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 MultiThread 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[]){
MultiThread m1=new MultiThread();
Thread t1=new Thread(m1);
t1.start();
}
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac MultiThread.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java MultiThread.java
Addition : 12
```

Conclusion:

Comment on how multithreading is supported in JAVA.

eading is supported in Java by providing various classes and interfaces that allow us to create and manage multiple threads within a single process. Some of the key features of multithreading in Java are:

- Java provides the Thread class and the Runnable interface to define and implement the logic of a thread. A thread can be created by either extending the Thread class or implementing the Runnable interface, and then overriding the run () method.
- Java provides the start () method to launch a new thread, which invokes the run () method internally. The start () method creates a new call stack for the thread, and allows the JVM to schedule the thread execution.
- Java provides various methods and keywords to control and synchronize the threads, such as join (), sleep (), wait (), notify (), synchronized, volatile, etc. These methods and keywords help to avoid race conditions, deadlocks, and memory inconsistencies among the threads.
- Java supports multiple inheritance of interfaces, which allows a class to implement multiple interfaces and inherit the abstract methods of those interfaces. This feature enables a class to implement the Runnable interface and extend another class at the same time, which is not possible with the Thread class.