LAB # 3

Introduction to Concurrency

Lab Task:-

1. Implement the following program on eclipse IDE and answer the following questions:

• How many threads are running?

• How many tasks are running?

• If more tasks are added than what will be the impact on number of threads?

• Explain the flow of program:

```
class Main extends Thread{
       public void run(){
         System.out.println("task one");
       }
       public static void main(String args[]){
        Main t1=new Main();
        Main t2=new Main();
        Main t3=new Main();
        t1.start();
        t2.start();
        t3.start();
       }
      }
```

➢ How many threads are running?

- You have **3 threads** explicitly created:
  - `t1`
  - `t2`
  - `t3`

Each of them is a separate instance of the `Main` class that extends `Thread`.

➢ How many tasks are running?

In this program, each thread executes the same `run()` method — printing `"task one"`.
Each thread performs one independent task.

- So, **Number of tasks = 3**, one per thread (`t1`, `t2`, `t3`).

➢ If more tasks are added, what will be the impact on number of threads?

That depends on **how** you add tasks.

Case 1: You add more `Thread` objects (like `t4`, `t5`, etc.)

```
Main t4 = new Main();
t4.start();
```

Each new `Thread` object you start creates a **new thread** in memory.
So the **number of threads will increase linearly** with the number of tasks.

 Example:

- 10 tasks → 10 threads (+ the main thread)

Case 2: You use the same thread to do multiple tasks

If you modify the code to make a single thread handle multiple tasks (for example, by looping inside `run()`), then:

- The number of threads remains the same.
- Each thread just executes more work sequentially.

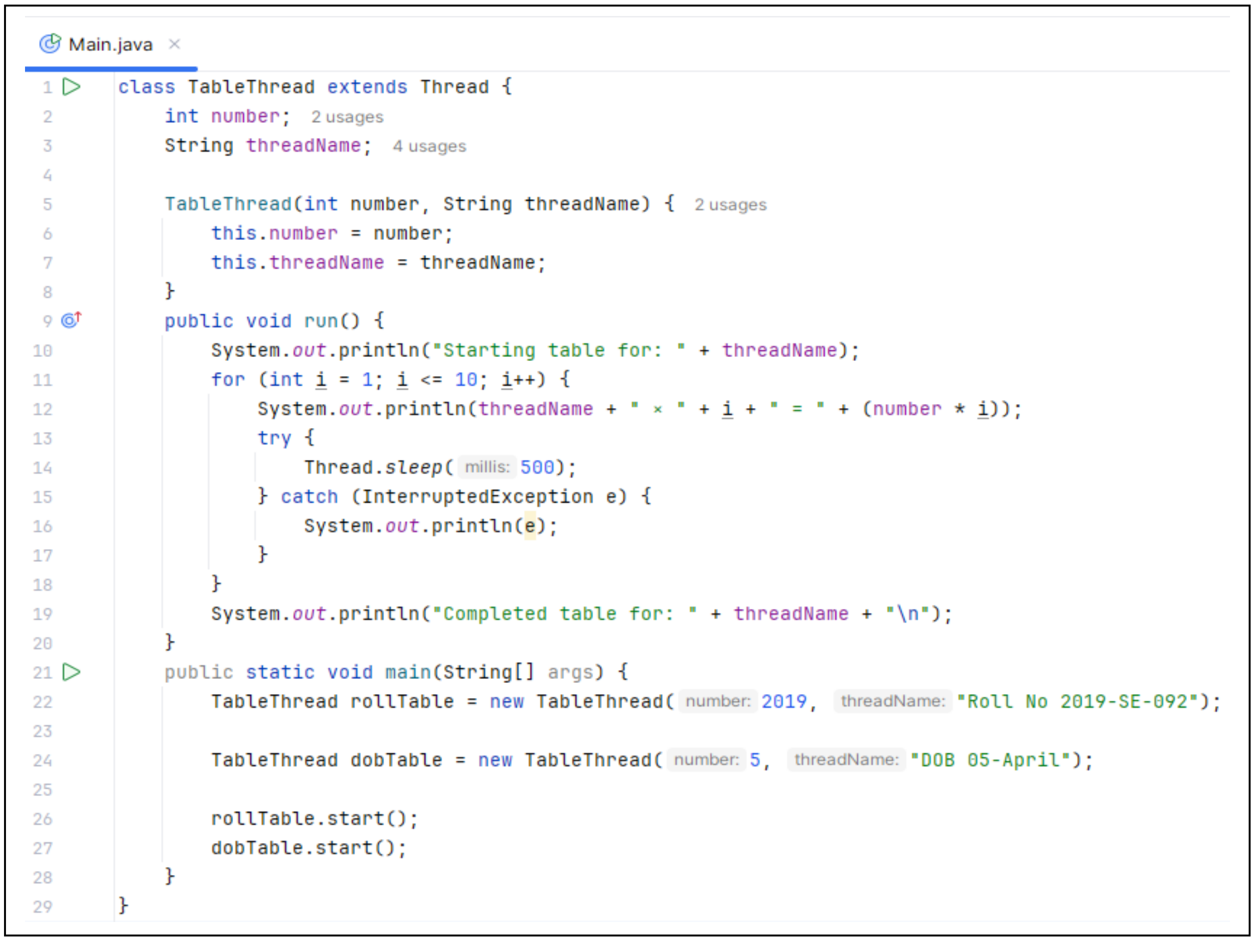
➢ Explain the flow of the program

Let's break it down:

1. **Program starts**:
   - The JVM starts the `main` thread, which executes `main()`.
2. **Thread objects created**:
   - `t1`, `t2`, and `t3` are created, each an instance of the `Main` class.
   - They are **not running yet**; they're just in the **"New"** state.
3. **Threads started**:
   - Calling `t1.start()`, `t2.start()`, `t3.start()` tells the JVM to create **three new call stacks** — one for each thread.
   - Each new thread moves from **"New"** → **"Runnable"** and then **"Running"** when the scheduler picks it.
4. **Run method executes**:
   - The JVM calls `run()` on each thread **concurrently**.
   - Each prints `"task one"` to the console.
   - The order of printing is **not guaranteed** (depends on thread scheduling).
5. **Threads complete**:
   - After printing, each thread finishes execution and enters the **"Terminated"** state.
   - The main thread may finish earlier or later depending on timing.

2. With the help of threading print two tables concurrently, print one table number of student roll number e.g. 2019-SE-092 and second number should be date of birth e.g. 05-April.

Code:-



Main.java

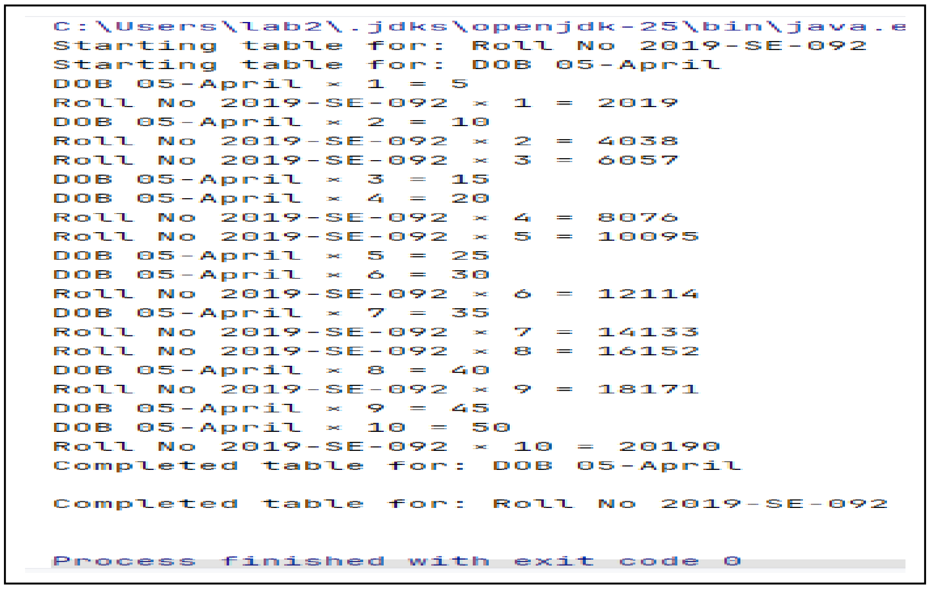
```java
class TableThread extends Thread {
    int number;  2 usages
    String threadName;  4 usages

    TableThread(int number, String threadName) {  2 usages
        this.number = number;
        this.threadName = threadName;
    }
    public void run() {
        System.out.println("Starting table for: " + threadName);
        for (int i = 1; i <= 10; i++) {
            System.out.println(threadName + " × " + i + " = " + (number * i));
            try {
                Thread.sleep( millis: 500);
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
        System.out.println("Completed table for: " + threadName + "\n");
    }
    public static void main(String[] args) {
        TableThread rollTable = new TableThread( number: 2019,  threadName: "Roll No 2019-SE-092");

        TableThread dobTable = new TableThread( number: 5,  threadName: "DOB 05-April");

        rollTable.start();
        dobTable.start();
    }
}
```

Output:-



```
C:\Users\lab2\.jdks\openjdk-25\bin\java.e
Starting table for: Roll No 2019-SE-092
Starting table for: DOB 05-April
DOB 05-April × 1 = 5
Roll No 2019-SE-092 × 1 = 2019
DOB 05-April × 2 = 10
Roll No 2019-SE-092 × 2 = 4038
Roll No 2019-SE-092 × 3 = 6057
DOB 05-April × 3 = 15
DOB 05-April × 4 = 20
Roll No 2019-SE-092 × 4 = 8076
Roll No 2019-SE-092 × 5 = 10095
DOB 05-April × 5 = 25
DOB 05-April × 6 = 30
Roll No 2019-SE-092 × 6 = 12114
DOB 05-April × 7 = 35
Roll No 2019-SE-092 × 7 = 14133
Roll No 2019-SE-092 × 8 = 16152
DOB 05-April × 8 = 40
Roll No 2019-SE-092 × 9 = 18171
DOB 05-April × 9 = 45
DOB 05-April × 10 = 50
Roll No 2019-SE-092 × 10 = 20190
Completed table for: DOB 05-April

Completed table for: Roll No 2019-SE-092


Process finished with exit code 0
```