

# Caso de Palíndromos – Pruebas Unitarias

Nicolás Paila, Jorge Quidel

Departamento de Ciencias de la Computación e Informática

Universidad de la Frontera – Temuco

ICC264-1: Programación

Dr. Samuel Sepúlveda Cuevas

Septiembre, 2022



```
function esPalindromo(cadena) {

    let resultado = "";

    resultado = cadena.split("").reverse().join("");

    return cadena === resultado;

}
```

## 1. Discutir y concluir

### 1.1 ¿Qué hace el método?

El método verifica que un string dado es un palíndromo o no.

### 1.2 ¿Cómo lo hace?

El método acepta la variable de entrada ‘cadena’ del tipo string, luego declara e inicializa la variable string ‘resultado’, cuya asignación subsiguiente consiste en una separación de los caracteres de la variable ‘cadena’ en un arreglo, en la cual después invierte el orden de sus elementos, y los junta en un nuevo string, representando así la versión invertida de la variable ‘cadena’.

Finalmente, retorna la comparación booleana entre este resultado y la cadena de entrada.

**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) {
    String resultado = new StringBuilder(cadena).reverse().toString();
    return resultado.equals(cadena);
}
```

### **3. Ok! Si el método funciona ¿Qué puede malir sal? ;-)**

La cantidad de expresiones que abarcan lo que es un palíndromo son más amplias para lo que el método verifica. Las sentencias con espacios si pueden llegar a ser palíndromos como también formatos numéricos como las fechas: '11/11/11 11:11' es considerado un palíndromo. Así mismo, el método no toma en cuenta si es vacía la entrada o el uso de mayúsculas o de acentos.

#### **3.1 Discutir en grupo el diseño de un plan de pruebas para este caso.**

Por lo dicho anteriormente, el plan de pruebas estará forzado a fallar ya que el método solo verifica un conjunto acotado con las formas más simples de palíndromos (aquellos que no posean espacios, acentos, puntuaciones, uso de mayúsculas.)

**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. Probar que una cadena no vacía sea palíndroma
2. Probar que una cadena no vacía no sea palíndroma
3. Probar que una cadena vacía no sea palíndroma (va a fallar)
4. Probar que una cadena que contenga espacios sea palíndroma (va a fallar)
5. Probar que una cadena que contenga tildes sea palíndroma (va a fallar)

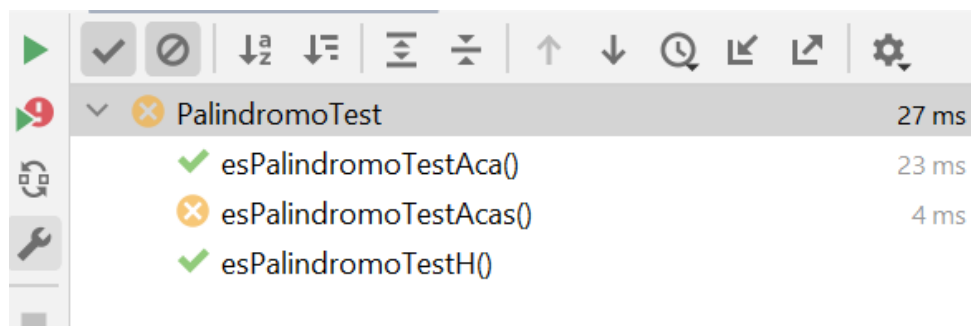
**3.3 Estando seguros de que sus casos de pruebas unitarias son amplios y relevantes, ahora impleméntelos en Java usando JUnit.**

```

@Test
void esPalindromoTest() {
    assertTrue(Palindromo.esPalindromo( cadena: "saippuakuppinippukauppias"));
}
@Test
void esPalindromoFallaTest() {
    assertFalse(Palindromo.esPalindromo( cadena: "palindromo"));
}
@Test
void esPalindromoFallaSiEsVacioTest() {
    assertFalse(Palindromo.esPalindromo( cadena: ""));
}
@Test
void esPalindromoSiContieneEspacioTest() {
    assertTrue(Palindromo.esPalindromo( cadena: "sata no scill atemy me tallicso nat as "));
}
@Test
void esPalindromoSiContieneTildeTest() {
    assertTrue(Palindromo.esPalindromo( cadena: "sabás"));
}

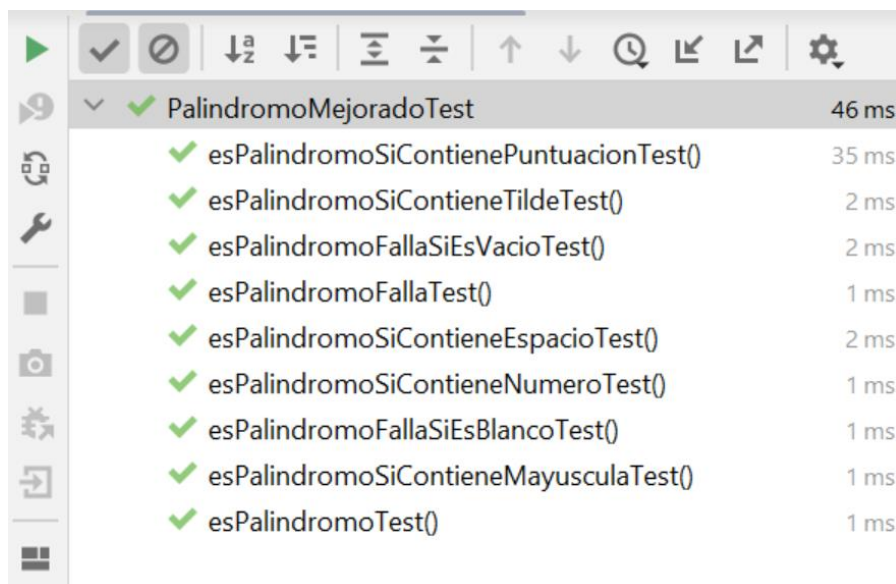
```

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



**4.1 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.**

```
public static boolean esPalindromo(String cadena) {
    if (cadena.isBlank()) {
        return false;
    }
    cadena = Normalizer.normalize(cadena, Normalizer.Form.NFD)
        .replaceAll(regex: "\\p{M}|/|\\:\\|-|\\s", replacement: "")
        .toLowerCase();
    String cadenaInvertida = new StringBuilder(cadena).reverse().toString();
    return cadenaInvertida.equals(cadena);
}
```



Test Method	Execution Time
PalindromoMejoradoTest	46 ms
esPalindromoSiContienePuntuacionTest()	35 ms
esPalindromoSiContieneTildeTest()	2 ms
esPalindromoFallaSiEsVacioTest()	2 ms
esPalindromoFallaTest()	1 ms
esPalindromoSiContieneEspacioTest()	2 ms
esPalindromoSiContieneNumeroTest()	1 ms
esPalindromoFallaSiEsBlancoTest()	1 ms
esPalindromoSiContieneMayusculaTest()	1 ms
esPalindromoTest()	1 ms

- ¿Qué consideraciones tomaron en cuenta?

En el método se tomó en cuenta la forma más ampliada de lo que es un palíndromo.

- ¿Qué mejoró en su método?

El método es agnóstico de si la entrada contiene algún acento, espacio, mayúsculas, o puntuación del tipo '/', ':', '-'. Además, verifica si la entrada es vacía.

- ¿Qué rol jugaron las pruebas en mejorar su código?

Con las pruebas unitarias se podía verificar en el instante si alguna modificación o característica nueva en el código obstruía la operación correcta de un método que funcionaba sin problemas previamente.

### **5. Discutir experiencia y resultados con TODO el curso y Concluir!!!**

Se puede concluir que a través del uso adecuado de las pruebas unitarias se puede mejorar de forma eficiente y rápida alguna solución o implementación sin comprometer alguna funcionalidad anterior.