EX:NO: 8 **MULTITHREADING**

DATE:

AIM:

To write simple java program for multithreading.

1. Write a Java program to print sequence number in 3 thread

Eg Thread 1 Thread2 Thread3

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

ALGORITHM:

Step1: start

Step2: create 3 class fun1, fun2, fun3 extending Thread class

Step3: in fun1 print 1 , 4 , 7

Step4: in fun2 print 2 , 5 , 8

Step5: in fun3 print 3 , 6 , 9

Step6: stop

PROGRAM:

```
package exno8;
public class Exno8 {
    public static void main(String[] args) {
        fun t=new fun();
        t.start();
        fun1 t1=new fun1();
        t1.start();
        fun2 t2=new fun2();
        t2.start();
    }
}
class fun extends Thread{
   public void run()
   {
       System.out.println("1");
```

```
      try {
          Thread.sleep(5);
      } catch (InterruptedException e) {
          System.out.println(e);
      }
    System.out.println("4");
    try {
          Thread.sleep(8);
      } catch (InterruptedException e) {
          System.out.println(e);
      }
    System.out.println("7");
}
}
class fun1 extends Thread{
  public void run()
  {
      System.out.println("2");
      try {
          Thread.sleep(6);
      } catch (InterruptedException e) {
          System.out.println(e);
      }
    System.out.println("5");
    try {
          Thread.sleep(9);
      } catch (InterruptedException e) {
          System.out.println(e);
      }


    System.out.println("8");
}
}
```

```
class fun2 extends Thread{
  public void run()
  {
      System.out.println("3");
      try {
          Thread.sleep(7);
      } catch (InterruptedException e) {
          System.out.println(e);
      }
    System.out.println("6");
    try {
          Thread.sleep(10);
      } catch (InterruptedException e) {
          System.out.println(e);
      }
   try {
          Thread.sleep(10);
      } catch (InterruptedException e) {
          System.out.println(e);
      }
    System.out.println("9");
}
}
```

OUTPUT:

```
1
2
3
4
5
6
7
8
9
```

2. Write a java program that implements inter-thread communication for producer-consumer pattern.

ALGORITHM:

Step1: start

Step2: create a class multithreading with extending Thread class. Other classes producer n consumer class

Step3: create object for producer n consumer class

Step4: call the methods inside these class with try n catch blocks

Step5: create void run() method inside producer n consumer class

Step6: stop

PROGRAM:

```
package multithreading;
import java.util.LinkedList;
public class Multithreading {
   public static void main(String[] args)
      throws InterruptedException
   {
      final PC p = new PC();
      Thread t1 = new Thread(new Runnable() {
         @Override
         public void run()
         {
            try {
               p.produce();
            }
            catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
      });
      Thread t2 = new Thread(new Runnable() {
         @Override
         public void run()
         {
            try {
               p.consume();
            }
            catch (InterruptedException e) {
               System.out.println(e);
            }
         }
      });
      t1.start();
      t2.start();
      t1.join();
      t2.join();
   }
   public static class PC {
        LinkedList<Integer> list = new LinkedList<>();
        int capacity = 2;
        public void produce() throws InterruptedException
        {
            int value = 0;
            while (true) {
                synchronized (this)
                {
                    while (list.size() == capacity)
                        wait();

                    System.out.println("Producer produced "
                                       + value);
                    break;
    }
            }
            }
         public void consume() throws InterruptedException
         {
            int value = 0;
            while (true) {
                synchronized (this)
                {
                    while (list.size() == capacity)
                        wait();

                    System.out.println("consumer brought "
                                       + value);
                  break;
    }
            }
            }
    }
}
```

OUTPUT:

```
Producer produced 0

consumer brought 0
```

Observation(20)	
Record(5)	
Total(25)	
initial	

RESULT:

Thus the java program for multithreading is written, executed successfully.