Task :6

Threading

Date: 17-10-2025 &

19-10-2025

Write a program MulThreads that creates 2 threads — one thread with the name Marimba and the other thread named Jini. Each thread should display its respective name and execute after the gap of 500 ms. Each thread should also display a number indicating the number of times it got a chance to execute.

Hint: use setName() method to set the name of the thread.

Aim:

To write a Java program MulThreads that creates two threads — Marimba and Jini. Each thread displays its name and execution count with a delay of 500 milliseconds between each execution.

Algorithm:

1. Class Name: MulThreads – to demonstrate multithreading.
2. Method Name:
    - run() – defines the task for each thread to execute.
    - main() – creates and starts two threads.
3. Function:
    - Use setName() to assign thread names.
    - Each thread prints its name and execution count (looped).
    - Apply Thread.sleep(500) for a 500 ms delay between executions.

Program:

```
class MulThreads extends Thread {

  // Constructor
  public MulThreads(String name) {
    setName(name); // Setting thread name
  }

  // Thread task
  public void run() {
    for (int i = 1; i <= 5; i++) {
      System.out.println(getName() + " executing... Count: " + i);
      try {
        Thread.sleep(500); // 500 ms delay
```

```
            } catch (InterruptedException e) {
                System.out.println(getName() + " interrupted.");
            }
        }
        System.out.println(getName() + " finished execution.\n");
    }

    // Main method
    public static void main(String[] args) {
        // Create two thread objects
        MulThreads t1 = new MulThreads("Marimba");
        MulThreads t2 = new MulThreads("Jini");

        // Start the threads
        t1.start();
        t2.start();
    }
}
```

Sample Output:

```
Marimba executing... Count: 1
Jini executing... Count: 1
Marimba executing... Count: 2
Jini executing... Count: 2
Marimba executing... Count: 3
Jini executing... Count: 3
Marimba executing... Count: 4
Jini executing... Count: 4
Marimba executing... Count: 5
Jini executing... Count: 5
Marimba finished execution.
Jini finished execution.
```

(The exact order may vary slightly because threads run concurrently.)

Result:

Thus, a Java program was successfully developed to demonstrate multithreading, where two threads (Marimba and Jini) run concurrently, display their names and execution counts with a 500 ms delay between each execution.