Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 12

AIM: To implement Multithreading

THEORY:

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program. By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

CODE (i): WAP to print Table of Five, Seven and Thirteen using Multithreading (Use Thread class for the implementation). Also print the total time taken by each thread for the execution.

Code1_Multithreading.java

Exp12 > Code1_Multithreading.java > ...

```java
package Exp12;

class Five extends Thread {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i < 10; i++) {
            System.out.println("5*" + i + "=" + i * 5);
            try {
                // miliseconds time
                Thread.sleep(millis: 2000);
            } catch (InterruptedException e) {
            }
        }
        long end = System.currentTimeMillis();
        System.out.println("Total time taken by 5 table:" + (end - start));
    }
}

class Seven extends Thread {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i < 10; i++) {
            System.out.println("7*" + i + "=" + i * 7);
            try {
                // miliseconds time
                Thread.sleep(millis: 2000);
            } catch (InterruptedException e) {
            }
        }
        long end = System.currentTimeMillis();
        System.out.println("Total time taken by 7 table:" + (end -
                start));
    }
}
```

Code1_Multithreading.java

Exp12 > Code1_Multithreading.java > ...

```java

class Thirteen extends Thread {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i < 10; i++) {
            System.out.println("13*" + i + "=" + i * 13);
            try {
                // miliseconds time
                Thread.sleep(millis: 2000);
            } catch (InterruptedException e) {
            }
        }
        long end = System.currentTimeMillis();
        System.out.println("Total time taken by 13 table:" + (end -
                start));
    }
}

public class Code1_Multithreading {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Five f = new Five();
        Seven s = new Seven();
        Thirteen t = new Thirteen();
        f.start();
        s.start();
        t.start();
    }
}
```

Output:

```
Prerna Sunil Jadhav - 60004220127
7*1=7
13*1=13
5*1=5
```

```
5*8=40
13*8=104
7*9=63
5*9=45
13*9=117
Total time taken by 7 table:18091
Total time taken by 5 table:18182
Total time taken by 13 table:18239
```

CONCLUSION: Hereby, implemented multithreading in Java.

AIM: Write java program to implement the concept of Thread Synchronization

THEORY:

When we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and finally, they can produce unforeseen result due to concurrency issues. The function Thread.sleep() is used so that it sleeps a thread for the specified amount of time. Till the time another thread is running. The function isAlive() is used so that it tests if the thread is alive.(It returns a boolean value).

CODE: Write java program to implement the concept of Thread Synchronization

Code2_ThreadSync.java

Exp12 > Code2_ThreadSync.java > { } Exp12

```
1   package Exp12;
2   class Movie extends Thread {
3       int vacant = 1, required;
4       Movie(int x) {
5           required = x;
6       }
7       public synchronized void run() {
8           if (required <= vacant) {
9               System.out.println(required + " for " +
10                      Thread.currentThread().getName());
11              try {
12                  Thread.sleep(millis: 100);
13              } catch (Exception e) {
14              }
15              vacant = vacant - required;
16          } else {
17              System.out.println("none for " +
18                      Thread.currentThread().getName());
19          }
20      }
21  }
22  class Code2_ThreadSync {
        Run | Debug
23      public static void main(String z[]) {
24          System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
25          Movie m = new Movie(x: 1);
26          Thread t1 = new Thread(m);
27          Thread t2 = new Thread(m);
28          t1.setName(name: "Prerna");
29          t2.setName(name: "Jadhav");
30          t1.start();
31          t2.start();
32      }
33  }
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
1 for Prerna
none for Jadhav
```

CONCLUSION: Thus, we implemented programs on Multithreading.