

Trabajo en clases: Caso Palíndromos

Nombre: Víctor Mardones Bravo

Desarrollo de los pasos 1-5

Paso 1: Leer y entender el contexto problema.

Se necesita saber si el input entregado a un método es o no un palíndromo.

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?

Respuesta: El método se encarga de revisar si un String es un palíndromo o no.

¿Cómo lo hace?

Respuesta: Crea un String "resultado" donde almacena la String "cadena" al revés y después las compara.

¿Cómo lo uso?

Respuesta: Se puede usar en un programa donde el usuario ingresa una palabra y después el programa le retorna texto indicando si la palabra ingresada es un palíndromo o no.

Dé al menos un ejemplo de su uso (sin codificar nada).

Respuesta: El usuario ingresa la palabra "anitalavalatina". El programa retorna "true" porque la palabra es un palíndromo.

Otro ejemplo: El usuario ingresa la palabra "holamundo". El programa retorna "false" porque la palabra no es un palíndromo.

1.2 Discutir resultados entre el grupo y con el profesor

Paso 2: Entender el método JavaScript

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

Respuesta: El método toma una palabra, la separa en un arreglo de caracteres, invierte el arreglo y después lo junta todo en una palabra nueva. Finalmente, compara la palabra que se tenía al principio con la palabra inversa. Retorna “verdadero” si las palabras coinciden (en otras palabras, es un palíndromo) y “falso en caso contrario.

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

Respuesta: La versión 100% equivalente está disponible en GitHub [aquí](#).

2.3 Discutir resultados entre el grupo y con el profesor.

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

3.1 Discutir en grupo el diseño de un plan de prueba unitarias para este caso.

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

Respuesta: Casos de prueba a implementar: “anitalavalatina” (palíndromo), “AnitaLavaLaTina” (palíndromo), “anita lava la tina” (palíndromo), “010010” (palíndromo) y “%#@#%” (palíndromo).

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

Respuesta: La versión con casos de prueba está disponible en GitHub [aquí](#). De los test realizados, “AnitaLavaLaTina” y “anita lava la tina” retornan “test failed”. El método todavía no considera mayúsculas y minúsculas por igual y no ignora los espacios entre letras.

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

Respuesta: “aca” (palíndromo) retorna “test passed”, “acas” (no palíndromo) retorna “test failed” y “h (¿palíndromo?) retorna “test passed”.

Paso 4: Mejorando el método, probando más.

4.1 Considere los siguientes casos de prueba, no codifique NADA, discuta en su grupo.

- Caso de prueba 1:

Input: 200 (número entero)

Output: ????

Respuesta: La palabra no es un palíndromo. Se espera que al ejecutar el caso de prueba retorne "test passed".

- Caso de prueba 2:

Input: "" (cadena vacía)

Output: ????

Respuesta: Se espera que retorne "test passed". El resultado es debatible. Técnicamente no se ingresó ninguna palabra. Es un caso excepcional que se puede mejorar.

- Caso de prueba 3:

Input: "aaabccbbaa"

Output: ????

Respuesta: La palabra es un palíndromo. Se espera que el caso de prueba retorne "test passed".

- Caso de prueba 4:

Input: "ahabccbbaa"

Output: ????

Respuesta: La palabra no es un palíndromo. Se espera que el caso de prueba retorne "test passed".

- Caso de prueba 5:

Input: "La tele letal"

Output: ????

Respuesta: La frase es un palíndromo, pero se espera que el caso de prueba retorne "test failed". El método todavía no ignora los espacios entre las palabras, ni toma en cuenta las mayúsculas.

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.

Respuesta: La versión final del programa está disponible [aquí](#).

¿Qué consideraciones tomaron en cuenta?

Respuesta: Consideré los casos de prueba que realicé durante el ejercicio. Hubo muchos casos que el ejemplo original no había considerado. Entre ellos están los espacios, las mayúsculas/minúsculas y las tildes. También están los casos de ingresar un String vacío o de 1 sólo carácter, los cuales no pueden calificar como palíndromos ya que no son palabras.

¿Qué mejoró en su método?

Respuesta: Se realizaron las siguientes mejoras:

- 1.-Se cambió el método del ejemplo en JavaScript a uno más fácil de entender, que no necesita separar la cadena ingresada para trabajar.
- 2.-El método ahora ignora los espacios entre palabras.
- 3.-El método transforma las mayúsculas en minúsculas y elimina las tildes que se usan comúnmente en el español.
- 4.-En caso de ingresar un String vacío o una única letra, número o símbolo, el método ahora retorna "false", porque no se puede considerar un solo carácter como palíndromo (no es una palabra o frase).

¿Qué rol jugaron las pruebas en mejorar su código?

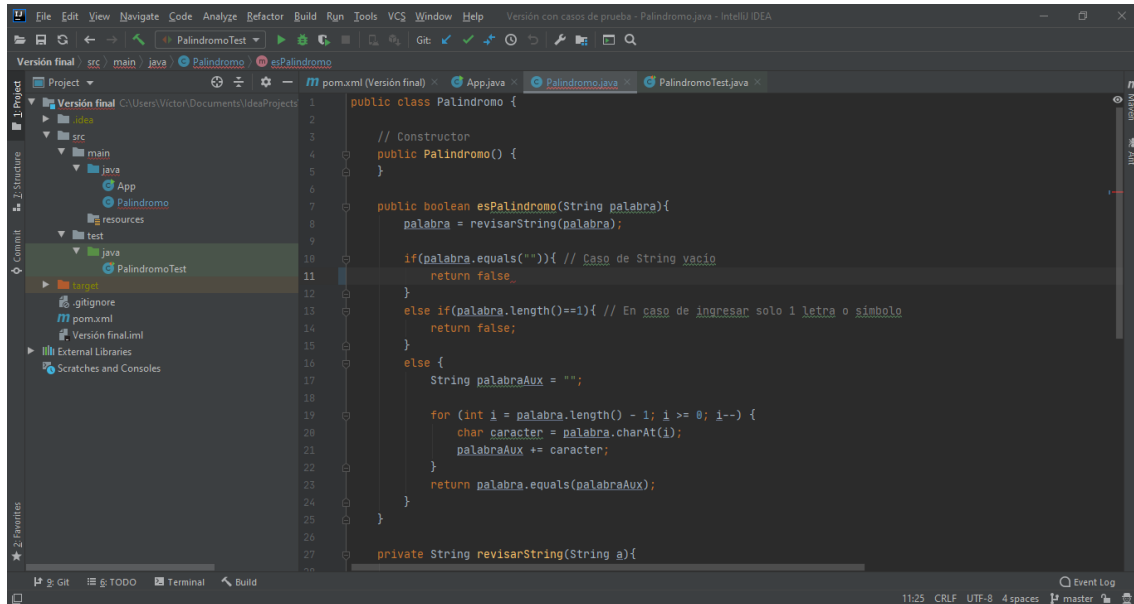
Respuesta: Fueron de gran utilidad para probar los casos en los que el programa puede fallar y para revisar en poco tiempo que la versión final funciona en todos ellos.

Paso 5: Discutir experiencia y resultados con TODO el curso y Concluir!!!

Conclusión personal: Esta actividad fue muy interesante porque gracias a ella aprendí a usar git bien y a hacer las pruebas unitarias con JUnit correctamente. Al principio fue difícil, pero después de perseverar un poco se empezó a hacer más fácil y el trabajo cada vez iba quedando más ordenado, a pesar de lo desordenado que soy para programar.

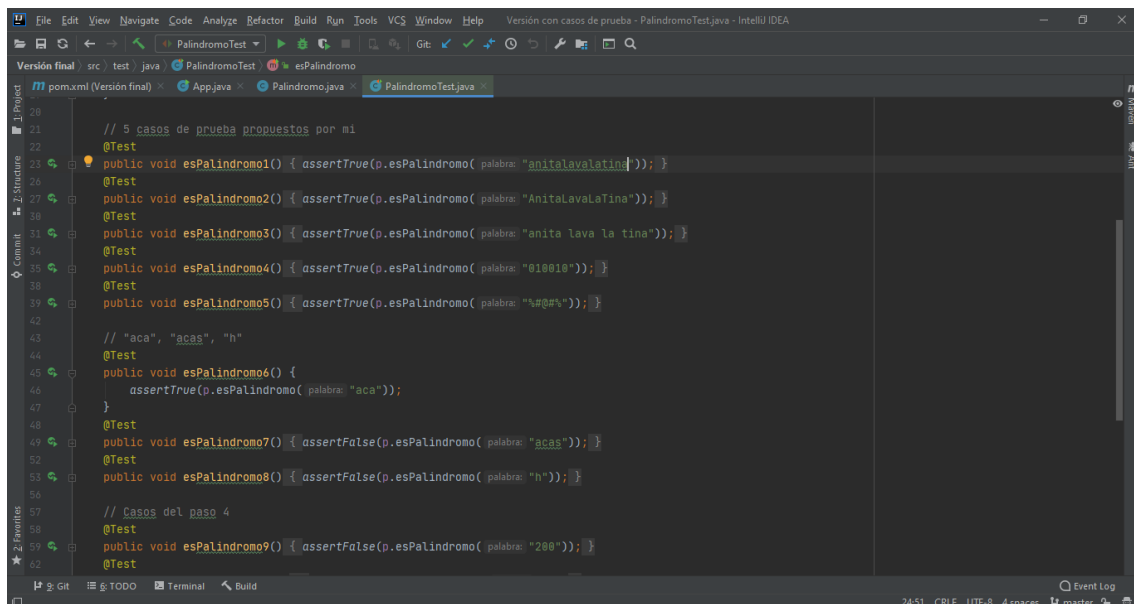
Evidencias del trabajo realizado

Trabajo con el código fuente:



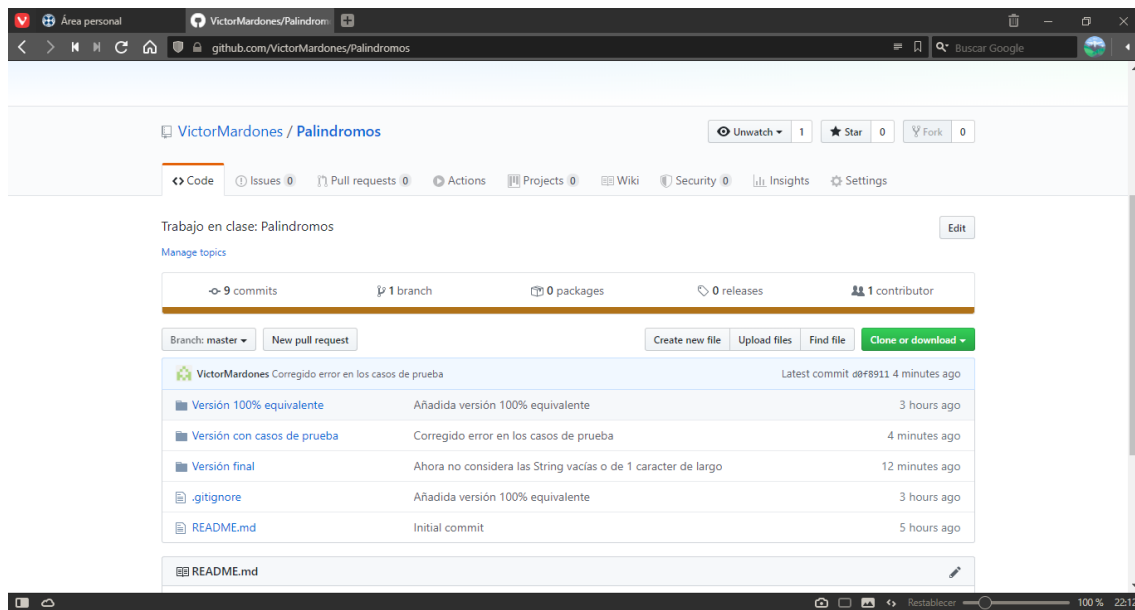
```
1 public class Palindromo {
2
3     // Constructor
4     public Palindromo() {
5     }
6
7     public boolean esPalindromo(String palabra){
8         palabra = revisarString(palabra);
9
10        if(palabra.equals("")){ // Caso de String vacio
11            return false;
12        }
13        else if(palabra.length()==1){ // En caso de ingresar solo 1 letra o simbolo
14            return false;
15        }
16        else {
17            String palabraAux = "";
18
19            for (int i = palabra.length() - 1; i >= 0; i--) {
20                char caracter = palabra.charAt(i);
21                palabraAux += caracter;
22            }
23            return palabra.equals(palabraAux);
24        }
25    }
26
27    private String revisarString(String a){
28    }
29 }
```

Diseño de los casos de prueba:

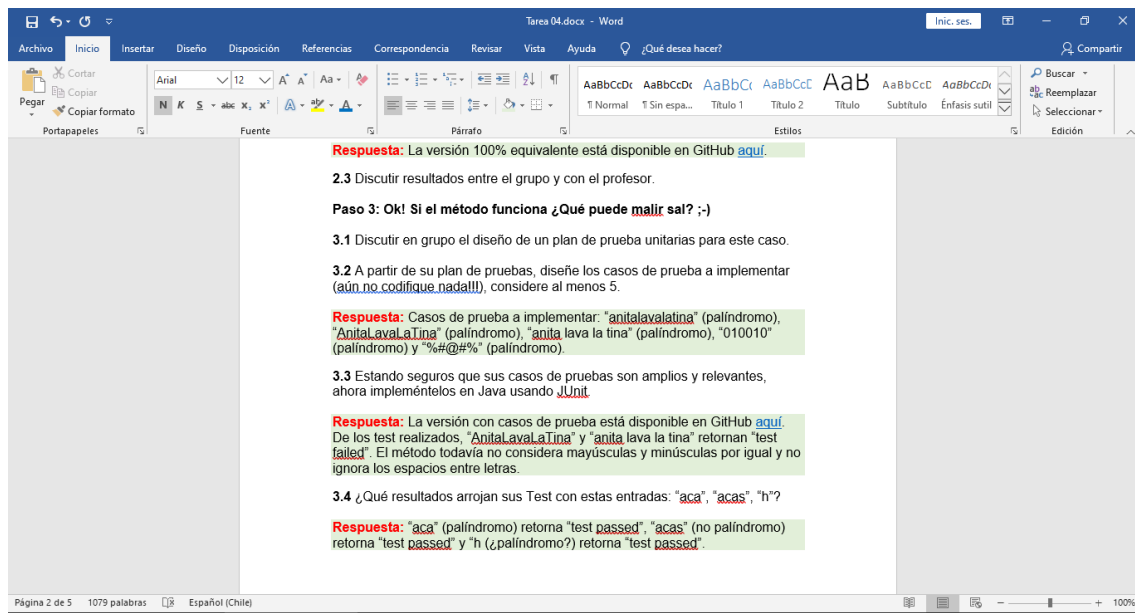


```
20 // 5 casos de prueba propuestos por mi
21 @Test
22 public void esPalindromo1() { assertTrue(p.esPalindromo( palabra: "anitalavalatin")); }
23 @Test
24 public void esPalindromo2() { assertTrue(p.esPalindromo( palabra: "AnitaLavaLaTina")); }
25 @Test
26 public void esPalindromo3() { assertTrue(p.esPalindromo( palabra: "anita lava la tina")); }
27 @Test
28 public void esPalindromo4() { assertTrue(p.esPalindromo( palabra: "010010")); }
29 @Test
30 public void esPalindromo5() { assertTrue(p.esPalindromo( palabra: "%#@#%")); }
31
32 // "aca", "acas", "h"
33 @Test
34 public void esPalindromo6() {
35     assertTrue(p.esPalindromo( palabra: "aca"));
36 }
37 @Test
38 public void esPalindromo7() { assertFalse(p.esPalindromo( palabra: "acas")); }
39 @Test
40 public void esPalindromo8() { assertFalse(p.esPalindromo( palabra: "h")); }
41
42 // Casos del paso 4
43 @Test
44 public void esPalindromo9() { assertFalse(p.esPalindromo( palabra: "200")); }
45 @Test
46 }
```

Repositorio de GitHub:



Trabajando en este PDF (en Microsoft Word):



Links y explicación del repositorio GitHub

El repositorio de GitHub completo está disponible [aquí](#).

Este repositorio contiene 3 versiones del trabajo realizado en clases. En orden de avance son las siguientes:

1. Versión original: Esta es la versión más parecida al primer ejemplo en JavaScript.
2. Versión con casos de prueba: Como su nombre indica, esta versión incluye casos de prueba para ver qué tan bien funcionaba el ejemplo original.
3. Versión final: Esta versión incluye muchas mejoras con respecto a las anteriores para que el programa funcione mejor que nunca. Ahora verifica muchos casos de prueba y no muestra ningún error en ninguno de ellos (al menos por ahora).

Nota: Si no compila, verificar que se está usando Java versión 14.