```
using System.Collections;
using NUnit.Framework;
using UnityEngine;
using UnityEngine.TestTools;

public class EnemyBoundaryTest
{
    // Tracks enemy X-position for a few seconds to find min and max positions
    private IEnumerator TrackX(GameObject go, float seconds, System.Action<float, float>
result)
    {
        float min = float.MaxValue, max = float.MinValue;
        float end = Time.time + seconds;
        while (Time.time < end)
        {
            float x = go.transform.position.x;
            if (x < min) min = x;
            if (x > max) max = x;
            yield return null;
        }
        result(min, max);
    }

    // Runs a patrol boundary test with given start position and offsets
    private IEnumerator RunBoundaryTest(Vector3 startPos, float leftOffset, float rightOffset)
    {
        var enemy = new GameObject("Enemy");
        enemy.transform.position = startPos;
        var controller = enemy.AddComponent<EnemyController>();
        controller.leftOffset = leftOffset;
        controller.rightOffset = rightOffset;
        controller.speed = 5f;

        yield return null; // Wait one frame for Start()

        float leftBound = startPos.x + Mathf.Min(leftOffset, rightOffset);
        float rightBound = startPos.x + Mathf.Max(leftOffset, rightOffset);
        float observedMin = 0, observedMax = 0;

        yield return TrackX(enemy, 5f, (min, max) => { observedMin = min; observedMax = max; });

        float tolerance = 0.05f;
        // Confirms enemy stays inside left and right limits
        Assert.GreaterOrEqual(observedMin, leftBound - tolerance);
```

```
        Assert.LessOrEqual(observedMax, rightBound + tolerance);

        Object.Destroy(enemy);
    }

    [UnityTest]
    // Checks normal patrol movement stays within -3 to +3 range
    public IEnumerator Enemy_StaysWithinBounds_Default() =>
        RunBoundaryTest(Vector3.zero, -3f, 3f);

    [UnityTest]
    // Checks patrol still works when offsets are entered backward
    public IEnumerator Enemy_StaysWithinBounds_InvertedOffsets() =>
        RunBoundaryTest(new Vector3(10f, 0, 0), 6f, -4f);

    [UnityTest]
    // Checks small patrol range near tolerance doesn't break or drift
    public IEnumerator Enemy_StaysWithinBounds_SmallRange() =>
        RunBoundaryTest(new Vector3(-5f, 0, 0), -0.15f, 0.15f);
}
```