Exercise 1: The "Reverse" Iterator

Create a class called ReverseArrayEnumerator that implements IEnumerator<int>. Instead of starting at index 0, it should start at the **last index** of an array and move backward to the beginning.

- **Goal:** Understand how to manipulate the _position variable in MoveNext().
- **Success Criteria:** A foreach loop over your collection prints an array [1, 2, 3] as 3, 2, 1.

Exercise 2: The "Skipper" (Every Other Item)

Create a custom IEnumerable called EveryOther<T> that takes a List<T> in its constructor. Its enumerator should skip every second item.

- **Goal:** Learn that MoveNext() can move the index by more than just +1.
- **Success Criteria:** Iterating over { "A", "B", "C", "D" } should only output A and C.

Exercise 3: The Infinite Fibonacci Generator

Implement a class FibonacciSequence that implements IEnumerable<int>. Use a manual IEnumerator (no yield return) to generate the Fibonacci sequence infinitely.

- **Goal:** Understand that an IEnumerable doesn't actually need a "list" or "array" behind it—it can calculate values on the fly.
- **Challenge:** Since it's infinite, make sure your MoveNext() always returns true, but be careful not to trigger an infinite loop in Main (use a break after 10 items).

Exercise 4: The "Circular" Buffer

Create a CircularIterator<T> that takes an array. When it reaches the end of the array, MoveNext() should wrap back around to the first element instead of returning false.

- **Goal:** See how to create a "never-ending" cursor.
- **Success Criteria:** If you have an array [Red, Blue], a loop should be able to print Red, Blue, Red, Blue, Red… indefinitely.

Exercise 5: The "Filtering" Enumerator (Manual Where)

Create a class EvenNumbersOnlyEnumerator that wraps an existing IEnumerator<int>. In its MoveNext() method, it should keep calling the underlying enumerator's MoveNext() until it finds an even number or hits the end.

- **Goal:** This is how LINQ's .Where() actually works under the hood. You are "decorating" one enumerator with another.
- **Success Criteria:** Passing [1, 2, 3, 4, 5, 6] to your class results in an iteration that only shows 2, 4, 6.