Cálculo del número de picos en un array

Un pico es un elemento del array que es mayor que sus vecinos inmediato anterior y posterior. Ejemplo:

```
Array:         [1, 3, 2, 4, 4, 6, 1, 5, 3]
Índices:        0  1  2  3  4  5  6  7  8
Picos:             ↑           ↑     ↑
```

En este caso, los picos están en las posiciones 1 (3), 5 (6) y 7 (5).

```java
public class PicoDetector {

    public static int contarPicos(int[] datos) {
        int contador = 0;

        for (int i = 1; i < datos.length - 1; i++) {
            if (datos[i] > datos[i - 1] && datos[i] > datos[i + 1]) {
                contador++;
            }
        }

        return contador;
    }

    public static void main(String[] args) {
        int[] valores = {1, 3, 2, 4, 4, 6, 1, 5, 3};
        int resultado = contarPicos(valores);
        System.out.println("Número de picos: " + resultado);
    }
}
```

¿Por qué este ejemplo es ideal para usar el debugger?

- Es muy fácil cometer errores al intentar seguir mentalmente qué elementos cumplen la condición.
- El uso del debugger permite:
  - Observar el valor de `i`, `datos[i - 1]`, `datos[i]` y `datos[i + 1]` en cada paso.
  - Ver cuándo y por qué se incrementa `contador`.
- Además, pueden ver cómo algunas condiciones **casi se cumplen pero no** (como los dos cuatros seguidos, que generan duda).

Actividad sugerida

1. Poner un *breakpoint* en la línea del `if`.
2. Ejecutar en modo debug.
3. En cada iteración:

- Observar los tres valores clave.
- Justificar si hay pico o no.
- Ver cuándo se incrementa `contador`.

Preguntas:

- ¿Qué habría pasado si el bucle empezara en 0 o terminara en `datos.length`?
- ¿Qué pasaría si el array tiene elementos iguales consecutivos?
- ¿Cómo adaptarías el algoritmo si quisieras contar también los "valles"?