**OBJECT ORIENTED PROGRAMMING WITH JAVA 8– LAB 10**

**Q1. Write a program to create two child threads using Thread class, one to compute the first 25 prime numbers and the other to compute the first 50 fibonacci numbers. After calculating 25 fibonacci numbers, make that thread to sleep and take up prime number computation.**

```
class PFThreads
{
  public static void main(String[] args)
  {
    Thread Prime = new Thread(()->{
                                    System.out.println("First 25 prime numbers :");
                                    int z = 0;
                                    for (int n = 2; n < 25 ; n++) {
                                        if (x(n))
                                           System.out.println(n);
                                           z++;
                                        }
                                     });
    Thread Fibonacci = new Thread(()->{
                                        System.out.println("First 50 Fibonacci numbers :");
                                        int a = 0, b = 1;
                                         System.out.println(a + " " +b);
                                        for ( int i = 3; i <= 50; i++) {
                                           int c = a + b;
                                           System.out.println(" c :" +c);
                                           a = b;
                                           b = c;

                                           if (i == 24) {
                                              try
                                               {
                                                 Thread.sleep(1000);
                                                 Prime.start();
                                               }
                                               catch (InterruptedException e) {
                                                System.out.println(e);
                                               }
                                         }
                                       }
                                     });
     Fibonacci.start();

   }
   public static boolean x(int n) {
       if (n <= 1)
          return false;
       if (n <= 3)
          return true;
       if (n % 2 == 0 || n % 3 == 0)
          return false;
       for (int i = 5; i * i <= n; i+=6) {
         if ( n % i == 0 || n % (i + 2) == 0)
           return false;
         }
          return true;
         }

   }
```

```
E:\java_notes>java  PFThreads
First 50 Fibonacci numbers :
0 1
 c :1
 c :2
 c :3
 c :5
 c :8
 c :13
 c :21
 c :34
 c :55
 c :89
 c :144
 c :233
 c :377
 c :610
 c :987
 c :1597
 c :2584
 c :4181
 c :6765
 c :10946
 c :17711
 c :28657
 c :46368
First 25 prime numbers :
2
3
5
7
11
13
17
19
23
 c :75025
 c :121393
 c :196418
 c :317811
 c :514229
 c :832040
 c :1346269
 c :2178309
 c :3524578
 c :5702887
 c :9227465
 c :14930352
 c :24157817
 c :39088169
 c :63245986
 c :102334155
 c :165580141
 c :267914296
 c :433494437
```

**Q2Write a Java program to create two child threads using Runnable interface, one to find the sum of natural numbers from 1 to n and the other to display the factorial of the numbers upto n.**

```
class SumCal implements Runnable
{
  int n;
  SumCal(int num)
 {
   n = num;
 }
  public void run()                //override the run method
{
  int sum = 0;
  for (int i = 1; i <= n ; i++) {
     sum = sum+i;
     }
     System.out.println("Sum of first n Natural numbers from 1 to " + n + " is :" + sum);
    }
  }
class FactorialCal implements Runnable
{
  int n;
  public FactorialCal(int num)
  {
    n = num;
  }
  public void run()
   {
   for ( int i = 1; i <= n; i++) {
      long F = calFactorial(i);
      System.out.println("Factorial of " + i + " is :" + F);
   }
 }
 public long  calFactorial(int num)
 {
  if (num == 0 || num == 1)
      return 1;
  else
      return num * calFactorial(num - 1);
  }
}
class SumFactorialMain
{
  public static void main(String args[])
  {
    int n = 6;
    Runnable sum = new SumCal(n);
    Runnable Facto = new FactorialCal(n);
    Thread t1 = new Thread(sum);
    Thread t2 = new Thread(Facto);
    t1.start();
    t2.start();
   }
}
```

```
E:\java_notes>javac SumFactorialMain.java

E:\java_notes>java SumFactorialMain
Sum of first n Natural numbers from 1 to 6 is :21
Factorial of 1 is :1
Factorial of 2 is :2
Factorial of 3 is :6
Factorial of 4 is :24
Factorial of 5 is :120
Factorial of 6 is :720
```

**Q3Create a child thread to print Right Triangle star pattern using Lambda expression.**

```
class RightTriaStar
{
  public static void main(String args[])
  {
   Runnable r = ()-> {
                       int n = 5;
                       for ( int i = 1; i <= 5; i++) {
                          for(int j = 1; j <= i; j++) {
                             System.out.print(" * ");
                           }
                           System.out.println();
                          }
                      };
  Thread t = new Thread(r);
  t.start();
 }
}
```

```
E:\java_notes>javac RightTriaStar.java

E:\java_notes>java RightTriaStar
 *
 *  *
 *  *  *
 *  *  *  *
 *  *  *  *  *
```

**Q4. Create a BankAccount class with data members accno, balance and methods deposit and withdraw. Create an object for it which is a joint account. Two threads are using the same account. One thread is trying to deposit an amount to that account and second thread trying to withdraw an amount from it after checking the minimum balance. Implement the program using synchronization.**

```
E:\java_notes>javac joinAccount.java

E:\java_notes>java joinAccount
Deposits Rs:1000.0
Balance after deposit :6000.0
Withdrawing Rs :400.0
Remaining amount Rs:5600.0
```

```
class BankAccount
{
    int accno;
    double balance;
  BankAccount(int a,double b)
  {
    accno = a;
    balance = b;
  }
  synchronized void deposit(double amt)
  {
    System.out.println("Deposits Rs:" +amt);
    balance += amt;
    System.out.println("Balance after deposit :" + balance);
   }
   synchronized void withdraw(double amt)
   {
    if (balance - amt >= 0) {
       System.out.println("Withdrawing Rs :" + amt);
       balance = balance - amt;
       System.out.println("Remaining amount Rs:" + balance);
       }
    else
      {
       System.out.println("No Balance In Account.Please Deposit Money In Your Account");
    }
   }

    synchronized double getBalance()
    {
      return balance;
    }
 }
 class joinAccount
 {
   public static void main(String args[])
   {
     BankAccount ja = new BankAccount(12340,5000.00);
     Thread t1 = new Thread(()->{
                                ja.deposit(1000.00);
                                });
     Thread t2 = new Thread(()->{
                                double minBalance = 300.0;
                                synchronized (ja) {
                                   if(ja.getBalance() >= minBalance)
                                      ja.withdraw(400.00);
                                   else
                                      System.out.println("No balance for withdrawal");
                                   }
                                 });
  t1.start();
  t2.start();
  }
}
```