```
1  ---------------FactorialCalculator.java----------
2  package com.ameya.callables;
3
4  import java.util.concurrent.Callable;
5  import java.util.concurrent.TimeUnit;
6
7  public class FactorialCalculator implements Callable<Integer> {
8
9      private Integer num;
10     public FactorialCalculator(Integer num) {
11         this.num=num;
12     }
13     @Override
14     public Integer call() throws Exception {
15         int result=1;
16         if(num==0 || num ==1) {
17             return 1;
18         }else {
19             for(int cnt=1;cnt<=num;cnt++) {
20                 result*=cnt;
21                 TimeUnit.MILLISECONDS.sleep(20);
22             }
23         }
24         System.out.println("Result for num - "+num+" -> "+result);
25         return result;
26     }
27
28 }
29 --------------------TestCallable.java----------
30 package com.ameya.test;
31
32 import java.util.ArrayList;
33 import java.util.List;
34 import java.util.Random;
35 import java.util.concurrent.ExecutionException;
36 import java.util.concurrent.Executors;
37 import java.util.concurrent.Future;
38 import java.util.concurrent.ThreadPoolExecutor;
39
40 import com.ameya.callables.FactorialCalculator;
41
```

```
42 public class TestCallable {
43
44     public static void main(String[] args) {
45         ThreadPoolExecutor executor=(ThreadPoolExecutor)
           Executors.newFixedThreadPool(2);
46         List<Future<Integer>> resultList=new
           ArrayList<Future<Integer>>();
47         Random random=new Random();
48         for(int i=1;i<=4;i++) {
49             Integer num=random.nextInt(10);
50             FactorialCalculator calculator=new FactorialCalculator(num);
51             Future<Integer> result=executor.submit(calculator);
52             resultList.add(result);
53         }
54         for(Future<Integer> f : resultList) {
55             try {
56                 System.out.println("Future result is : "+f.get()+" And
                   Task Completion is : "+f.isDone());
57                 //f.get(long timeout,TimeUnit unit)
58             }
59             catch(InterruptedException | ExecutionException e) {
60                 e.printStackTrace();
61             }
62
63         }
64         executor.shutdown();
65     }
66
67 }
68
```