# **USCS3P01:USCS303-Operating System (OS) Practical-05**

## 

## **Threads**

Contents

[**USCS3P01:USCS303-Operating System (OS) Practical-05** 1](#_Toc79788542)

[**Threads** 1](#_Toc79788543)

[**Practical Date: 13th August,2021** 2](#_Toc79788544)

[**Practical Aim: Threads(Multi-Threading)** 2](#_Toc79788545)

[**Thread States: Life Cycle of a Threads** 2](#_Toc79788546)

[2](#_Toc79788547)

[**1.** **New and Runnable States :** 3](#_Toc79788548)

[**2.** **Waiting State:** 3](#_Toc79788549)

[**3.** **Timed Waiting State:** 3](#_Toc79788550)

[**4.** **Blocked State:** 3](#_Toc79788551)

[**5.** **Terminated State:** 3](#_Toc79788552)

[**Summation** 4](#_Toc79788553)

[**Source Code:** 4](#_Toc79788554)

[**Primes** 7](#_Toc79788555)

[**Question-02:** 7](#_Toc79788556)

[**Source Code 1:** 7](#_Toc79788557)

[**Source Code 2:** 8](#_Toc79788558)

[**Output:** 11](#_Toc79788559)

[**Fibonacci** 12](#_Toc79788560)

[**Question-03:** 12](#_Toc79788561)

[**Source Code:** 12](#_Toc79788562)

[**Output :** 14](#_Toc79788563)

### **Practical Date: 13th August,2021**

### **Practical Aim: Threads(Multi-Threading)**

### 

### **Thread States: Life Cycle of a Threads**

**Thread States: Life Cycle of a Threads**

A java thread can be in any of following thread states during its life cycle i.e.

* New,
* Runnable,
* Blocked,
* Waiting,
* Timed Waiting or Terminated.

New

Acquire lock interrupt I/O completes

Program start the thread

Runnable

Enter Syncronized statement

Notify notifyAll

Task completes

Interval expires notify notifyAll

Wait sleep

wait

Waiting

Timed waiting

Terminated

Blocked

### **New and Runnable States :**

* A new thread begins its life cycle in the new state.
* It remains in this state until the program starts the thread , which places in the running state.
* A thread in the runnable state is considered to be excuting its task.

### **Waiting State:**

* Sometimes a runnable thread transition to the waiting state while it waits for another thread to perform a task.
* A waiting thread transition back to the runnable state only when another thread notifies it to continue executing .

1. **Timed Waiting State:**

* A runnable thread can enter the timed waiting state for a specified interval of time . It transition back to the runnable state when the time interval expires or when the event it’s waiting for occurs .

1. **Blocked State:**

* A runnable thread transition to the blocked state when it attempts to perform a task that cannot be complete immediately and it must temporarily wait until the task completes.

1. **Terminated State:**

* A runnable thread enters the terminated state (sometimes called dead state) when it successfully completes its task or otherwise terminates (perhaps due to an error).

### **Summation**

**Summation**

**Question-01:**

Write a multithreaded java program that determines the summation of a non -negative integer. The Summation class implements the Runnable interface . Thread creation is performed by creating an object instance of the Thread class and passing the constructor a Runnable object.

### **Source Code:**

**//Name: Gaurang Sanyasi**

**// Batch: B2**

**// PRN: 202001640078546**

**// Date: 13th August 2021**

**// Prac-05: Threads**

**class P5\_Q1\_Summation\_BL implements Runnable**

**{**

**int upperLimit,sum;**

**public P5\_Q1\_Summation\_BL(int upperLimit)**

**{**

**this.upperLimit=upperLimit;**

**}**

**public void run()**

**{**

**for(int i =1;i<=upperLimit;i++)**

**sum +=i;**

**}**

**}//ends of class P5\_Q1\_Summation\_GS**

**public class P5\_Q1\_SummationTest\_GS**

**{**

**public static void main(String args[])**

**{**

**if(args.length<= 0)**

**System.out.println("Usage: P5\_Q1\_SummationTest\_BL<integervalue>");**

**else**

**{**

**int upp = Integer.parseInt(args[0]);**

**if(upp<=0)**

**System.out.println("args[0]:" + args[0] + " must be a positive number");**

**else**

**{**

**P5\_Q1\_Summation\_BL s = new P5\_Q1\_Summation\_BL(upp);**

**Thread t = new Thread(s);**

**t.start();**

**try{**

**t.join();**

**System.out.println("The sum of first " + upp + " elements is " + (s.sum));**

**}**

**catch(Exception e){**

**e.printStackTrace();**

**}**

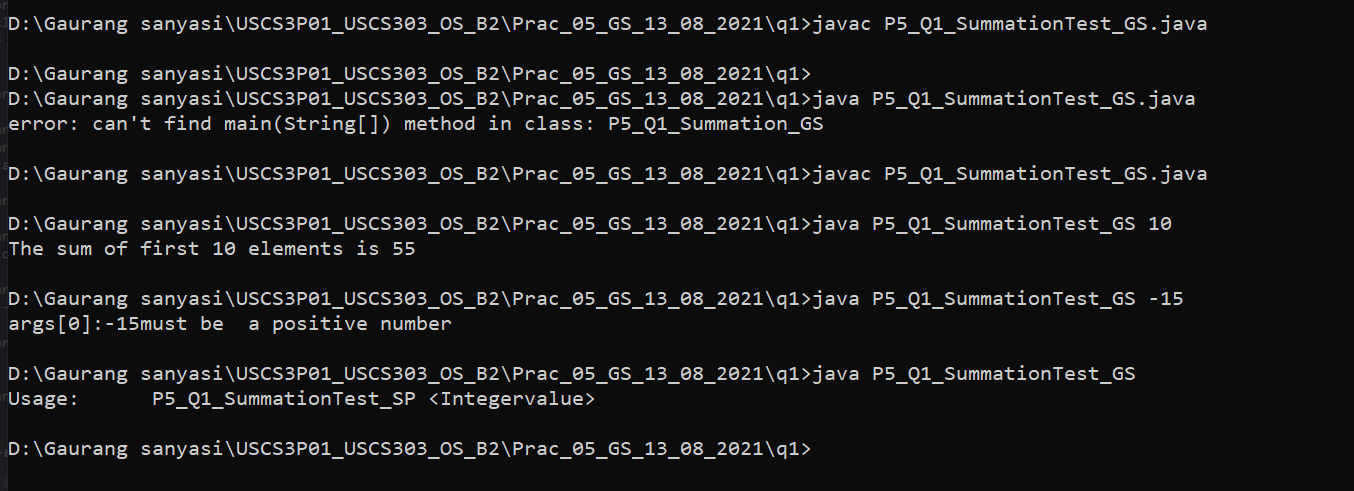
**}//inner else ends**

**}//outer else ends**

**}//main ends**

**}//end of class class P5\_Q1\_SummationTest\_GS**

**Output:**

****

### **Primes**

**Primes**

### **Question-02:**

Write a multithreaded java program that outputs prime numbers. This program should work as follows :

The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the numbers entered by the user.

### **Source Code 1:**

**//Name: : Gaurang Sanyasi**

**// Batch: B2**

**// PRN: 202001640078546**

**// Date: 13th August 2021**

**// Prac-05: Threads**

**import java.io.\*;**

**import java.util.\*;**

**public class P5\_Q2\_Primes\_GS {**

**public static void main(String args[]){**

**try{**

**P5\_Q2\_PrimeThread\_GS pt = null;**

**System.out.print("Enter a number> ");**

**Scanner scan = new Scanner(System.in);**

**int limit = scan.nextInt();**

**System.out.print("Enter a file name to store the results>");**

**String fName = scan.next();**

**if(fName.length()>0)**

**pt = new P5\_Q2\_PrimeThread\_GS(limit, new FileOutputStream(fName));**

**else**

**pt = new P5\_Q2\_PrimeThread\_GS(limit);**

**pt.run();**

**}catch(Exception e){**

**e.printStackTrace();**

**}**

**}//main ends**

**}//class ends**

### **Source Code 2:**

**//Name: Gaurang Sanyasi**

**// Batch: B2**

**// PRN: 202001640078546**

**// Date: 13th August 2021**

**// Prac-05: Threads**

**import java.io.\*;**

**class P5\_Q2\_PrimeThread\_GS extends Thread {**

**private PrintStream pOut = null;**

**private int limit = 0;**

**//default constructor.does nothing**

**public P5\_Q2\_PrimeThread\_GS(){**

**}**

**//constructor to set the number below which to generate primes**

**//no output stream is specified,so it outputs to the System.out**

**public P5\_Q2\_PrimeThread\_GS(int I){**

**limit = I;**

**try{**

**pOut = System.out;**

**}catch(Exception e){**

**e.printStackTrace();**

**}**

**}**

**//constructor that sets both the number, as above, and specifies an output stream**

**//if the specified stream is null, uses System.out**

**public P5\_Q2\_PrimeThread\_GS(int I, OutputStream outS){**

**limit = I;**

**try{**

**if(outS != null){**

**pOut = new PrintStream(outS);**

**}else{**

**pOut = System.out;**

**}**

**} catch(Exception e){**

**e.printStackTrace();**

**}**

**}**

**//method that performs the work of the thread,**

**//in this case the generation of prime numbers.**

**public void run(){**

**//compute primes via the seive**

**boolean numbers[] = new boolean[limit+1];**

**numbers[0] = false;**

**numbers[1] = false;**

**for(int i = 2; i<numbers.length; i++){**

**numbers[i] = true;**

**}**

**for(int i = 2; i<numbers.length; i++){**

**if(numbers[i]){**

**for(int j=(2\*i);j< numbers.length;j+=i){**

**numbers[j] = false;**

**}//inner for ends**

**}//if ends**

**}//outer for ends**

**for(int i=0;i< numbers.length;i++){**

**if(numbers[i])**

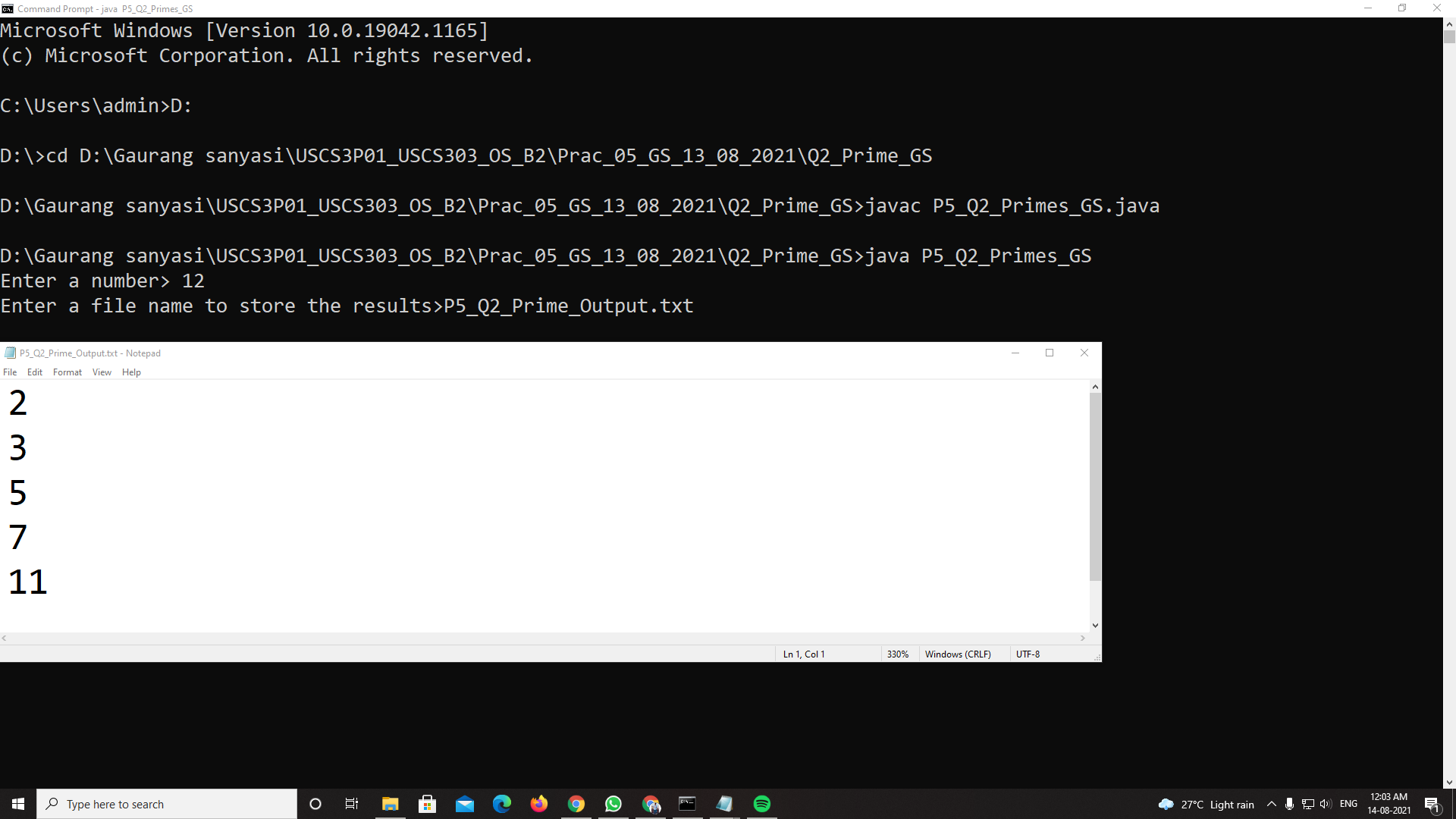
**pOut.println(i);**

**}//for ends**

**}//run ends**

**}//class ends**

### **Output:**

****

### **Fibonacci**

**Febonacci**

### **Question-03:**

The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, ….Formally, it can be expressed as : fib0 = 0, fib1 = 1, fibn = fibn-1 + fibn-2.Write a multithreaded program that generates the Fibonacci sequence using either the Java.

### **Source Code:**

**//Name: Gaurang Sanyasi**

**// Batch: B2**

**// PRN: 202001640078546**

**// Date: 13th August 2021**

**// Prac-05: Threads**

**import java.util.ArrayList;**

**import java.util.Scanner;**

**public class P5\_Q3\_Fibo\_GS**

**{**

**public static void main(String args[]){**

**Scanner scan = new Scanner(System.in);**

**ArrayList al = new ArrayList();**

**int a;**

**System.out.print("Enter the number: ");**

**a = scan.nextInt();**

**P5\_Q3\_FiboThread\_BL fibTh = new P5\_Q3\_FiboThread\_GS(a);**

**fibTh.start();**

**try{**

**fibTh.join();**

**}catch(InterruptedException ex){**

**ex.printStackTrace();**

**}**

**int fseries[] = fibTh.arr;**

**System.out.println("First "+a+" fibonacc numbers are:");**

**for(int i=0;i<a;i++){**

**System.out.print(fseries[i]+ " ");**

**}**

**}//main ends**

**}//class ends**

**class P5\_Q3\_FiboThread\_GS extends Thread**

**{**

**private int a,i;**

**Thread t;**

**int arr[];**

**public P5\_Q3\_FiboThread\_GS(int a){**

**this.a = a;**

**arr = new int[a];**

**}**

**public void run(){**

**arr[0] = 0;**

**arr[1] = 1;**

**for(i=2;i<a;i++){**

**arr[i] = arr[i-1] + arr[i-2];**

**}**

**}//run ends**

**}//class ends**

### **Output :**

