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 ?

ANS:

```
public class InterleavedThreadExample {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                printMessage("Thread 1 says hello!");
            }
        });

        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                printMessage("Thread 2 says hi!");
            }
        });

        thread1.start();
        thread2.start();
    }

    public static void printMessage(String message) {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getId() + " " + message);
            try {
                Thread.sleep(500); // Just to simulate a longer-running process
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

OUTPUT:

```
11 Thread 2 says hi!
10 Thread 1 says hello!
11 Thread 2 says hi!
10 Thread 1 says hello!
11 Thread 2 says hi!
10 Thread 1 says hello!
11 Thread 2 says hi!
10 Thread 1 says hello!
10 Thread 1 says hello!
11 Thread 2 says hi!
```

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 ?

ANS:

```
public class ThreadPriorityExample {
    public static void main(String[] args) {
        // Create threads with different priorities
        Thread thread1 = new Thread(new WorkerThread("Thread 1"), "Thread 1");
        Thread thread2 = new Thread(new WorkerThread("Thread 2"), "Thread 2");
        Thread thread3 = new Thread(new WorkerThread("Thread 3"), "Thread 3");

        // Set different priorities for threads
        thread1.setPriority(Thread.MIN_PRIORITY);   // Minimum priority (1)
        thread2.setPriority(Thread.NORM_PRIORITY);  // Default priority (5)
        thread3.setPriority(Thread.MAX_PRIORITY);   // Maximum priority (10)

        // Start the threads
        thread1.start();
        thread2.start();
        thread3.start();
    }

    static class WorkerThread implements Runnable {
        private String name;

        public WorkerThread(String name) {
            this.name = name;
        }

        @Override
        public void run() {
            for (int i = 1; i <= 5; i++) {
                System.out.println(name + " is running (" + i + "/5)");
                try {
                    Thread.sleep(100); // Just to simulate some work
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

OUTPUT:

```
Thread 3 is running (1/5)
Thread 1 is running (1/5)
Thread 2 is running (1/5)
Thread 3 is running (2/5)
Thread 1 is running (2/5)
Thread 2 is running (2/5)
Thread 3 is running (3/5)
Thread 1 is running (3/5)
Thread 2 is running (3/5)
Thread 1 is running (4/5)
Thread 3 is running (4/5)
Thread 2 is running (4/5)
Thread 1 is running (5/5)
Thread 2 is running (5/5)
Thread 3 is running (5/5)
```

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?

ANS:

```java
public class ConcurrentThreadsExample {
    public static void main(String[] args) {
        Thread threadA = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 1; i <= 5; i++) {
                    System.out.println("Thread A");
                    try {
                        Thread.sleep(100); // Just to simulate some work
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        });

        Thread threadB = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 1; i <= 5; i++) {
                    System.out.println("Thread B");
                    try {
                        Thread.sleep(100); // Just to simulate some work
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        });

        threadA.start();
        threadB.start();
    }
}
```

OUTPUT:

```
Thread A
Thread B
Thread A
Thread B
Thread A
Thread B
Thread A
Thread B
Thread B
Thread A
```