R.Anusha(AF0311774)

Q1. Write a program that creates two threads. Each thread should print its thread ID (TID) and a unique message to the console. Ensure that the output from both threads is interleaved.

```
package Lab7;

public class InterleavedThread {

public static void main(String[] args) {

Thread thread1 = new Thread(new MessagePrinter(1, "Hello from Thread-1"));

Thread thread2 = new Thread(new MessagePrinter(2, "Greetings from Thread-2"));

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

}catch (InterruptedException e)

{
 e.printStackTrace();

}

System.out.println("Both threads have finished.");

}

}
```

```
class MessagePrinter implements Runnable {

private int threadNum;
private String message;

public MessagePrinter(int threadNum, String message) {

this.threadNum = threadNum;

this.message = message;

}

@Override
public void run()

{

for (int i = 0; i < 5; i++) {

System.out.println("Thread-"      +      threadNum      +      "      (TID-"      +
Thread.currentThread().~~getId()~~ + "): " + message);

try {

Thread.sleep(500);

} catch (InterruptedException e){

e.printStackTrace();

}

}

}

}
```

Output:

```
Thread-1 (TID-21): Hello from Thread-1
Thread-2 (TID-22): Greetings from Thread-2
Thread-1 (TID-21): Hello from Thread-1
Thread-2 (TID-22): Greetings from Thread-2
Thread-2 (TID-22): Greetings from Thread-2
Thread-1 (TID-21): Hello from Thread-1
Thread-1 (TID-21): Hello from Thread-1
Thread-2 (TID-22): Greetings from Thread-2
Thread-1 (TID-21): Hello from Thread-1
Thread-2 (TID-22): Greetings from Thread-2
Both threads have finished.
```

Q2. Write a program that creates multiple threads with different priorities. Observe how the operating system schedules threads with different priorities and explain the results.

```
package Lab7;

public class ThreadPriority1 implements Runnable{

public void run() {

for (int i = 0; i < 5; i++) {

System.out.println(Thread.currentThread().getName() + ": Priority " + Thread.currentThread().getPriority() + ", Count: " + i);

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}


package Lab7;

public class Priorities {

public static void main(String[] args) {

Thread Thread1 = new Thread(new ThreadPriority1(), "Low Priority Thread");
Thread Thread2 = new Thread(new ThreadPriority1(), "Normal Priority Thread");

Thread Thread3 = new Thread(new ThreadPriority1(), "High Priority Thread");

// Set thread priorities

Thread1.setPriority(Thread.MIN_PRIORITY);

Thread2.setPriority(Thread.NORM_PRIORITY);

Thread3.setPriority(Thread.MAX_PRIORITY);

Thread1.start();

Thread2.start();

Thread3.start();

}

}
```

Output:

```
Normal Priority Thread: Priority 5, Count: 0
Low Priority Thread: Priority 1, Count: 0
High Priority Thread: Priority 10, Count: 0
High Priority Thread: Priority 10, Count: 1
Low Priority Thread: Priority 1, Count: 1
Normal Priority Thread: Priority 5, Count: 1
High Priority Thread: Priority 10, Count: 2
Normal Priority Thread: Priority 5, Count: 2
Low Priority Thread: Priority 1, Count: 2
High Priority Thread: Priority 10, Count: 3
Normal Priority Thread: Priority 5, Count: 3
Low Priority Thread: Priority 1, Count: 3
Normal Priority Thread: Priority 5, Count: 4
High Priority Thread: Priority 10, Count: 4
Low Priority Thread: Priority 1, Count: 4
```

Q3. Write a Java program that creates two threads and prints "Thread A" from the first thread and "Thread B" from the second thread. Make sure both threads run concurrently.

```
package Lab7;

public class ConcurrentThread {

public static void main(String[] args) {

Thread threadA = new Thread(() -> {

for (int i = 0; i < 5; i++) {

System.out.println("Thread A");

 try {


Thread.sleep(500); // Sleep for 0.5 seconds

} catch (InterruptedException e) {

 e.printStackTrace();

}

}

});

Thread threadB = new Thread(() -> {

for (int i = 0; i < 5; i++) {

System.out.println("Thread B");

{

Thread.sleep(500); // Sleep for 0.5 seconds

} catch (InterruptedException e) {
 e.printStackTrace();

}

}

});

threadA.start();

threadB.start();

try {

threadA.join();

threadB.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Both threads have finished.");

}

}
```

Output:

```
Thread A
Thread B
Thread A
Thread B
Thread A
Thread B
Thread A
Thread B
Thread B
Thread A
Both threads have finished.
```