

6.B. Cadenas de caracteres.

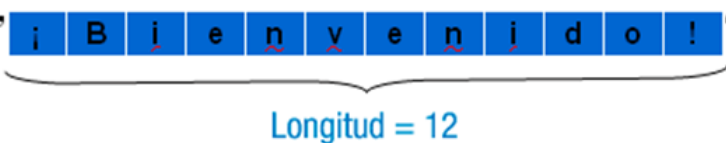
1. Cadenas de caracteres.

1.2. Operaciones avanzadas con cadenas de caracteres (II).

Vamos a continuar revisando las operaciones que se pueden realizar con cadenas. Como verás las operaciones a realizar se complican un poco a partir de ahora. En todos los ejemplos la variable `cad` contiene la cadena "¡Bienvenido!", como se muestra en las imágenes.

- `int length()`. Retorna un número entero que contiene la longitud de una cadena, resultado de contar el número de caracteres por la que está compuesta. Recuerda que un espacio es también un carácter.

`String cad="¡ B i e n v e n i d o !"`



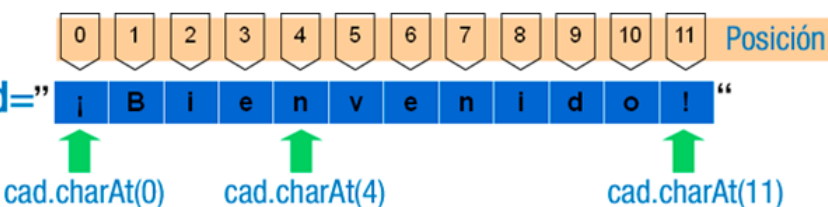
Longitud = 12

- `char charAt(int pos)`. Retorna el carácter ubicado en la posición pasada por parámetro. El carácter obtenido de dicha posición será almacenado en un tipo de dato `char`. Las posiciones se empiezan a contar desde el 0 (y no desde el 1), y van desde 0 hasta longitud - 1. Por ejemplo, el código siguiente mostraría por pantalla el carácter "v":

```
char t = cad.charAt(5);
```

```
System.out.println(t);
```

`String cad="¡ B i e n v e n i d o !"`

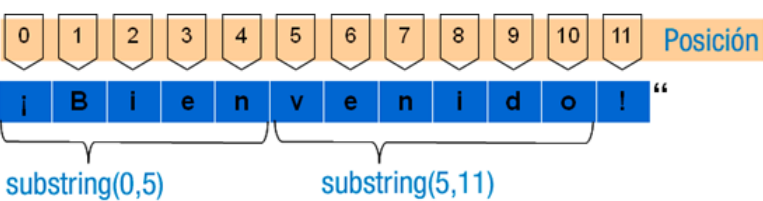


Posición

`cad.charAt(0)` `cad.charAt(4)` `cad.charAt(11)`

- `String substring(int beginIndex, int endIndex)`. Este método permite extraer una subcadena de otra de mayor tamaño. Una cadena compuesta por todos los caracteres existentes entre la posición `beginIndex` y la posición `endIndex - 1`. Por ejemplo, si pusiéramos `cad.substring(0,5)` en nuestro programa, sobre la variable `cad` anterior, dicho método devolvería la subcadena "¡Bien" tal y como se muestra en la imagen.

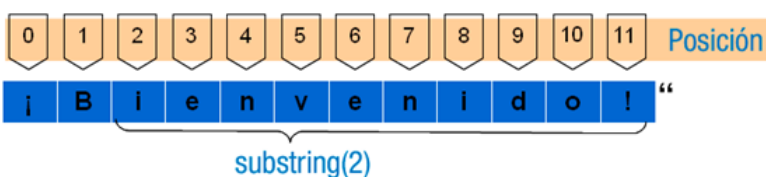
`String cad="¡ B i e n v e n i d o !"`



Posición

`substring(0,5)` `substring(5,11)`

`String cad="¡ B i e n v e n i d o !"`



Posición

`substring(2)`

- `String substring (int beginIndex)`. Cuando al método `substring` solo le proporcionamos un parámetro, extraerá una cadena que comenzará en el carácter con posición `beginIndex` e irá hasta el final de la cadena. En el siguiente ejemplo se mostraría por pantalla la cadena "ienvenido!":

```
String subcad = cad.substring(2);
```

```
System.out.println(subcad);
```

Otra operación muy habitual es la conversión de número a cadena y de cadena a número. Imagínate que un usuario introduce su edad. Al recoger la edad desde la interfaz de usuario, capturarás generalmente una cadena, pero, ¿cómo comprobas que la edad es mayor que 0?

Para poder realizar esa comprobación tienes que pasar la cadena a número. Empezaremos por ver como se convierte un número a cadena.

Los números generalmente se almacenan en memoria como números binarios, es decir, secuencias de unos y ceros con los que se puede operar (sumar, restar, etc.). No debes confundir los tipos de datos que contienen números (`int`, `short`, `long`, `float` y `double`) con las secuencias de caracteres que representan un número. No es lo mismo 123 que "123", el primero es un número y el segundo es una cadena formada por tres caracteres: '1', '2' y '3'.

Convertir un número a cadena es fácil desde que existe, para todas las clases Java, el método `toString()`. Gracias a ese método podemos hacer cosas como las siguientes:

```
String cad2="Número cinco: " + 5;
```

```
System.out.println(cad2);
```

El resultado del código anterior es que se mostrará por pantalla "Número cinco: 5", y no dará ningún error. Esto es posible gracias a que Java convierte el número 5 a su "clase envoltorio" (wrapper class) correspondiente (`Integer`, `Float`, `Double`, etc.), y después ejecuta automáticamente el método `toString()` de dicha clase.

Reflexiona

¿Cuál crees que será el resultado de poner `System.out.println("A"+5f)`? Pruébalo y recuerda: no olvides indicar el tipo de literal (f para los literales de números flotantes, y d para los literales de números dobles), así obtendrás el resultado esperado y no algo diferente.