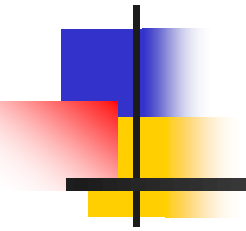


Network Programming

Consumer / Producer

Multithreads

Write a *Java program* to use a recursive function to calculate factorial?

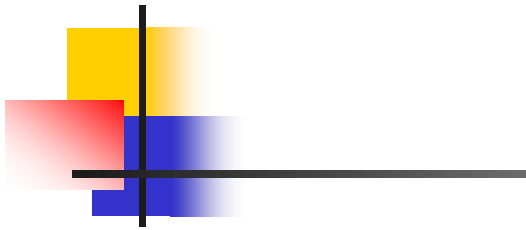
```
public class BClass
{
    public static void main(String[] args) {
            int x = 5;
            int answer;
            answer =  factorial(x);
            System.out.println("The Answer " + x + " is " + answer);
       }
    static int factorial(int number)
    {
            if(number <= 1)
                        return 1;
            return number * factorial(number - 1);
    }
 }
```

Write a Multithreaded *Java program* to use a recursive function to calculate factorial?

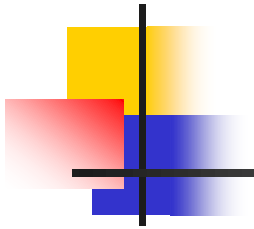
```
class  MyThread implements Runnable
{
    Thread t;
    int num;
    MyThread(int num) {
            this.num=num;
            t = new Thread(this, "My Thread");
            t.start();
    }
    int factorial(int number)
    {
        if(number <= 1)
                return 1;
        return number * factorial(number - 1);
     }
    public void run( ) {
        int answer;
        answer =  factorial(num);
        System.out.println("The Answer"
         + num + " is " + answer);
    }
}
public class BClass
{
    public static void main(String[]
                                        args)
    {
            int x = 5;
            new MyThread(x);
     }
}
```

Max and min numbers?

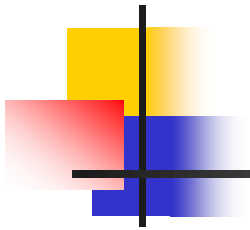
```
import java.util.Arrays;
import java.util.Collections;
import java.io.*;
public class MinMax extends Thread {
    static Integer[] numbers = { 8, 2, 7, 1, 4, 9, 5};
    int i;
     MinMax(int i) {
            this.i = i;
            this.start();
        }
        public void run() {
        if(i==0) {
             int min = (int) Collections.min( Arrays.asList(numbers) );
             System.out.println("Min number: " + min);
       }
       else {
             int max = (int) Collections.max( Arrays.asList(numbers) );
             System.out.println("Max number: " + max);
      }
```



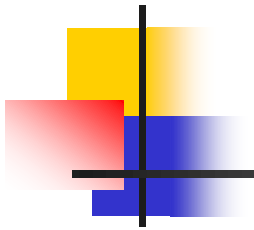
```
} // run
        public static void main(String args[])
        {
                        MinMax min = new MinMax(0);
                        MinMax max = new MinMax(1);
                         try {
                                min.join();
                                max.join();
                         } catch(Exception e){ }
                        System.out.println("done :");
        } // end main()
} // end class
```



```
class PrintJava
{
        public static void main(String args[])
        {
                Q q = new Q();

                new Producer( q );

                new Consumer( q );

                System.out.println("Press Control-C to  stop.");
        }
}
```

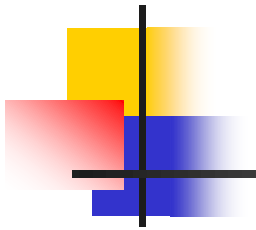


```
class Producer implements Runnable
{
        Q q;
        Producer(Q q)
        {
                this.q = q;
                new Thread(this, "Producer").start();
        }

        public void run()
        {
                int i = 0;
                while(true)
                {
                q.put(i++);
                }
        }
}
```



```
class Consumer implements Runnable
{
        Q q;

        Consumer(Q q)
        {
                this.q = q;
                new Thread(this, "Consumer").start();
        }

        public void run()
        {
                while(true)
                {
                        q.get();
                }
        }
}
```

```
class Q
{
        int n;
        boolean valueSet = false;

        synchronized int get()
        {
                if(!valueSet)
                        try
                        {
                                wait();
                        }
                        catch(InterruptedException e)
                        {
                           System.out.println(" InterruptedException caught");
                        }
                        System.out.println("Got: " + n);
                        valueSet = false;
                        notify();
                        return n;
        }
```



```
        synchronized void put(int n)
        {

                if(valueSet)
                try
                {
                        wait();
                }
                catch(InterruptedException e)
                {
                        System.out.println("InterruptedException caught");
                }
                this.n = n;
                valueSet = true;
                System.out.println("Put: " + n);
                notify();
        }
}
```