Ejercicios I – **Cadenas en Java**

Utiliza tu **IDE** preferido: Apache NetBeans, Eclipse o IntelliJ IDEA.
También puedes utilizar VSCode con el *Java Extension Pack* instalado.

Crea un Nuevo **proyecto** Java llamado **EjerciciosStrings-1**.
Este proyecto contendrá todas las aplicaciones propuestas a continuación.

Utiliza un paquete con nombre **es.maestredam.str** o cualquier otro de tu elección.

1) NombreApp

Crea una aplicación Java que realice las siguientes operaciones:

a) Crear un **objeto** cadena (de la clase String) llamada miNombre, que contenga tu nombre y apellidos.
Debes crearlo usando el operador **new**.
b) Escribir en consola el número de caracteres que tiene la cadena miNombre.
c) Escribir en consola si la cadena miNombre tiene o no la letra **z**.

Una solución sería:

```
public static void main(String[] args) {
    // Declaro un objeto de la clase String
    String miNombre;

    // Creo el objeto para el texto de mi nombre y apellidos
    miNombre = new String("Miguel Marrero Marquez");

    // Oye, objeto miNombre ¿cuál es tu longitud?
    int longi = miNombre.length();
    System.out.printf("%s tiene %d caracteres ", miNombre, longi);

    // Oye, objeto miNombre ¿tienes alguna z?
    int ret = miNombre.indexOf('z');
    // indexOf devuelve -1 cuando no aparece el carácter indicado
    if (ret == -1) {
         System.out.printf("y NO tiene 'z' %n");
    }
    else {
         System.out.printf("y sí tiene al menos una 'z' %n");
    }
}
```

Salida por consola:

```
Miguel Marrero Marquez tiene 22 caracteres y sí tiene al menos una 'z'
```

2) InfoStringApp [Obteniendo información de una cadena]

Crea una aplicación Java que realice las siguientes operaciones:

a) Crear un **objeto** cadena (de la clase String) llamada texto, con el siguiente contenido: *"Estoy seguro de que el universo está lleno de vida inteligente"*
Debes crearlo usando el operador **new**.

b) Mostrar la longitud de la cadena.
c) Mostrar el carácter que hay en la primera posición; el que hay en la posición 23 y el que hay en penúltima posición.
d) Mostrar la posición que ocupa el carácter 'y'.
e) Comprobar si el primer carácter de la cadena es 'E', mostrando por consola un mensaje que lo indique.

Investiga cómo funciona los métodos **length**, **charAt**, **indexOf**, **startsWith**, todos ellos de la clase **java.lang.String**.

Salida por consola:

```
Longitud: 62
Primera posición: E. Posición 23: u. Penúltima posición: t
La posición del carácter 'y' es 4
¿Comienza la cadena por E? -> true
```

3) ComparaPalabrasApp [Examinando si dos cadenas son iguales}

Crea una aplicación Java que lea dos palabras por teclado y nos muestre en pantalla:

a) Si son exactamente iguales o no lo son.
b) Si tienen la misma longitud o no.
c) En caso de que NO la tengan, escribir cuál de ellas tiene más caracteres.
d) Si son iguales en caso de no tener en cuenta mayúsculas y minúsculas.

Recuerda que el método **showInputDialog devuelve un objeto de la clase String** con la cadena tecleada.

Ejemplo 1

```
Tecleadas "abeto" y "aBeto"
NO son exactamente iguales
pero la longitud es la misma
y serían iguales si no tenemos en cuenta las mayúsculas/minúsculas
```

Ejemplo 2

```
Tecleadas "Manzana" y "Apple"
NO son exactamente iguales
ni siquiera tienen la misma longitud (la primera tiene más caracteres)
```

4) OrdenaPalabrasApp [Examinando si una cadena es mayor, menor o igual a otra]

Crea una aplicación Java que lea por teclado dos palabras y las escriba en pantalla ordenadas alfabéticamente de la A a la Z.

Investiga cómo funciona el método **compareTo**

Ejemplo 1

```
Tecleadas "Patata" y "Batata"
En orden: "Batata" y "Patata"
```

Ejemplo 2

```
Tecleadas "patata" y "Patata"
En orden: "Patata" y "patata"
```

5) AnalizaCadenaApp [Analizando caracteres]

Crea una aplicación Java que lea por teclado una cadena y compruebe si hay alguna **a** en los 5 primeros caracteres.

Después, debe visualizar en consola la cadena completa, pero evitando escribir vocales; en su lugar debes escribir un asterisco *****

6) TransformaCadenaApp [Extrayendo y transformando cadenas]

Crea una aplicación Java que realice las siguientes operaciones:

a) Crear un **objeto** cadena (de la clase String) llamado texto, con el siguiente contenido: *"Estoy seguro de que el universo está lleno de vida inteligente"*
Debes crearlo usando el operador **new**.

b) Añadir al objeto texto, la frase "Simplemente ha sido demasiado inteligente como para venir aquí", utilizando el operador de concatenar cadenas **+**

c) Añade el punto final al objeto texto mediante el método **concat**. ¿Hay alguna diferencia entre concat y +?

d) Crear una nueva cadena con la porción de texto, que contiene "universo".

Investiga cómo funciona el método **substring**

e) Crear una nueva cadena con la cadena texto transformada en mayúsculas.

Salida por consola:

```
texto contiene "Estoy seguro de que el universo está lleno de vida inteligente"
texto ahora contiene: "Estoy seguro de que el universo está lleno de vida
inteligente. Simplemente ha sido demasiado inteligente como para venir aquí."
texto en mayúsculas: "ESTOY SEGURO DE QUE EL UNIVERSO ESTÁ LLENO DE VIDA
INTELIGENTE. SIMPLEMENTE HA SIDO DEMASIADO INTELIGENTE COMO PARA VENIR AQUÍ."
```

7) DivideCadenaApp [Partiendo una cadena en dos subcadenas]

Crea una aplicación Java que lea una cadena por teclado y la "rompa" en dos mitades.

Después deberá escribir en consola cada una de las mitades.

- Las mitades serán iguales siempre que la cadena tenga un número de caracteres par.
- En caso de que el número de caracteres sea impar, no se podrá hacer la mitad exacta, pero partiremos la cadena lo más aproximado a la mitad.

Ejemplo 1

```
Tecleado: abetillo
abet | illo
```

Ejemplo 2

```
Tecleado: abeto
abe | to
```

8) NumOcurrenciasApp [Recorriendo una cadena]

Crea una aplicación Java que contenga **sólo al método main** y que realice las siguientes operaciones:

a) Crear un **objeto** cadena con el siguiente contenido: *"Lo mejor de los booleanos es que si te equivocas estás a un sólo bit de la solución correcta"*

b) Mostrar por consola el número de veces que aparece la letra **a** y la letra **b** en la cadena anterior.

Piensa y escribe un pseudocódigo que resuelve el problema ANTES de comenzar su implementación en Java.

Salida por consola:

```
La letra a aparece 5 veces.
La letra b aparece 2 veces.
```

9) NumOcurrenciasMetodoApp [Implement. un método que recorre una cadena]

Crea una aplicación Java que contenga al método main y cualquier **otro método** que creas conveniente, para que muestre por consola el número de veces que aparece cada vocal en la cadena del ejercicio anterior.

a) Crea un **objeto** cadena con el siguiente contenido: *"Lo mejor de los booleanos es que si te equivocas estás a un sólo bit de la solución correcta"*

b) Define un **método** que cuente las ocurrencias de un carácter en una cadena dada.

> Piensa la cabecera del método a implementar: ¿qué categoría tiene? (escritura, lectura o cálculo) ¿qué tipo devuelve? ¿qué parámetros toma?

c) Si el número de veces que se repite una vocal es superior a 5, debe aparecer el mensaje "Exceso de " y la letra correspondiente.

d) Documenta el método con comentarios para javadoc.

e) Implementa y prueba el método. Si no funciona correctamente, utiliza el Depurador.

f) Utiliza el método en la aplicación para realizar lo que se pide.

Salida por consola:

```
La letra a aparece 5 veces.
La letra e aparece 10 veces.
La letra i aparece 4 veces.
La letra o aparece 10 veces.
La letra u aparece 4 veces.
```

10) PalindromoApp [Recorriendo una cadena]

Crea una aplicación Java que lea una cadena por teclado y escriba si se trata de un palíndromo o no.

> Un palíndromo es una cadena que se lee igual de izquierda a derecha que de derecha a izquierda. Ejemplo: "Dabale arroz a la zorra el abad".

Primero será necesario ignorar los espacios en blanco. Utiliza un método de la clase String para eliminar dichos espacios.

También deberíamos no tener en cuenta mayúsculas y minúsculas.

Salida por consola:

```
Tecleado: Dabale arroz a la zorra el abad
La cadena ahora es: dabalearrozalazorraelabad
SI es palíndromo
```

11) EdadApp [Concatenando cadenas]

Crea una aplicación Java que lea por teclado tu año de nacimiento y el año actual.

La aplicación deberá crear una cadena (objeto String) con el valor de tu edad y después añadirle la palabra "años".

Escribe en pantalla mensajes de traza con el contenido de la cadena que has creado.

Recuerda que el método genérico **valueOf** convierte valores a String.

Salida por consola:
Traza: La cadena tiene "24"
Traza: Ahora la cadena tiene "24 años"
24 años

12) SerieEnCadenaApp

Añade a la aplicación anterior, un **método** que dado un número entero, devuelva un objeto String con todos los dígitos desde 0 al número pasado por parámetro.

- Por ejemplo, para 5, la cadena devuelta tendrá "0 1 2 3 4 5"
- Por ejemplo, para 12, la cadena devuelta tendrá "0 1 2 3 4 5 6 7 8 9 10 11 12"

Prueba el método haciendo dos o tres llamadas más con números diferentes.

Recuerda la diferencia entra los objetos String y StringBuffer/StringBuilder.