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:

Program1 :-

```
public class  multi1 extends
Thread
{
 public void run()
 {
   System.out.println("Thread is Running.....");
 }
 public static void main(String arhs[])
 {
        multi1 t1=new multi1();
        t1.start();
 }
}
}
```

Program2:-

```
public class multi2 implements Runnable
```

```
{
 public void run()
 {
  System.out.println("Thread is Running.....");
 }
 public static void main(String args [])
 {
  multi2 r1=new multi2();     Thread
t1=new Thread(r1);
        t1.start();
 }
}
```

Conclusion:

Comment on how multithreading is supported in JAVA.

Answer: Java supports multithreading through its robust `Thread` class and associated APIs. It provides thread lifecycle management, synchronization mechanisms, thread prioritization, safety measures for concurrent data access, and tools for intercommunication between threads. Java's `Executor` framework simplifies thread pooling, and the `java.util.concurrent` package offers high-level utilities for concurrent programming, making it a versatile platform for building efficient and responsive multithreaded applications.