Problem Statement: Print Numbers Using `extends Thread`

1. Create a class that extends `Thread` and overrides the `run()` method to print numbers from 1 to 10. Create an instance of this class and start the thread.

Problem Statement: Print Numbers Using `implements Runnable`

2. Create a class that implements `Runnable` and overrides the `run()` method to print numbers from 1 to 10. Create an instance of this class, pass it to a `Thread` object, and start the thread.

Problem Statement: Sum of Array Elements Using `extends Thread`

3. Create a class that extends `Thread` to calculate the sum of elements in an array. Initialize the array in the constructor and override the `run()` method to compute the sum and print it.

Problem Statement: Sum of Array Elements Using `implements Runnable`

4. Create a class that implements `Runnable` to calculate the sum of elements in an array. Initialize the array in the constructor and override the `run()` method to compute the sum and print it.

Problem Statement: Using `sleep()` to Pause Execution

5. Create a thread that prints numbers from 1 to 5, pausing for 1 second between each number using `Thread.sleep()`.

Problem Statement: Using `join()` to Ensure Thread Completion

6. Create two threads: one that prints even numbers from 2 to 10 and another that prints odd numbers from 1 to 9. Use the `join()` method to ensure the main thread waits for both threads to complete before printing "Done".

Problem Statement: Using `yield()` to Yield Execution

7. Create a thread that prints "Hello" 5 times and another thread that prints "World" 5 times. Use the `yield()` method to allow other threads to execute between each print statement.

Problem Statement: Using `join()` for Sequential Execution

8. Create three threads that print numbers from 1 to 5. Use the `join()` method to ensure that the threads execute sequentially (i.e., the second thread starts only after the first thread finishes, and the third thread starts only after the second thread finishes).

Problem Statement: Using `join()` for Cooperative Multithreading

9. Create two threads: one that prints "Ping" and another that prints "Pong". Use the **join**() method to allow the threads to alternate their execution, simulating a ping-pong game.