



**Vidyavardhini's College of Engineering and Technology**

**Department of Artificial Intelligence & Data Science**

---

Experiment No. 10
Implement program on Multithreading
Date of Performance:
Date of Submission:



**Aim:** Implement program on Multithreading

**Objective:**

**Theory:**

**Multithreading in Java** is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Java provides **Thread class** to achieve thread programming. Thread class provides constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

**Thread class:**

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

### 1) Java Thread Example by extending Thread class

**FileName:** Multi.java

```
class Multi extends Thread{
    public void run(){
        System.out.println("thread is running...");
    }
    public static void main(String args[]){
        Multi t1=new Multi();
        t1.start();
    }
}
```



### Output:

```
thread is running...
```

### 2) Java Thread Example by implementing Runnable interface

**FileName:** Multi3.java

```
class Multi3 implements Runnable{
    public void run(){
        System.out.println("thread is running...");
    }

    public static void main(String args[]){
        Multi3 m1=new Multi3();
        Thread t1=new Thread(m1); // Using the constructor Thread(Runnable r)
        t1.start();
    }
}
```

### Output:

```
thread is running...
```

### Code:

```
class Multi2 implements Runnable{
    public void run()
    {
        int a=5;
        int b=7;
        int c=a+b;
        System.out.println("Addition :"+c);
    }

    public static void main(String args[]){
        Multi2 m1=new Multi2();
        Thread t1=new Thread(m1);
        t1.start();
    }
}
```



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\swaru\Desktop\java>javac Multithreading.java

C:\Users\swaru\Desktop\java>java Multithreading.java
Addition : 69

C:\Users\swaru\Desktop\java>
```

### Conclusion:

In the world of Java, multithreading finds its support through the Thread class and the Runnable interface. These components allow you to create and control threads either by extending the Thread class or implementing the Runnable interface. Java's multithreading capabilities encompass synchronization, thread administration, and various thread states, all of which facilitate the simultaneous execution of tasks. Multithreading stands as a foundational feature that optimizes resource utilization, enhances application responsiveness, and boosts performance in scenarios where multiple tasks need to be managed concurrently.