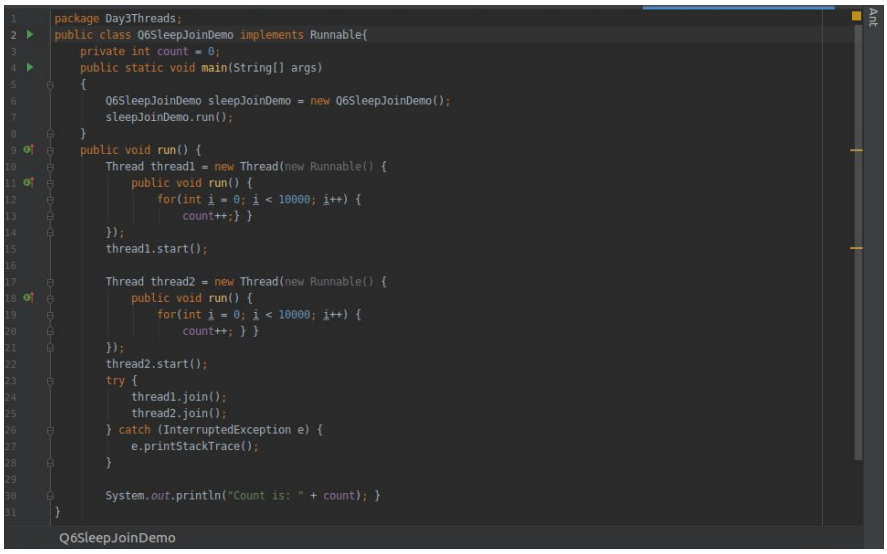
```java
package Day3Threads;

public class Q5WaitNotifyDemo {
    public static void main(String[] args) throws InterruptedException {
        final Q5Process process = new Q5Process();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    process.produce();
                }catch (InterruptedException e){

                }
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    process.consumer();
                }catch (InterruptedException e){

                }
            }
        });

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

```java


import java.util.Scanner;

public class Q5Process {
    public void produce() throws InterruptedException {
        synchronized (this) {
            System.out.println("Producer thread running ....");
            wait();
            System.out.println("Resumed.");
        }
    }

    public void consumer() throws InterruptedException{
        Scanner scanner = new Scanner(System.in);
        Thread.sleep( millis: 2000);

        synchronized (this) {
            System.out.println("Waiting for return key.");
            scanner.nextLine();
            System.out.println("Return key pressed.");
            notify();
            Thread.sleep( millis: 5000);
        }
    }
}
```

```
CDlatch × Q5WaitNotifyDemo ×
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Producer thread running ....
Waiting for return key.


Return key pressed.
Resumed.

Process finished with exit code 0
```

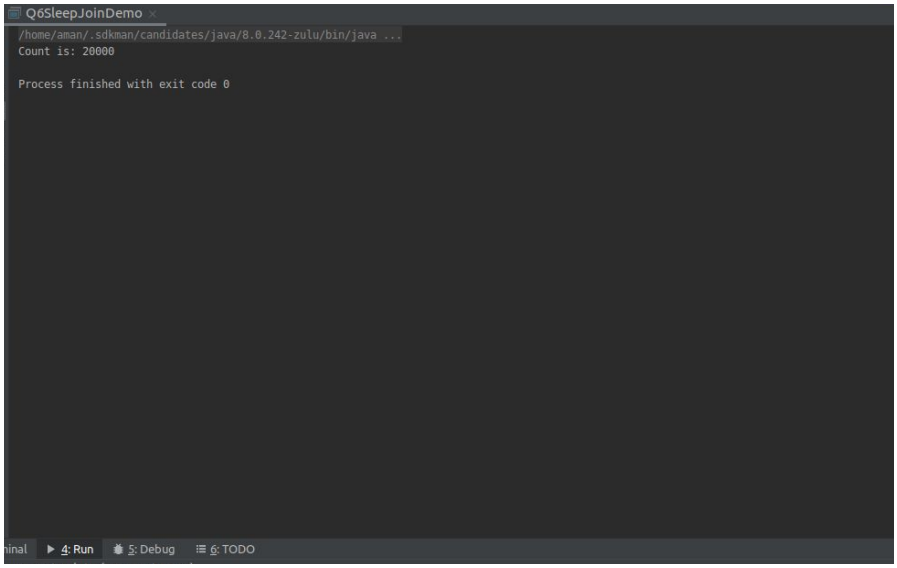
6. Write a program to demonstrate sleep and join methods.

Sol 6.



```
package Day3Threads;
public class Q6SleepJoinDemo implements Runnable{
    private int count = 0;
    public static void main(String[] args)
    {
        Q6SleepJoinDemo sleepJoinDemo = new Q6SleepJoinDemo();
        sleepJoinDemo.run();
    }
    public void run() {
        Thread thread1 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10000; i++) {
                    count++;} }
        });
        thread1.start();

        Thread thread2 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10000; i++) {
                    count++; } }
        });
        thread2.start();
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Count is: " + count); }
}
```

Q6SleepJoinDemo

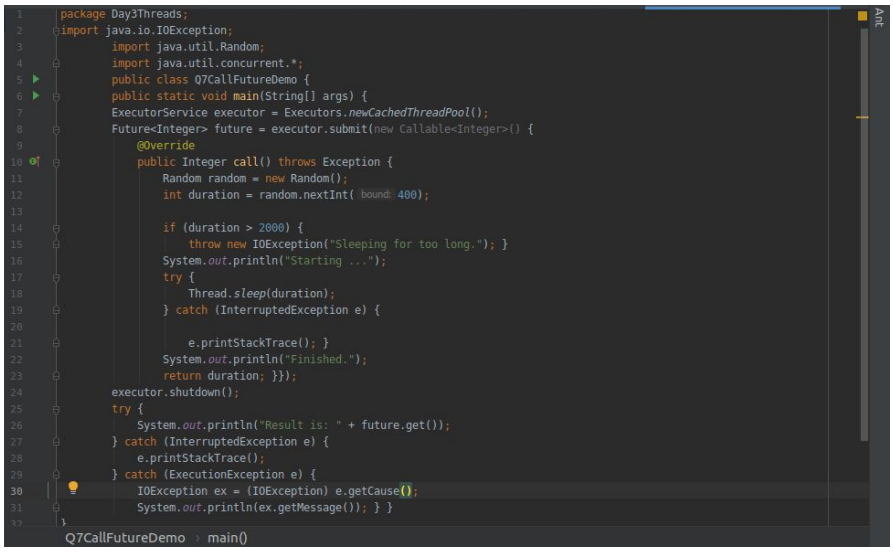


```
Q6SleepJoinDemo
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Count is: 20000

Process finished with exit code 0
```

4: Run 5: Debug 6: TODO

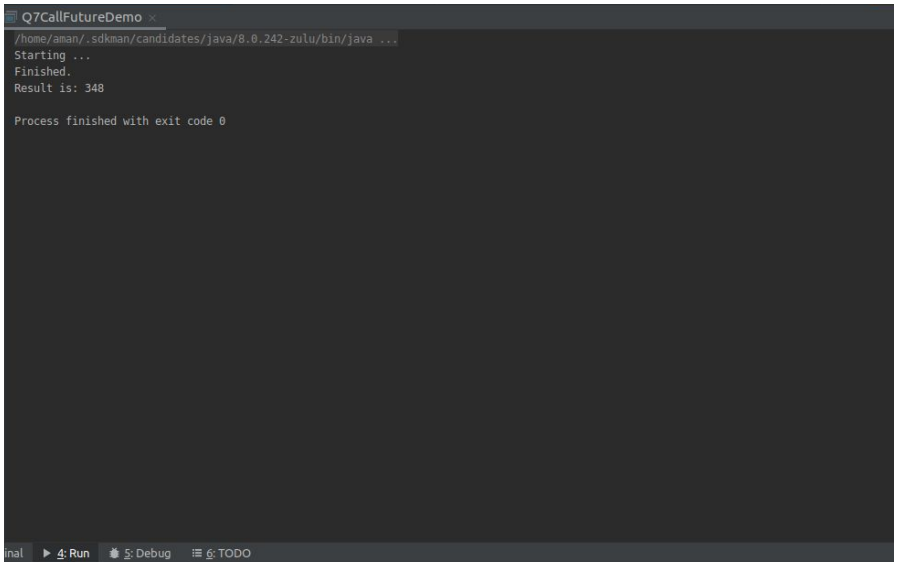
7. Run a task with the help of callable and store it's result in the Future.

Sol 7.



```
package Day3Threads;
import java.io.IOException;
        import java.util.Random;
        import java.util.concurrent.*;
        public class Q7CallFutureDemo {
        public static void main(String[] args) {
        ExecutorService executor = Executors.newCachedThreadPool();
        Future<Integer> future = executor.submit(new Callable<Integer>() {
            @Override
            public Integer call() throws Exception {
                Random random = new Random();
                int duration = random.nextInt( bound: 400);

                if (duration > 2000) {
                    throw new IOException("Sleeping for too long."); }
                System.out.println("Starting ...");
                try {
                    Thread.sleep(duration);
                } catch (InterruptedException e) {

                    e.printStackTrace(); }
                System.out.println("Finished.");
                return duration; }});
        executor.shutdown();
        try {
            System.out.println("Result is: " + future.get());
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (ExecutionException e) {
            IOException ex = (IOException) e.getCause();
            System.out.println(ex.getMessage()); } }
```

Q7CallFutureDemo › main()



```
Q7CallFutureDemo
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Starting ...
Finished.
Result is: 348

Process finished with exit code 0
```

inal ▶ 4: Run 5: Debug 6: TODO

8. Write a program to demonstrate the use of semaphore.

Sol 8.

```java
package Day3Threads;


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

public class Q8SemaphoreDemo {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newCachedThreadPool();

        for (int i = 0; i < 200; i++) {
            executor.submit(new Runnable() {
                public void run() {
                    Q8Connections.getInstance().connect();
                }
            });
        }

        executor.shutdown();

        try {
            executor.awaitTermination( timeout: 1, TimeUnit.DAYS);
        } catch (InterruptedException e) {

        }
    }
}
```

Q8SemaphoreDemo › main()

Thu 14:12

core-java [~/IdeaProjects/core-java] - .../src/Day3Threads/Q8Connections.java

ctor Build Run Tools VCS Window Help

onnections

Q7CallFutureDemo

cess.java × Q5WaitNotifyDemo.java × Q6SleepJoinDemo.java × Q7CallFutureDemo.java × Q8Connections.java ×

```java
package Day3Threads;
import java.util.concurrent.Semaphore;
public class Q8Connections {
    private static Q8Connections instance = new Q8Connections();
    private Semaphore sem = new Semaphore( permits: 10,  fair: true);
    private int connections = 0;
    private Q8Connections() { }
    public static Q8Connections getInstance() {
        return instance;
    }
    public void connect() {
        try {
            sem.acquire();
        } catch (InterruptedException e1) {
            e1.printStackTrace(); }
        try {
            doConnect();
        } finally {
            sem.release();
        } }
    public void doConnect() {
        synchronized (this) {
            connections++;
            System.out.println("Current connections: " + connections); }
            try {
            Thread.sleep( millis: 2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
            synchronized (this) {
            connections--; } }
```

Q8Connections › connect()

Q8SemaphoreDemo

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Current connections: 1
Current connections: 2
Current connections: 3
Current connections: 4
Current connections: 5
Current connections: 6
Current connections: 7
Current connections: 8
Current connections: 9
Current connections: 10
```

9. Write a program to demonstrate the use of CountDownLatch

Sol 9.

```java
package Day3Threads;


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

public class CDlatch {

    public static void main(String[] args) {

        CountDownLatch latch = new CountDownLatch(3);

        ExecutorService executor = Executors.newFixedThreadPool( nThreads: 3);

        for(int i=0; i < 3; i++) {
            executor.submit(new CDprocessor(latch));
        }

        try {
            latch.await();
        } catch (InterruptedException e)
        {
            e.printStackTrace();
        }

        System.out.println("Completed.");
    }

}
```

CDlatch › main()

```java
package Day3Threads;

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

class CDprocessor implements Runnable {
    private CountDownLatch latch;

    public CDprocessor(CountDownLatch latch) {
        this.latch = latch;
    }

    public void run() {
        System.out.println("Started.");

        try {
            Thread.sleep( millis: 3000);
        } catch (InterruptedException e) {

            e.printStackTrace();
        }

        latch.countDown();
    }
}
```

CDprocessor › run()

Debug 6: TODO Event Log

oments ago) 20:13 LF UTF-8 4 spaces

```
CDlatch
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Started.
Started.
Started.
Completed.
```

10. Write a program which creates deadlock between 2 threads

Sol 10.

```java
package Day3Threads;

class DeadAcc {
    private int balance = 10000;

    public void deposit(int amount) {
        balance += amount;
    }

    public void withdraw(int amount) {
        balance -= amount;
    }

    public int getBalance() {
        return balance;
    }

    public static void transfer(DeadAcc acc1, DeadAcc acc2, int amount) {
        acc1.withdraw(amount);
        acc2.deposit(amount);
    }

}
```

```java
package Day3Threads;
public class DeadApp {
    public static void main(String[] args) throws Exception {

        final DeadRunner runner = new DeadRunner();

        Thread t1 = new Thread(new Runnable() {
            public void run() {
                try {
                    runner.firstThread();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            public void run() {
                try {
                    runner.secondThread();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
        });
        t1.start();
        t2.start();
        t1.join();
        t2.join();
        runner.finished();
```

```java
package Day3Threads;

import java.util.Random;
       import java.util.concurrent.locks.Lock;
       import java.util.concurrent.locks.ReentrantLock;

public class DeadRunner {
    private DeadAcc acc1 = new DeadAcc();
    private DeadAcc acc2 = new DeadAcc();

    private Lock lock1 = new ReentrantLock();
    private Lock lock2 = new ReentrantLock();

    private void acquireLocks(Lock firstLock, Lock secondLock) throws InterruptedException {
        while(true) {
            // Acquire locks

            boolean gotFirstLock = false;
            boolean gotSecondLock = false;

            try {
                gotFirstLock = firstLock.tryLock();
                gotSecondLock = secondLock.tryLock();
            }
            finally {
                if(gotFirstLock && gotSecondLock) {
                    return;
```

```
            }
            finally {
                if(gotFirstLock && gotSecondLock) {
                    return;
                }

                if(gotFirstLock) {
                    firstLock.unlock();
                }

                if(gotSecondLock) {
                    secondLock.unlock();
                }
            }

            // Locks not acquired
            Thread.sleep( millis: 1);
        }
    }

    public void firstThread() throws InterruptedException {

        Random random = new Random();

        for (int i = 0; i < 10000; i++) {

            acquireLocks(lock1, lock2);

            try {
                DeadAcc.transfer(acc1, acc2, random.nextInt( bound: 100));
//                Account.transfer(acc1, acc2, );
```

```
        }
    }

    public void secondThread() throws InterruptedException {
        Random random = new Random();

        for (int i = 0; i < 10000; i++) {

            acquireLocks(lock2, lock1);

            try {
                DeadAcc.transfer(acc2, acc1, random.nextInt( bound: 100));
            } finally {
                lock1.unlock();
                lock2.unlock();
            }
        }
    }

    public void finished() {
        System.out.println("Account 1 balance: " + acc1.getBalance());
        System.out.println("Account 2 balance: " + acc2.getBalance());
        System.out.println("Total balance: "
                + (acc1.getBalance() + acc2.getBalance()));
    }
}
```

```
DeadApp
man/candidates/java/8.0.242-zulu/bin/java ...
ce: 14593
ce: 5407
20000

d with exit code 0
```