Government College of Engineering, Jalgaon
(An Autonomous Institute of Government of Maharashtra)

Name :	**PRN :**
Subject : CO310U (Application programming Lab)	**Sem :** V(Odd)
Class : T.Y. B.Tech	**Academic Year :** 2024-25
Date of Performance :	**Date of Completion :**

Practical No : 14

Aim: Write a java program that creates threads by extending the Thread class .First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds ,(Repeat the same by implementing Runnable)

Required Software: OpenJDK version "1.8.0_131"
OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-2ubuntu1.16.04.3-b11)
OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)

Java Compiler Version - JAVAC 1.8.0_131

Theory:

Threading in java:

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 rather than multiprocessing because threads use a shared memory area. They don't allocate separate memory areas so saves memory, and context-switching between the threads takes less time than process.Java Multithreading is mostly used in games, animation, etc.

A thread is a lightweight subprocess, the smallest unit of processing. It is a separate path of execution.Threads are independent. If there occurs an exception in one thread, it doesn't affect other threads. It uses a shared memory area.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.

Java Thread Methods

S. N.	**Modifier and Type**	**Method**	**Description**
1)	void	start()	It is used to start the execution of the thread.

2)	void	run()	It is used to do an action for a thread.
3)	static void	sleep()	It sleeps a thread for the specified amount of time.
4)	static Thread	currentThread()	It returns a reference to the currently executing thread object.
5)	void	join()	It waits for a thread to die.
6)	int	getPriority()	It returns the priority of the thread.
7)	void	setPriority()	It changes the priority of the thread.
8)	String	getName()	It returns the name of the thread.
9)	void	setName()	It changes the name of the thread.
10)	long	getId()	It returns the id of the thread.
11)	boolean	isAlive()	It tests if the thread is alive.

12)	static void	yield()	It causes the currently executing thread object to pause and allow other threads to execute temporarily.
13)	void	suspend()	It is used to suspend the thread.
14)	void	resume()	It is used to resume the suspended thread.
15)	void	stop()	It is used to stop the thread.
16)	void	destroy()	It is used to destroy the thread group and all of its subgroups.
17)	boolean	isDaemon()	It tests if the thread is a daemon thread.
18)	void	setDaemon()	It marks the thread as daemon or user thread.
19)	void	interrupt()	It interrupts the thread.
20)	boolean	isinterrupted()	It tests whether the thread has been interrupted.
21)	static boolean	interrupted()	It tests whether the current thread has been interrupted.

22)	static int	activeCount()	It returns the number of active threads in the current thread's thread group.
23)	void	checkAccess()	It determines if the currently running thread has permission to modify the thread.
24)	static boolean	holdLock()	It returns true if and only if the current thread holds the monitor lock on the specified object.
25)	static void	dumpStack()	It is used to print a stack trace of the current thread to the standard error stream.
26)	StackTraceElement[]	getStackTrace()	It returns an array of stack trace elements representing the stack dump of the thread.
27)	static int	enumerate()	It is used to copy every active thread's thread group and its subgroup into the specified array.
28)	Thread.State	getState()	It is used to return the state of the thread.
29)	ThreadGroup	getThreadGroup()	It is used to return the thread group to which this thread belongs

30)	String	toString()	It is used to return a string representation of this thread, including the thread's name, priority, and thread group.
31)	void	notify()	It is used to give the notification for only one thread which is waiting for a particular object.
32)	void	notifyAll()	It is used to give the notification to all waiting threads of a particular object.
33)	void	setContextClassLoader()	It sets the context ClassLoader for the Thread.
34)	ClassLoader	getContextClassLoader()	It returns the context ClassLoader for the thread.
35)	static Thread.UncaughtExceptionHandler	getDefaultUncaughtExceptionHandler()	It returns the default handler invoked when a thread abruptly terminates due to an uncaught exception.
36)	static void	setDefaultUncaughtExceptionHandler()	It sets the default handler invoked when a thread abruptly terminates due to an uncaught exception.

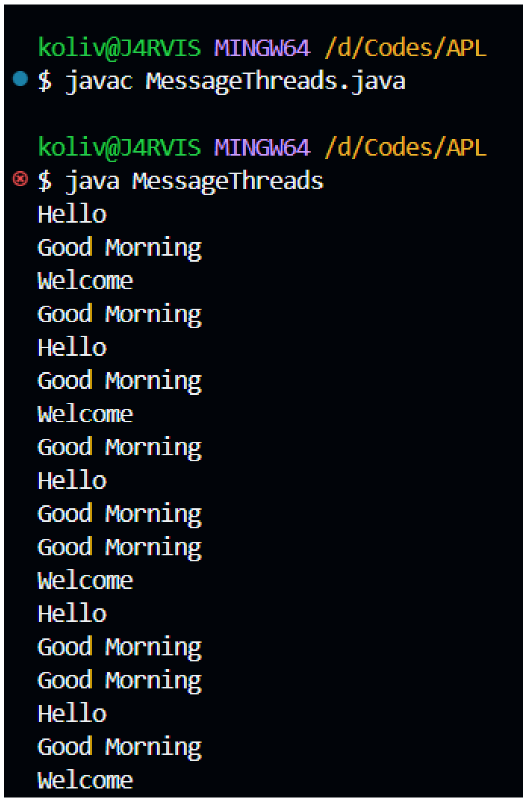
Conclusion:

Name & sign of Teacher

Program :

```
class GoodMorningThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Good Morning");
                Thread.sleep(1000);  // 1 second
            }
        } catch (InterruptedException e) {
            System.out.println("GoodMorningThread interrupted");
        }
    }
}

class HelloThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Hello");
                Thread.sleep(2000);  // 2 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("HelloThread interrupted");
        }
    }
}

class WelcomeThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Welcome");
                Thread.sleep(3000);  // 3 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("WelcomeThread interrupted");
        }
    }
}

public class MessageThreads {
    public static void main(String[] args) {
        GoodMorningThread t1 = new GoodMorningThread();
        HelloThread t2 = new HelloThread();
        WelcomeThread t3 = new WelcomeThread();

        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output :

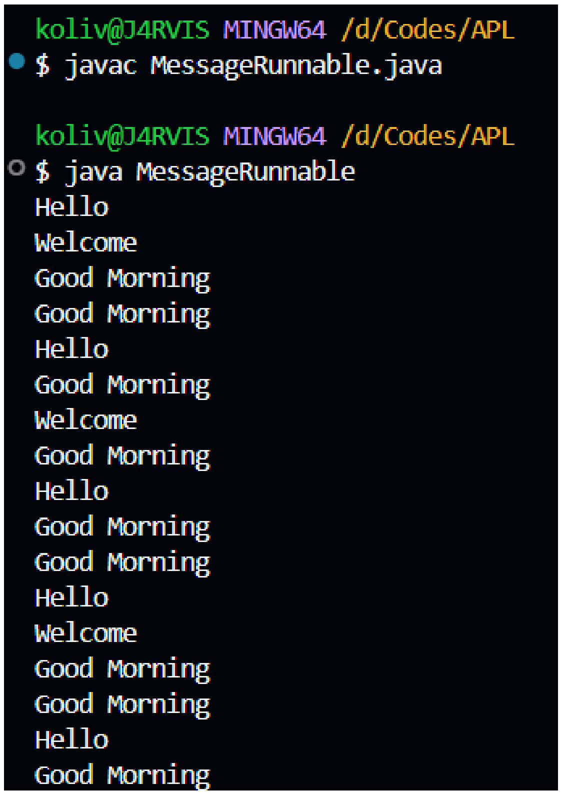


```
koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac MessageThreads.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java MessageThreads
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
```

Program : (with Runnable Interface)

```
class GoodMorningRunnable implements Runnable {
    public void run() {
        try {
            while (true) {
                System.out.println("Good Morning");
                Thread.sleep(1000);  // 1 second
            }
        } catch (InterruptedException e) {
            System.out.println("GoodMorningRunnable interrupted");
        }
    }
}

class HelloRunnable implements Runnable {
    public void run() {
        try {
            while (true) {
                System.out.println("Hello");
                Thread.sleep(2000);  // 2 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("HelloRunnable interrupted");
        }
    }
}

class WelcomeRunnable implements Runnable {
    public void run() {
        try {
            while (true) {
                System.out.println("Welcome");
                Thread.sleep(3000);  // 3 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("WelcomeRunnable interrupted");
        }
    }
}

public class MessageRunnable {
    public static void main(String[] args) {
        Thread t1 = new Thread(new GoodMorningRunnable());
        Thread t2 = new Thread(new HelloRunnable());
        Thread t3 = new Thread(new WelcomeRunnable());

        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output :



```
koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac MessageRunnable.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java MessageRunnable
Hello
Welcome
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Hello
Welcome
Good Morning
Good Morning
Hello
Good Morning
```