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: To implement a multithreading program in Java and understand the basic concepts of multithreading, including the creation and execution of threads, using both the extension of the Thread class and the implementation of the Runnable interface.

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:

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

OUTPUT:

```
C:\Users\admin\Desktop\ketan>javac multi1.java

C:\Users\admin\Desktop\ketan>java multi1.java
Thread is Running.....

C:\Users\admin\Desktop\ketan>
```

2. 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();
}
}
OUTPUT:

```
C:\Users\admin\Desktop\ketan>javac multi2.java

C:\Users\admin\Desktop\ketan>java multi2.java
Thread is Running.....

C:\Users\admin\Desktop\ketan>
```

Conclusion:

Comment on how multithreading is supported in JAVA.

Multithreading Support in Java:
- Java offers robust support for multithreading.
- It provides the `Thread` class for creating and managing threads.
- Threads can be created by extending `Thread` or implementing `Runnable`.

Key Concepts:
- Java manages thread states and provides synchronization for safe multithreading.
- It's widely used in applications like games, web servers, and concurrent data processing.

In summary, Java's multithreading support simplifies concurrent programming for enhanced application performance.