1.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 nikki.com;

public class Interleaved_thread {

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

2. 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 nikki.com;

public class Priority_demo {

public static void main(String[] args) {
Thread t1 = new Thread(new MyRunnable(), "Thread 1");
Thread t2 = new Thread(new MyRunnable(), "Thread 2");
Thread t3 = new Thread(new MyRunnable(), "Thread 3");
t1.setPriority(Thread.MIN_PRIORITY);
t2.setPriority(Thread.NORM_PRIORITY);
t3.setPriority(Thread.MAX_PRIORITY);
t1.start();
t2.start();
t3.start();
}
static class MyRunnable implements Runnable {
public void run() {
String name = Thread.currentThread().getName();
int priority = Thread.currentThread().getPriority();
for (int i = 0; i < 5; i++) {
System.out.println(name + " running with priority " +
priority);
try {
Thread.sleep(100); // Sleep for 100 milliseconds
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
}
}
```

OUTPUT:

```
Thread 2 running with priority 5
Thread 3 running with priority 10
Thread 1 running with priority 1
Thread 3 running with priority 10
Thread 1 running with priority 1
Thread 2 running with priority 5
Thread 3 running with priority 10
Thread 2 running with priority 5
Thread 1 running with priority 1
Thread 3 running with priority 10
Thread 2 running with priority 5
Thread 1 running with priority 1
Thread 3 running with priority 10
Thread 2 running with priority 5
Thread 1 running with priority 1
```

3. 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 nikki.com;

public class ThreadA implements Runnable{
public void run( )
{
for (int i = 1; i <= 5; i++)
{
System.out.println("Thread A");
try
{
Thread.sleep(1000); // Pause for 1 second
} catch (InterruptedException e)
{
e.printStackTrace();
}
}
}
}
package nikki.com;

public class ThreadB implements Runnable {
public void run( ) {
for (int i = 1; i <= 5; i++) {
System.out.println("Thread B");
try {
Thread.sleep(1000); // Pause for 1 second
} catch (InterruptedException e) {
e.printStackTrace();
}
}
System.out.println("Both threads have finished.");
}
}
package nikki.com;
```

```
public class Concurrent_threads {
public static void main(String[ ] args) {
Thread threadA = new Thread(new ThreadA());
Thread threadB = new Thread(new ThreadB());
threadA.start();
threadB.start();
}

}
```

OUTPUT:

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