Strings

Programación I - UNGS

Strings

El tipo de datos **String** permite representar secuencias de caracteres, junto con algunas operaciones sencillas para trabajar con las mismas.

Strings

El tipo de datos **String** permite representar secuencias de caracteres, junto con algunas operaciones sencillas para trabajar con las mismas.

Ejemplos de strings:

- “pepe”

Strings

El tipo de datos **String** permite representar secuencias de caracteres, junto con algunas operaciones sencillas para trabajar con las mismas.

Ejemplos de strings:

- “pepe”
- “Hola, mundo!”

Strings

El tipo de datos **String** permite representar secuencias de caracteres, junto con algunas operaciones sencillas para trabajar con las mismas.

Ejemplos de strings:

- “pepe”
- “Hola, mundo!”
- “253” (notar que no es un **int**)

Strings

El tipo de datos **String** permite representar secuencias de caracteres, junto con algunas operaciones sencillas para trabajar con las mismas.

Ejemplos de strings:

- "pepe"
- "Hola, mundo!"
- "253" (notar que no es un **int**)
- "2+2" (notar que no es el **int** 4!)

Strings

El tipo de datos **String** permite representar secuencias de caracteres, junto con algunas operaciones sencillas para trabajar con las mismas.

Ejemplos de strings:

- “pepe”
- “Hola, mundo!”
- “253” (notar que no es un **int**)
- “2+2” (notar que no es el **int** 4!)
- “” (string vacío)

Declarando un String en Java

- Declaramos las variables anteponiendo la palabra **String**

Declarando un String en Java

- Declaramos las variables anteponiendo la palabra **String** (con la inicial en mayúscula!)

Declarando un String en Java

- Declaramos las variables anteponiendo la palabra **String** (con la inicial en mayúscula!)

```
1    String s1;
2    String s2 = "";
3    String s3 = "Hola mundo!";
```

Declarando un String en Java

- Declaramos las variables anteponiendo la palabra **String** (con la inicial en mayúscula!)

```
1    String s1;
2    String s2 = "";
3    String s3 = "Hola mundo!";
```

- Podemos imprimir un string de la siguiente manera:

```
1    String s = "Hola mundo!";
2    System.out.println(s);
```

Declarando un String en Java

- Declaramos las variables anteponiendo la palabra **String** (con la inicial en mayúscula!)

```
1    String s1;
2    String s2 = "";
3    String s3 = "Hola mundo!";
```

- Podemos imprimir un string de la siguiente manera:

```
1    String s = "Hola mundo!";
2    System.out.println(s);
```

- O, en forma equivalente, haciendo:

```
1    System.out.println("Hola mundo!");
```

Accediendo a un String

- El método **charAt** permite acceder a los caracteres que conforman el **String**.

Accediendo a un String

- El método **charAt** permite acceder a los caracteres que conforman el **String**.
- Recibe como parámetro un **int** que indica la posición, y retorna un **char** con el caracter ubicado en la posición indicada.

Accediendo a un String

- El método **charAt** permite acceder a los caracteres que conforman el **String**.
- Recibe como parámetro un **int** que indica la posición, y retorna un **char** con el caracter ubicado en la posición indicada.
- La primera posición del **String** es la posición cero.

Accediendo a un String

- El método **charAt** permite acceder a los caracteres que conforman el **String**.
- Recibe como parámetro un **int** que indica la posición, y retorna un **char** con el caracter ubicado en la posición indicada.
- La primera posición del **String** es la posición cero.

- Ejemplo:

```
1    String fruta = "banana";
2    char letra = fruta.charAt(1);
3    System.out.println(letra);
```

Accediendo a un String

- El método **charAt** permite acceder a los caracteres que conforman el **String**.
- Recibe como parámetro un **int** que indica la posición, y retorna un **char** con el caracter ubicado en la posición indicada.
- La primera posición del **String** es la posición cero.

- Ejemplo:

```
1    String fruta = "banana";
2    char letra = fruta.charAt(1);
3    System.out.println(letra);
```

b	a	n	a	n	a
0	1	2	3	4	5

Accediendo a un String

- Si el parámetro que se pasa a **charAt** no corresponde a un índice (posición) válido del **String**, se genera una excepción (error en tiempo de ejecución).

Accediendo a un String

- Si el parámetro que se pasa a **charAt** no corresponde a un índice (posición) válido del **String**, se genera una excepción (error en tiempo de ejecución).

- Ejemplo:

```
1    String fruta = "banana";
2    char letra = fruta.charAt(11);
```

Accediendo a un String

- Si el parámetro que se pasa a **charAt** no corresponde a un índice (posición) válido del **String**, se genera una excepción (error en tiempo de ejecución).

- Ejemplo:

```
1    String fruta = "banana";
2    char letra = fruta.charAt(11);
```

- Este código genera la siguiente excepción:

  Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 11

La longitud de un String

- La función **length** permite consultar la longitud de un **String**, y retorna un **int** como resultado.

La longitud de un String

- La función **length** permite consultar la longitud de un **String**, y retorna un **int** como resultado.

- Ejemplo:

```
1    String fruta = "banana";
2    System.out.println(fruta.length());
```

La longitud de un String

- La función **length** permite consultar la longitud de un **String**, y retorna un **int** como resultado.

- Ejemplo:

```
1    String fruta = "banana";
2    System.out.println(fruta.length());
```

- Esto imprime 6 por pantalla

Sintaxis

- Observar la sintaxis que utilizamos para acceder a los métodos de la clase **String**:

```
1    String fruta = "banana";
2    fruta.charAt(3);
3    fruta.length();
```

Sintaxis

- Observar la sintaxis que utilizamos para acceder a los métodos de la clase **String**:

```
1    String fruta = "banana";
2    fruta.charAt(3);
3    fruta.length();
```

- Estas funciones (charAt y length) son métodos de la clase **String**, que se ejecutan sobre una instancia de la clase.

Concatenación

- Utilizamos el operador + para concatenar strings:

```
1    String frutas = "banana" + " y " + "pera";
```

Concatenación

- Utilizamos el operador + para concatenar strings:

```
1    String frutas = "banana" + " y " + "pera";
```

- También podemos usarlo para concatenar strings con otros tipos de datos, como un **char** o un **int**:

```
1    String frutas = "banana" + 's';
2    String frase = frutas + " tiene " + frutas.length() + " letras";
```

Recorrido de un String

- Es habitual recorrer un **String** de izquierda a derecha (secuencialmente), accediendo a cada **char** de la secuencia:

```
1    String fruta = "banana";
2    for (int i = 0; i < fruta.length(); i++) {
3        System.out.println(fruta.charAt(i));
4    }
```

Recorrido de un String

- Es habitual recorrer un **String** de izquierda a derecha (secuencialmente), accediendo a cada **char** de la secuencia:

```
1    String fruta = "banana";
2    for (int i = 0; i < fruta.length(); i++) {
3        System.out.println(fruta.charAt(i));
4    }
```

- Notemos que la posición del último elemento de una cadena **s** (siempre que la misma no sea vacía) es **s.length() - 1**

Recorrido de un String

- El código anterior es equivalente a ...

```
1   String fruta = "banana";
2   int i = 0;
3   while (i < fruta.length()) {
4       System.out.println(fruta.charAt(i));
5       i++;
6   }
```

Recorrido de un String

- Los elementos del tipo de datos **char** se notan entre apóstrofes simples.

Recorrido de un String

- Los elementos del tipo de datos **char** se notan entre apóstrofes simples.
- Ejemplo:

```
1  String fruta = "banana";
2  int cont = 0;
3  for (int i = 0; i < fruta.length(); i++) {
4    if (fruta.charAt(i) == 'a') {
5        cont++;
6    }
7  }
8  return cont;
```

Recorrido de un String

- Los elementos del tipo de datos **char** se notan entre apóstrofes simples.
- Ejemplo:

```
1    String fruta = "banana";
2    int cont = 0;
3    for (int i = 0; i < fruta.length(); i++) {
4      if (fruta.charAt(i) == 'a') {
5            cont++;
6      }
7    }
8    return cont;
```

- Este código devuelve la cantidad de letras 'a' que tiene la palabra "banana"

El método indexOf

- Dado un **char**, el método **indexOf** encuentra el índice donde aparece ese caracter en el **String** por primera vez.

El método indexOf

- Dado un **char**, el método **indexOf** encuentra el índice donde aparece ese caracter en el **String** por primera vez.

- Ejemplo:

```
1    String fruta = "banana";
2    int indice = fruta.indexOf('a');
```

El método indexOf

- Dado un **char**, el método **indexOf** encuentra el índice donde aparece ese caracter en el **String** por primera vez.

- Ejemplo:

```
1    String fruta = "banana";
2    int indice = fruta.indexOf('a');
```

- En cierto sentido, es el método opuesto de **charAt**.

El método indexOf

- Dado un **char**, el método **indexOf** encuentra el índice donde aparece ese caracter en el **String** por primera vez.

- Ejemplo:

```
1    String fruta = "banana";
2    int indice = fruta.indexOf('a');
```

- En cierto sentido, es el método opuesto de **charAt**.
- Si el **char** que se pasa como parámetro no está en el **String**, entonces **indexOf** retorna -1 como resultado.

El método indexOf

- Una segunda versión de **indexOf** toma como segundo parámetro un **int** que indica desde qué índice de la cadena se debe comenzar la búsqueda.

```
1    int indice = fruta.indexOf('a', 2);
```

Los Strings son inmutables

- Algo muy particular con respecto a los Strings es que “no se los puede modificar”. Es decir, no es posible alterar su contenido.

Los Strings son inmutables

- Algo muy particular con respecto a los Strings es que “no se los puede modificar”. Es decir, no es posible alterar su contenido.
- Obviamente, si quiero modificar el string que está guardado en una variable, siempre puedo cambiarlo por otro:

```
1    String fruta = "anana";
2    fruta = "b" + fruta;
3    fruta = fruta + "s";
4    String otro = fruta.toUpperCase();
```

- Los métodos **toUpperCase** y **toLowerCase** devuelven un **String** convertido a mayúsculas y minúsculas, respectivamente.

Los Strings son inmutables

- Algo muy particular con respecto a los Strings es que "no se los puede modificar". Es decir, no es posible alterar su contenido.
- Obviamente, si quiero modificar el string que está guardado en una variable, siempre puedo cambiarlo por otro:

```
1    String fruta = "anana";
2    fruta = "b" + fruta;
3    fruta = fruta + "s";
4    String otro = fruta.toUpperCase();
```

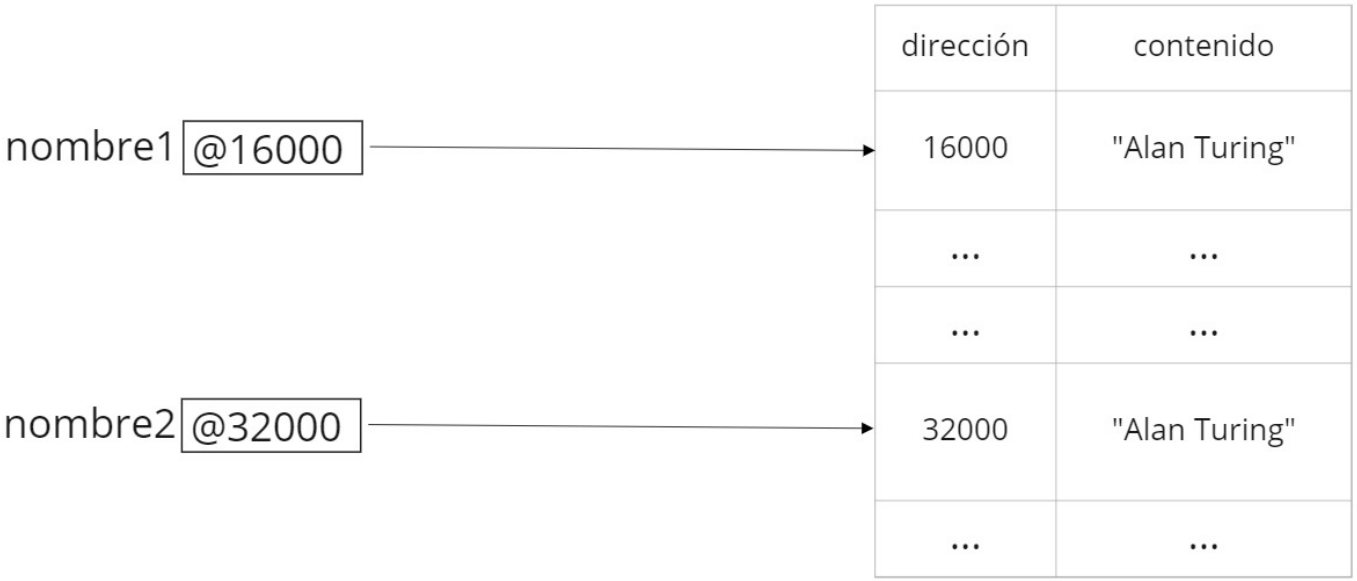
- Los métodos **toUpperCase** y **toLowerCase** devuelven un **String** convertido a mayúsculas y minúsculas, respectivamente.
- Suelen generar confusión, porque pareciera que modifican (mutan) el **String** sobre el que actúan, pero esto no es así.

Los Strings no se comparan así nomás

- Para ver si dos **String**s son iguales, se utiliza el método **equals**.

```
1    String nombre1 = "Alan Turing";
2    String nombre2 = "Ada Lovelace";
3
4    if (nombre1.equals(nombre2)) {
5        System.out.println("Los nombres son iguales.");
6    }
```

Los Strings no se comparan así nomás



nombre1 == nombre2 **es false**

nombre1.equals(nombre2) **es true**

Los Strings no se comparan así nomás

- **No** se debe utilizar el operador de comparación == en este caso!

Los Strings no se comparan así nomás

- **No** se debe utilizar el operador de comparación == en este caso!
- Las variables nombre1 y nombre2 contienen la posición de memoria de los **String**s involucrados, y no las secuencias de caracteres en sí mismas.

Los Strings no se comparan así nomás

- **No** se debe utilizar el operador de comparación == en este caso!

- Las variables nombre1 y nombre2 contienen la posición de memoria de los **String**s involucrados, y no las secuencias de caracteres en sí mismas.
- Entonces, al comparar nombre1 == nombre2, estamos determinando si están ubicadas en la misma posición de memoria, y no su contenido. Podría haber dos cadenas iguales ubicadas en diferentes posiciones de memoria. En ese caso serían iguales pero la comparación == nos daría falso.
- Para comparar el contenido de nombre1 y nombre2 debemos usar **nombre1.equals(nombre2)** o **nombre2.equals(nombre1)**

En el libro...

Los temas que vimos en esta clase los pueden encontrar en el **Capítulo 7** del libro.