Smt. Chandibai Himathmal Mansukhani College

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

Threads

Contents

Name: ABHISHEK NIKAM_ **Batch: B2**

Smt. Chandibai Himathmal Mansukhani College

Practical Date: 13th August,2021

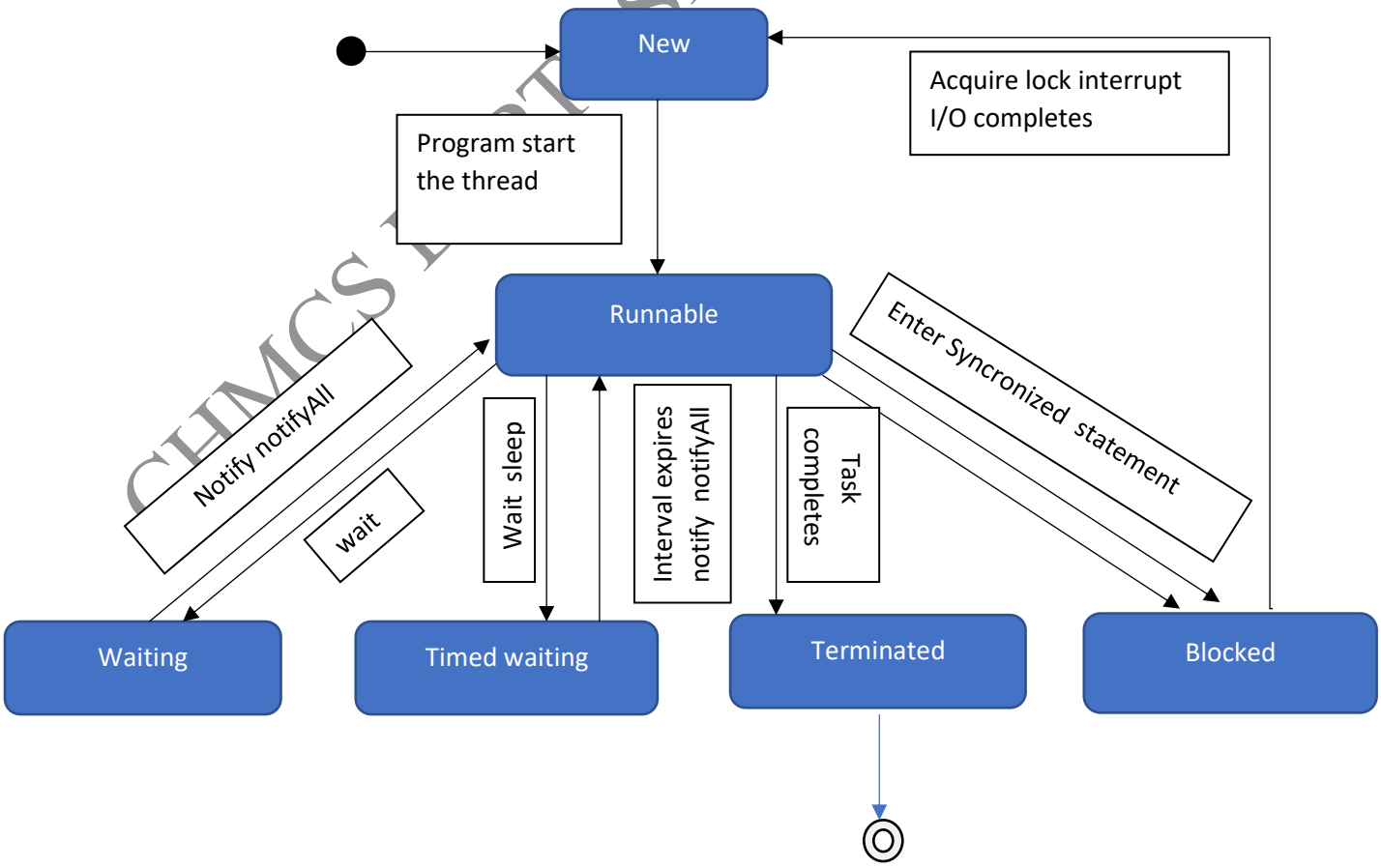
Practical Aim: Threads(Multi-Threading)

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.



Name: ABHISHEK NIKAM_ **Batch: B2**

1. **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.

2. **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 .

3. **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 .

4. **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.

5. **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: ABHISHEK NIKAM
// Batch: B2
// PRN: 2020016400805951
// Date: 13th August 2021
// Prac-05: Threads

class P5_Q1_Summation_AN implements Runnable
{
	int upperLimit,sum;
	public P5_Q1_Summation_AN(int upperLimit)
	{
		this.upperLimit=upperLimit;
	}
	public void run()
	{
		for(int i =1;i<=upperLimit;i++)
			sum +=i;
	}
}//ends of class P5_Q1_Summation_AN
```

```
public class P5_Q1_SummationTest_AN
{
	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_AN 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_AN
```

Output:

```
The sum of first 15 elements is 120
```


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: ABHISHEK NIKAM
// Batch: B2
// PRN: 2020016400805951
// Date: 13th August 2021
// Prac-05: Threads

import java.io.*;
import java.util.*;
public class P5_Q2_Primes_AN {
	public static void main(String args[]){
		try{
			P5_Q2_PrimeThread_AN 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_AN(limit, new
FileOutputStream(fName));
		else
			pt = new P5_Q2_PrimeThread_AN(limit);
```

```
                pt.run();
        }catch(Exception e){
                e.printStackTrace();
        }
   }//main ends
}//class ends
```

Source Code 2:

```
//Name: ABHISHEK NIKAM
// Batch: B2
// PRN: 2020016400805951
// Date: 13th August 2021
// Prac-05: Threads

import java.io.*;
class P5_Q2_PrimeThread_BL extends Thread {
        private PrintStream pOut = null;
        private int limit = 0;

        //default constructor.does nothing
        public P5_Q2_PrimeThread_BL(){
    }
//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_BL(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_BL(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:

```
Enter a number> 12
Enter a file name to store the results>P5_Q2_Primes_Output.txt
```

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 : $fib_0 = 0$, fib1 = 1, fibn = fibn-1 + fibn-2.Write a multithreaded program that generates the Fibonacci sequence using either the Java.

Source Code:

```
//Name: ABHISHEK NIKAM
// Batch: B2
// PRN: 2020016400805951
// Date: 13th August 2021
// Prac-05: Threads

import java.util.ArrayList;
import java.util.Scanner;

public class P5_Q3_Fibo_AN
{
        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_AN(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_AN extends Thread
{
        private int a,i;
        Thread t;
        int arr[];

        public P5_Q3_FiboThread_AN(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 :

```
Enter the number: 15
First 15 fibonacc numbers are:
0  1  1  2  3  5  8  13  21  34  55  89  144  233  377
```