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.**

```java
package multiprocess.com;
public class TwoThreads {
public static void main(String[] args) {
Thread thread1 = new Thread(new MyRunnable("Thread 1"));
Thread thread2 = new Thread(new MyRunnable("Thread 2"));
thread1.start();
thread2.start();
}
static class MyRunnable implements Runnable {
private String threadName;
public MyRunnable(String threadName) {
this.threadName = threadName;
}
public void run() {
for (int i = 1; i <= 5; i++) {
System.out.println(threadName + " (TID: " + Thread.currentThread().~~getId~~() + "): Message " + i);
try {
Thread.sleep(1000); // Pause for 1 second
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
}
}
```

Output:

```
Thread 1 (TID: 21): Message 1

Thread 2 (TID: 22): Message 1

Thread 2 (TID: 22): Message 2

Thread 1 (TID: 21): Message 2

Thread 1 (TID: 21): Message 3

Thread 2 (TID: 22): Message 3

Thread 1 (TID: 21): Message 4

Thread 2 (TID: 22): Message 4

Thread 1 (TID: 21): Message 5
Thread 2 (TID: 22): Message 5
```

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.

```java
package multiprocess.com;

public class PriorityThread {

public static void main(String[] args) {

Thread lowPriorityThread = new Thread(new MyRunnable("Low
Priority Thread"));

Thread normalPriorityThread = new Thread(new
MyRunnable("Normal Priority Thread"));

Thread highPriorityThread = new Thread (new MyRunnable("High
Priority Thread"));

// Set thread priorities

lowPriorityThread.setPriority(Thread.MIN_PRIORITY);

normalPriorityThread.setPriority(Thread.NORM_PRIORITY);

highPriorityThread.setPriority(Thread.MAX_PRIORITY);

// Start the threads

lowPriorityThread.start();

normalPriorityThread.start();

highPriorityThread.start();

}

static class MyRunnable implements Runnable {

private String threadName;

public MyRunnable(String threadName) {

this.threadName = threadName;
}

@Override

public void run() {

for (int i = 1; i <= 5; i++) {

System.out.println(threadName + " (Priority " +
Thread.currentThread().getPriority() + "): Message " + i);

try {

Thread.sleep(1000); // Pause for 1 second

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}
```

Output :

```
Low Priority Thread (Priority 1): Message 1

Normal Priority Thread (Priority 5): Message 1

High Priority Thread (Priority 10): Message 1

High Priority Thread (Priority 10): Message 2

Normal Priority Thread (Priority 5): Message 2

Low Priority Thread (Priority 1): Message 2

High Priority Thread (Priority 10): Message 3

Normal Priority Thread (Priority 5): Message 3

Low Priority Thread (Priority 1): Message 3

High Priority Thread (Priority 10): Message 4
```

```
Normal Priority Thread (Priority 5): Message 4

Low Priority Thread (Priority 1): Message 4

High Priority Thread (Priority 10): Message 5

Normal Priority Thread (Priority 5): Message 5

Low Priority Thread (Priority 1): Message 5
```

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.

```java
package multiprocess.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 multiprocess.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();

}

}

}

}

package multiprocess.com;

public class ThreadAB {

public static void main(String[] args) {

Thread threadA = new Thread(new ThreadA());

Thread threadB = new Thread(new ThreadB());

threadA.start();

threadB.start();

}
```

```
}
```

Output :

```
Thread A

Thread B

Thread A

Thread B

Thread B

Thread A

Thread A

Thread B

Thread B

Thread A
```