

Evidencia 03: Caso Palíndromos

Por Gabriel Valenzuela y Gabriel Vergara

Paso 1: Leer y entender el contexto problema

Se necesita saber si el input entregado a un método es o no un *palíndromo*. Sino recuerda o no sabe que es un palíndromo, puede verlo [aquí](#).

Claro que tenemos un pequeño problema, pues el programador a cargo olvidó dejar la versión Java de la solución y sólo tenemos el siguiente código Javascript:

```
function esPalindromo(cadena) {  
  
    let resultado = "";  
  
    resultado = cadena.split("").reverse().join("");  
  
    return cadena === resultado;  
  
}
```

1.1 Discutir y concluir:

- ¿Qué hace el método?
 - El método dice si una cadena de texto es un palíndromo.
- ¿Cómo lo hace?
 - Iguala la cadena original con la misma cadena pero al revés.
- ¿Cómo lo uso?
 - Le tiene que entregar una cadena al método, y este, en cambio regresara un booleano, dependiendo si es un palíndromo o no.

Paso 2: Entender el método JavaScript

2.1 Tras una discusión individual, cada grupo deberá explicar qué hace el método detalladamente.

Este método, primero inicia una variable string llamada "resultado", luego copia a "resultado" la cadena entregada convertida en un array de caracteres, ordenada de forma inversa, y unida en un string. Después, se comparara si "resultado", es igual a la cadena entregada inicialmente, esta comparación entregará un boolean dependiendo del resultado.

2.2 Construya en grupo ahora una versión Java que sea 100% equivalente en funcionalidad (lo bueno y lo malo) al anterior método.

```
public static boolean esPalindromo(String cadena){ no usages new *
    String resultado = "";
    for (int i = 0; i < cadena.length() ; i++) {
        resultado = resultado + cadena.charAt(cadena.length()-i -1);
    }
    return cadena.equals(resultado);
}
```

Paso 3: Ok! Si el método funciona ¿Qué puede malir sal? ;-)

3.2 A partir de su plan de pruebas, diseñe los casos de pruebas unitarias a implementar (aún no codifique nada!!!), considere al menos 5.

1. Entrada: "arriba", Resultado esperado: false
2. Entrada: "rotor", Resultado esperado: true
3. Entrada: "1221221", Resultado esperado: true
4. Entrada: "roto r", Resultado esperado: true
5. Entrada: " casa asac", Resultado esperado: true

3.3 Estando seguros que sus casos de pruebas unitarias son amplios y relevantes, ahora impleméntelos en Java usando JUnit.

```
@Test new *
void notPalindromeTest(){
    assertFalse(checker.esPalindromo( cadena: "arriba"));
}

@Test new *
void palindromeTest(){
    assertTrue(checker.esPalindromo( cadena: "rotor"));
}

@Test new *
void numberTest(){
    assertTrue(checker.esPalindromo( cadena: "1221221"));
}

@Test new *
void spaceTest(){
    assertTrue(checker.esPalindromo( cadena: "roto r"));
}

@Test new *
void phraseSpaceTest(){
    assertTrue(checker.esPalindromo( cadena: " casa asac"));
}
```

✖ PalindromoTest	41 ms
✖ phraseSpaceTest()	38 ms
✖ spaceTest()	2 ms
✓ numberTest()	
✓ notPalindromeTest()	1 ms
✓ palindromeTest()	

3.4 ¿Qué resultados arrojan sus Test con estas entradas: “aca”, “acas”, “h”?

```
@Test new *
void acaTest(){
    assertTrue(checker.esPalindromo( cadena: "aca"));
}

@Test new *
void acasTest(){
    assertFalse(checker.esPalindromo( cadena: "acas"));
}

@Test new *
void hTest(){
    assertTrue(checker.esPalindromo( cadena: "h"));
}
```

✓ PalindromoTest	31 ms
✓ acaTest()	30 ms
✓ hTest()	
✓ acasTest()	1 ms

Paso 4: Mejorando el método, probando más.

4.2 De las pruebas analizadas, concluya y construya una versión mejorada de su método. Construya además nuevas pruebas unitarias considerando los casos anteriores y verifique sus resultados teóricos con los empíricos.

- ¿Qué consideraciones tomaron en cuenta?
 - Dentro de algunas frases, hay espacios y mayúsculas, también los strings sin caracteres.
- ¿Qué mejoró en su método?
 - Se adapta más a distintos tipos de cadenas, ahora puede decir que una frase es un palíndromo por ejemplo.
- ¿Qué rol jugaron las pruebas en mejorar su código?
 - Fueron necesarias para poder ver que factores no se estaban tomando en cuenta a la hora de analizar la cadena, por ejemplo, antes no se eliminaban

los espacios, por lo que frases que si son palíndromos los daba como que no lo eran.

```
@Test new *
void notPalindromeTest() {
    assertFalse(checker.esPalindromo(texto: "arriba"));
}

@Test new *
void palindromeTest() {
    assertTrue(checker.esPalindromo(texto: "rotor"));
}

@Test new *
void numberTest() {
    assertTrue(checker.esPalindromo(texto: "1221221"));
}

@Test new *
void spaceTest() {
    assertTrue(checker.esPalindromo(texto: "roto r"));
}

@Test new *
void phraseSpaceTest() {
    assertTrue(checker.esPalindromo(texto: " casa asac"));
}
```

```
@Test new *
void nullTest(){
    assertFalse(checker.esPalindromo(texto: ""));
}
```

✓	PalindromoTest	29 ms
✓	phraseSpaceTest()	25 ms
✓	acaTest()	1 ms
✓	spaceTest()	1 ms
✓	numberTest()	1 ms
✓	notPalindromeTest()	1 ms
✓	palindromeTest()	