| Experiment No. 14 |
| --- |
| To run multiple tasks simultaneously using Multithreading |
| Date of Performance:30/08/24 |
| Date of Submission:06/09/24 |

**Aim:-** To run multiple tasks simultaneously using Multithreading

**Objective:-** Develop a Java program that demonstrates multithreading by creating three separate threads, each responsible for displaying the multiplication table of a different number (e.g., 3, 6, and 9). Each thread should generate and print the multiplication table for its respective number up to 10. The program should illustrate the concept of multithreading by concurrently running these threads to display the results.

**Theory :-**

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.  
  
Threads can be created by using two mechanisms:   
1. Extending the Thread class  
2. Implementing the Runnable Interface

Thread creation by extending the Thread class:  
 We create a class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

**Code :**

class Thread1 extends Thread {

public void run() {

for (int i = 1; i <= 10; i++) {

System.out.println("3 \* " + i + " = " + (3 \* i));

}

}

}

class Thread2 extends Thread {

public void run() {

for (int i = 1; i <= 10; i++) {

System.out.println("6 \* " + i + " = " + (6 \* i));

}

}

}

class Thread3 extends Thread {

public void run() {

for (int i = 1; i <= 10; i++) {

System.out.println("9 \* " + i + " = " + (9 \* i));

}

}

}

public class DemoThread {

public static void main(String[] args) {

Thread1 t1 = new Thread1();

t1.start();

try {

Thread.sleep(400);

} catch (InterruptedException e) {

e.printStackTrace();

}

Thread2 t2 = new Thread2();

t2.start();

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

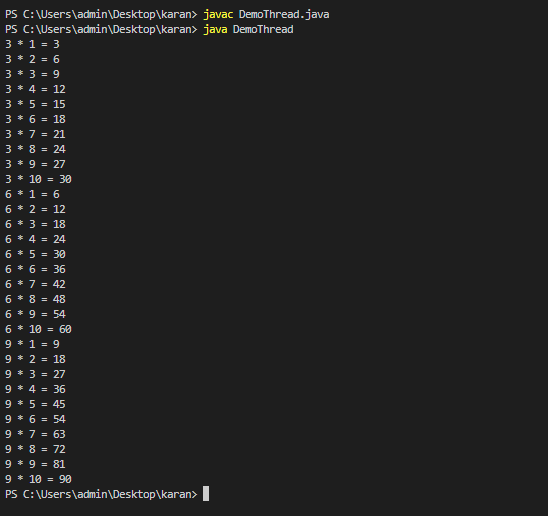
Thread3 t3 = new Thread3();

t3.start();

}

}

**Output:**

****

**Conclusion:**

**Web Servers:** Multithreading enables servers to handle multiple client requests simultaneously, improving response times and resource utilization.

**Real-time Data Processing:** In applications like stock trading, multithreading allows for real-time analysis of data streams, ensuring timely execution of trades without delays in processing.