Name: Yash Mahajan Class: SE IT B Roll No.:04 PID:191061

St. Francis Institute of Technology
Borivli (West), Mumbai-400103
Department of Information Technology

Experiment – 13

1. Aim: To implement concurrent programming using Java programming language

2. Objective: After performing the experiment, the students will be able to implement

▪ Understand the concept of process and thread
▪ Create a thread
▪ Perform simple thread operations

3. Lab objective mapped: To understand alternative paradigm through concurrent programming fundamentals and design, develop applications based on concurrent programming (PSO2) (PO2)

4. Prerequisite: Basics of Java programming- classes, objects, functions, data abstraction

5. Requirements: The following are the requirements – Java (JDK8) Compiler

6. Pre-Experiment Theory:

Concurrency generally refers to events or circumstances that are happening or existing at the same time. In programming terms, concurrent programming is a technique in which

▪ Two or more processes start
▪ Run in an interleaved fashion through switching and
▪ Complete in an overlapping time period by managing access to shared resources

Process

▪ Process means any program is in execution.
▪ Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc.
▪ A process can create other processes which are known as **Child Processes**.
▪ Process takes more time to terminate and it is isolated means it does not share memory with any other process.
▪ Process is called heavy-weight process

Thread

▪ One or more **threads** run in the context of the **process**.
▪ A **thread** is the basic unit to which the operating system allocates processor time. ▪ A **thread** can execute any part of the **process** code, including parts currently being executed by another **thread**
▪ **Thread is called light weight process**

7. Laboratory Exercise

A. Steps to be implemented

 1. **Compilation using JDK 8 using Turbo C**
▪ Write the code in notepad and save as a .java file.
▪ Run command prompt and set the path (Eg. Set path= ‘C:\Users\m09mu\Desktop\Javacodes’)

- Compile the code using the command ‘javac name_of_file.java’
- Correct compile time errors (if any) and rerun the code
- After successful compilation, run the code using the command ‘java name_of_file’

Using Online IDE for C/C++/Java

- Log on to www.onlinegdb.com / www.jdoodle.com
- Select the programming language for coding
- Create new project
- Save the project using CTRL+S
- Run the program using F9
- For debugging use F8, along with step-into function of onlinegdb

B. Program Code

1. Create a thread t1 to display the message “Hello World”.

```java
package Experiments;
//Creating HelloThread class that extends Thread class
class HelloThread extends Thread{
    //Defining run method
    public void run(){
        System.out.println("Hello World");
    }
}
public class Exp13_1 {
    //Driver code
    public static void main(String[] args) {
        //Instantiating object of HelloThread class
        HelloThread t1 = new HelloThread();
        t1.start();
    }
}
```

2. Create two threads t1 and t2 to display messages “Welcome to SFIT” and “Welcome to IT”.

```java
package Experiments;
//Creating myThread1 class that extends Thread class
class myThreadT1 extends Thread{
    //Defining run method
    public void run(){
        System.out.println("Welcome to SFIT");
    }
}
//Creating myThread2 class that extends Thread class
class myThreadT2 extends Thread{
    //Defining run method
    public void run(){
        System.out.println("Welcome to IT");
    }
}
public class Exp13_2{
    //Driver code
    public static void main(String[] args) {
        //Instantiating object of myThread1 class
        myThreadT1 t1 = new myThreadT1();
        t1.start();
        //Instantiating object of myThread2 class
        myThreadT2 t2 = new myThreadT2();
        t2.start();
    }
}
```

3. Create a thread t1 by implementing runnable interface to display “Hello world” message

```
package Experiments;
//class myThread Implements runnable interface
class myThread implements Runnable{
    //Defining run method
    public void run(){
        System.out.println("Hello World");
    }
}
public class Exp13_3 {
    //Driver code
    public static void main(String[] args) {
        //Instantiating object of myThread class
        myThread m1 = new myThread();
        Thread t1 = new Thread(m1);
        t1.start();
    }
}
```

Name: Yash Mahajan Class: SE IT B Roll No.:04 PID:191061

4. WAP to display n-thread ids

```java
package Experiments;

import java.util.Scanner;

//creating class nThreads that extends Thread
class nThreads extends Thread{
    public void run(){
        try {
            //Displaying the thread that is running
            System.out.println("Thread " + Thread.currentThread().getId()
+ " is running");
        } catch (Exception e) {
            //Throwing exception
            System.out.println(e);
        }
    }
}
public class Exp13_4 {
    public static void main(String[] args) {
        int n; //Number of threads
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of threads:");
        n = scanner.nextInt();
        for (int i = 0; i < n; i++) {
            //Instantiating object of class nThreads
            nThreads thread = new nThreads();
            thread.start();
        }
        scanner.close();
    }
}
```

8. Post Experimental Exercise

A. Questions:

1. Define Process and highlight its important points

A process is an active program i.e. a program that is under execution. It is more than the program code as it includes the program counter, process stack, registers, program code etc. Compared to this, the program code is only the text section.

- Processes require more time for context switching as they are more heavy.
- Processes are totally independent and don’t share memory.
- If a process gets blocked, remaining processes can continue execution.
- Processes have independent data and code segments.
- All the different processes are treated separately by the operating system .
- Individual processes are independent of each other.

2. Define thread and highlight its important points

A thread is a lightweight process that can be managed independently by a scheduler. It improves the application performance using parallelism. A thread shares information like data segment, code segment, files etc. with its peer threads while it contains its own registers, stack, counter etc.

- Threads require less time for context switching as they are lighter than processes.
- A thread may share some memory with its peer threads.
- If a user level thread gets blocked, all of its peer threads also get blocked.
- A thread shares the data segment, code segment, files etc. with its peer threads.
- All user level peer threads are treated as a single task by the operating system.
- Threads are parts of a process and so are dependent.

3. Explain the life cycle of a thread with a proper diagram.

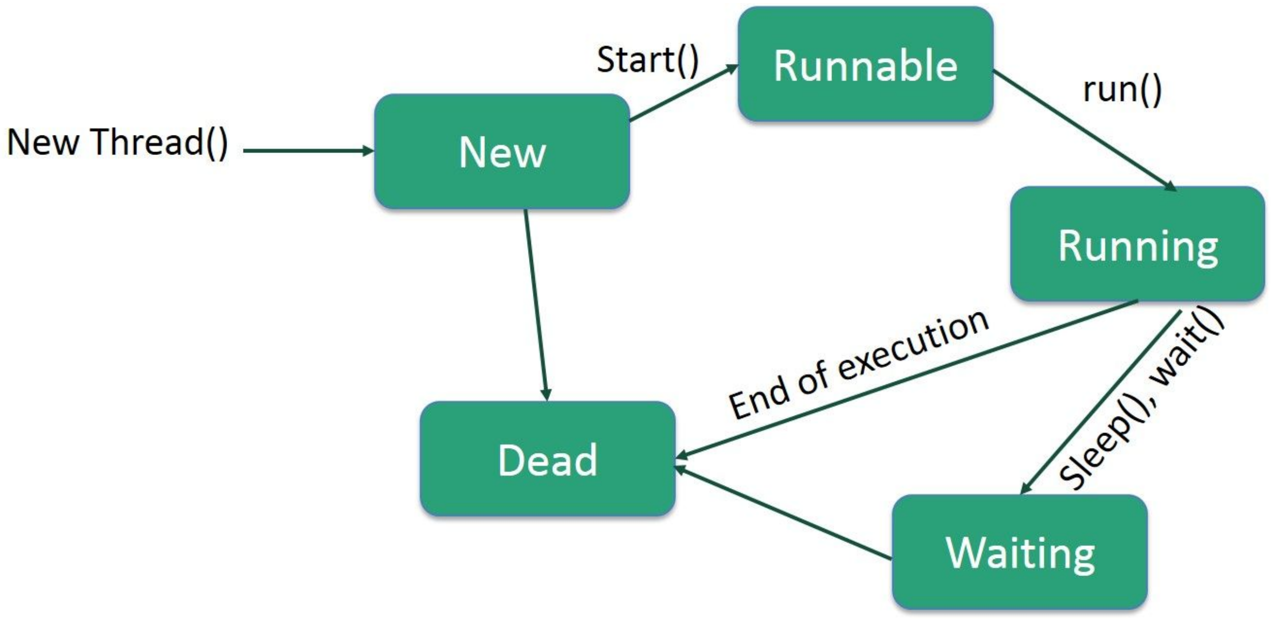


Image reference- https://www.tutorialspoint.com/java/java_multithreading.htm

- New − A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.
- Runnable − After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
- Waiting/Blocked − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. Thread transitions back to the runnable state only

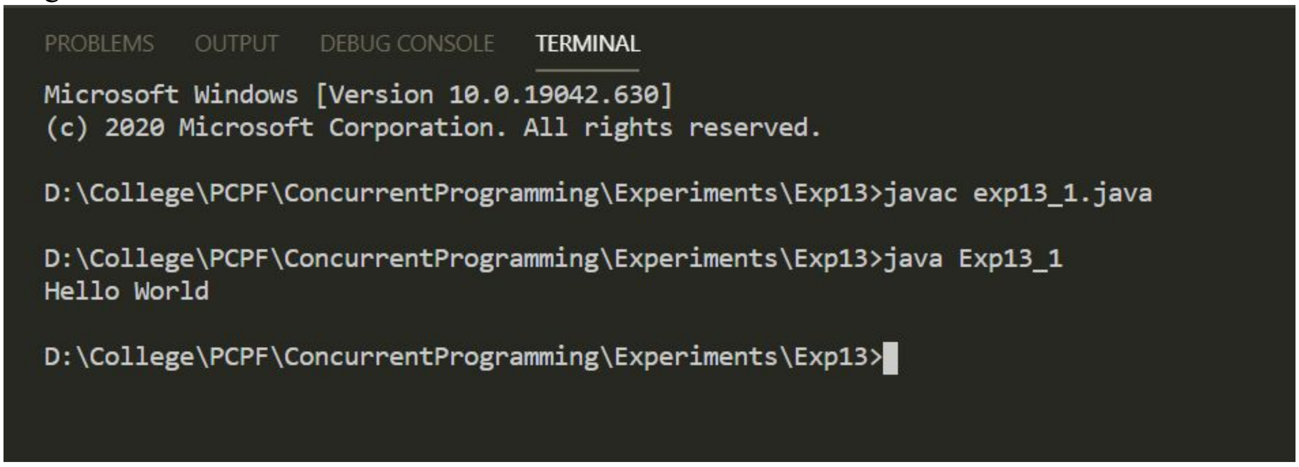
when another thread signals the waiting thread to continue executing.

- Timed Waiting − A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.
- Terminated − A runnable thread enters the terminated state when it completes its task or otherwise terminates.

B. Results/Observations/Program output:

Present the program input/output results if any and comment on the same.

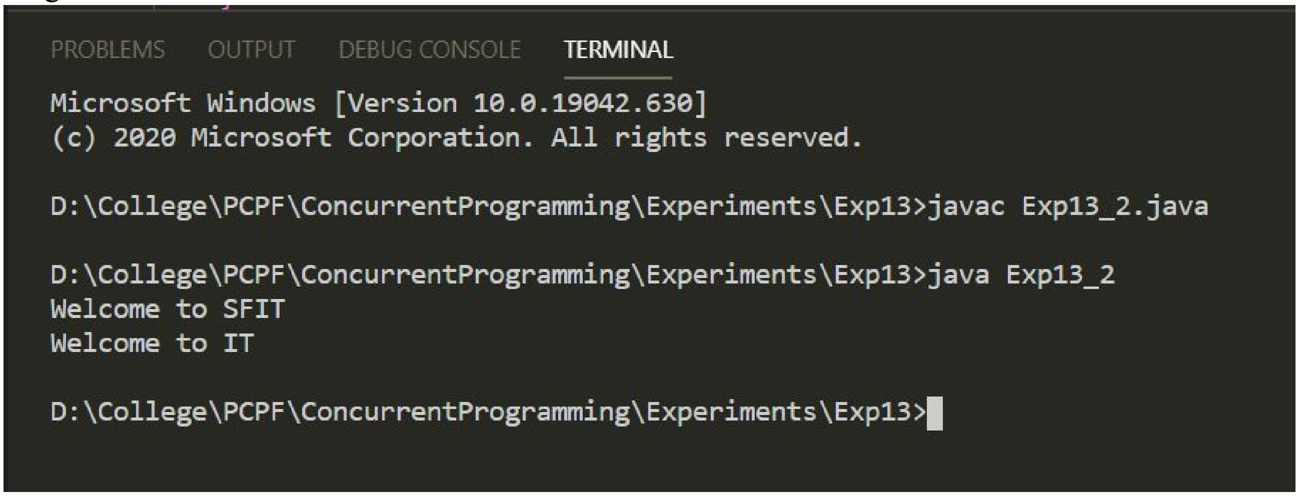
Program 1:-



```
Microsoft Windows [Version 10.0.19042.630]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>javac exp13_1.java

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>java Exp13_1
Hello World

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>
```

From this output we infer that we can create a thread in Java by extending the Thread Class.

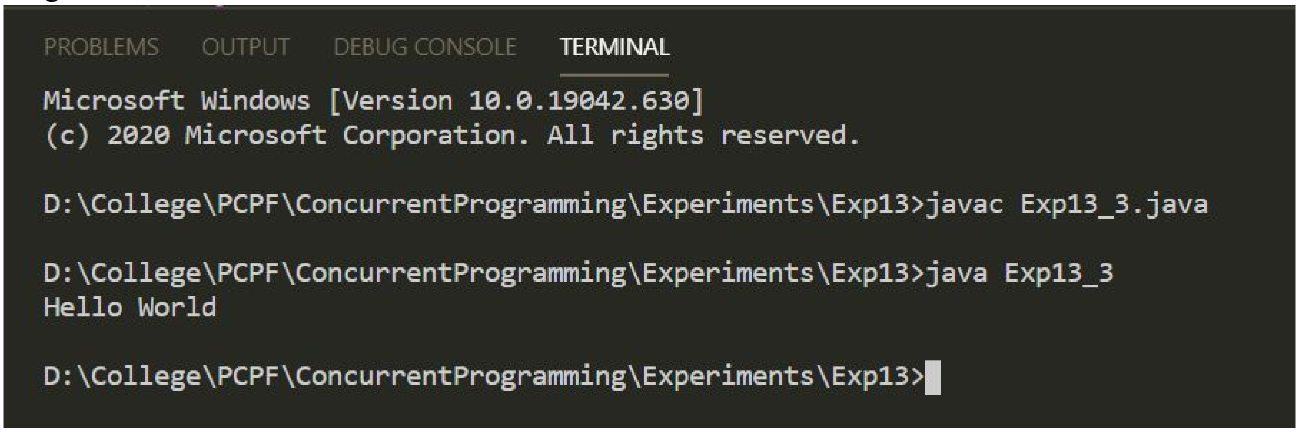
Program 2:-



```
Microsoft Windows [Version 10.0.19042.630]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>javac Exp13_2.java

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>java Exp13_2
Welcome to SFIT
Welcome to IT

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>
```

From this output we infer that we can define separate run methods for two different classes of threads.

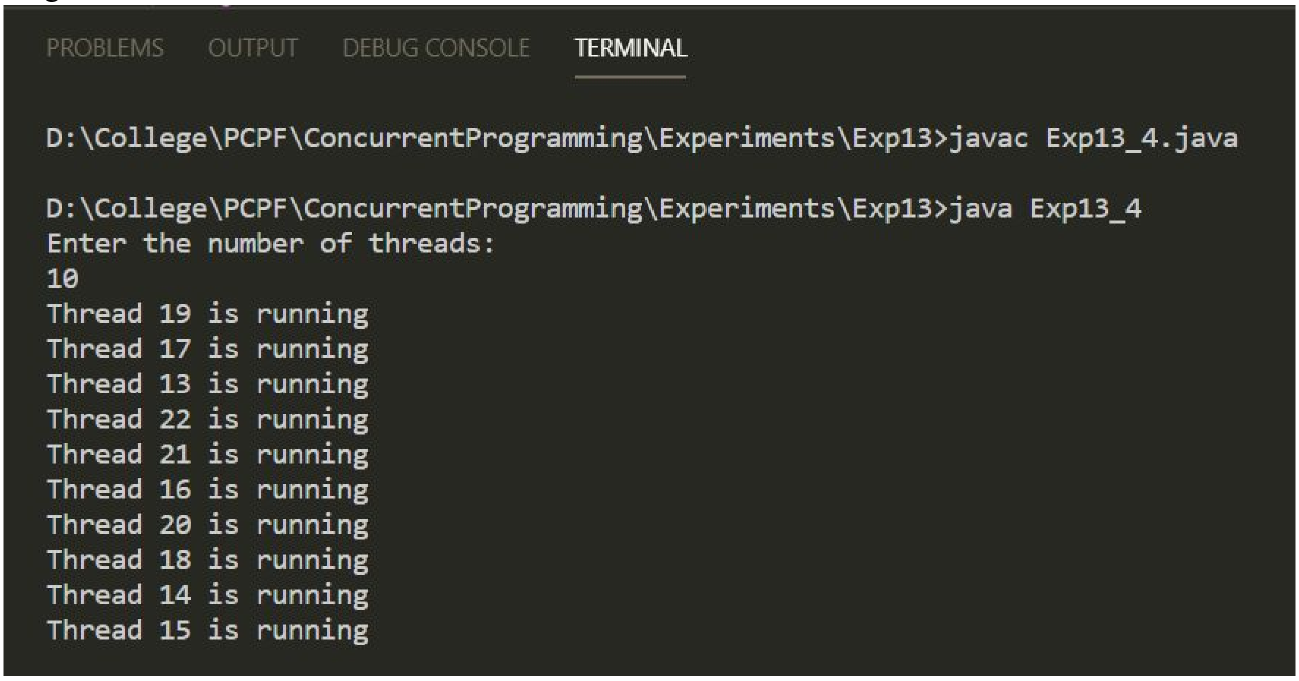
Program 3:-



```
Microsoft Windows [Version 10.0.19042.630]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>javac Exp13_3.java

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>java Exp13_3
Hello World

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>
```

From this output we can infer that we can create a thread in Java by implementing the runnable interface.

Program 4:-



```
D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>javac Exp13_4.java

D:\College\PCPF\ConcurrentProgramming\Experiments\Exp13>java Exp13_4
Enter the number of threads:
10
Thread 19 is running
Thread 17 is running
Thread 13 is running
Thread 22 is running
Thread 21 is running
Thread 16 is running
Thread 20 is running
Thread 18 is running
Thread 14 is running
Thread 15 is running
```

From this output we can infer that we can display thread id by using getId() method.

C. Conclusion:

In this experiment we have studied what multithreading is and what are its advantages and disadvantages, also we have written Java programs to implement multithreading by extending the Thread class and by implementing the Runnable interface. We have also studied that the start() method invokes the run() method on the Thread object.

To perform this experiment we have used Java JDK-15 and Visual Studio Code.

From this experiment we infer that multithreading is economical as they share the same processor resources. It takes less time to create threads. It allows the threads to share resources like data, memory, files, etc. Therefore, an application can have multiple threads within the same address space.It increases parallelism on multiple CPU machines. It enhances the performance of multiprocessor machines.

D. References:

[1] Michael L Scott, “Programming Language Pragmatics”, Third edition, Elsevier publication
[2] Doug Lea, “Concurrent Programming in Java: Design Principles and Pattern