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:

```
public class Multithreading {
    public static void main(String[] args) {
        int n = 8; // Number of threads
        for (int i = 0; i < n; i++)
{
            MultithreadingDemo object = new MultithreadingDemo();
            object.start();
        }
    }
}
class MultithreadingDemo extends Thread {
    public void run() {
        try {
            // Your code that might throw exceptions
            System.out.println("Thread " + Thread.currentThread() + " is running");
        } catch (Throwable t) {
            // Handle the exception
            System.out.println("Exception is caught: " + t.getMessage());
        }
    }
}
```

Output:

```
C:\Users\yedu0\OneDrive\Documents\java>javac Multithreading.java

C:\Users\yedu0\OneDrive\Documents\java>java Multithreading.java
Thread Thread[#24,Thread-1,5,main] is running
Thread Thread[#30,Thread-7,5,main] is running
Thread Thread[#28,Thread-5,5,main] is running
Thread Thread[#29,Thread-6,5,main] is running
Thread Thread[#27,Thread-4,5,main] is running
Thread Thread[#26,Thread-3,5,main] is running
Thread Thread[#25,Thread-2,5,main] is running
Thread Thread[#23,Thread-0,5,main] is running
```

Conclusion:

Comment on how multithreading is supported in JAVA.

Java provides built-in support for multithreading, allowing developers to create applications that can perform multiple tasks concurrently. Multithreading in Java is primarily achieved using the java.lang.Thread class and the java.lang.Runnable interface. Here are some key features and components of multithreading in Java:

Thread Class and Runnable Interface:
The Thread class in Java represents a thread of execution. Threads can be created by either extending the Thread class or implementing the Runnable interface. The Runnable interface is preferred when creating threads, as it allows for better separation of concerns.