ikastaroak.ulhi.net

PROG06.- Desarrollo de clases.

5-7 minutos

Sintaxis de las cadenas de formato y uso del método `format`.

En Java, el método estático `format` de la clase `String` permite formatear los datos que se muestran al usuario o la usuaria de la aplicación. El método `format` tiene los siguientes argumentos:

- Cadena de formato. Cadena que especifica cómo será el formato de salida, en ella se mezclará texto normal con especificadores de formato, que indicarán cómo se debe formatear los datos.
- Lista de argumentos. Variables que contienen los datos cuyos datos se formatearán. Tiene que haber tantos argumentos como especificadores de formato haya en la cadena de formato.

Los especificadores de formato comienzan siempre por "`%`", es lo que se denomina un carácter de escape (carácter que sirve para indicar que lo que hay a continuación no es texto normal, sino algo especial). El especificador de formato debe llevar como mínimo el símbolo "`%`" y un carácter que indica la conversión a realizar, por ejemplo "`%d`".

La conversión se indica con un simple carácter, y señala al método `format` cómo debe ser formateado el argumento. Dependiendo del tipo de dato podemos usar unas conversiones u otras. Veamos las conversiones más utilizadas:

Listado de conversiones más utilizada y ejemplos.

Tipo de conversión	Especificación de formato	Tipos de datos aplicables	Ejemplo	Resulta ejen

Valor lógico o booleano.	"`%b`" o "`%B`"	`Boolean` (cuando se usan otros tipos de datos siempre lo formateará escribiendo `true`).	`boolean b = true;`
			`String d = String.format("Resultado: %b", b);`
			`System.out.println(d);`
				Resultac `true`

Cadena de caracteres.	"`%s`" o "`%S`"	Cualquiera, se convertirá	`String cad = "hola mundo";` `String d =`	Resultac `hola m`

Tipo de conversión	Especificación de formato	Tipos de datos aplicables	Ejemplo	Resulta ejen
		el objeto a cadena si es posible (invocando el método `toString`).	`String.format("Resultado: %s", cad);` `System.out.println(d);`	

Entero decimal	`"%d"`	Un tipo de dato entero.	`int i = 10;`
			`String d = String.format("Resultado: %d", i);`
			`System.out.println(d);`
				Resultad

Número en notación científica	`"%e"` o `"%E"`	Flotantes simples o dobles.	`double i = 10.5;`
			`String d = String.format("Resultado: %E", i);`
			`System.out.println(d);`
				Resultad 1.0500

Número decimal	`"%f"`	Flotantes simples o dobles.	`double i = 10.5;`
			`String d = String.format("Resultado: %f", i);`
			`System.out.println(d);`
				Resultad 10,500

Número en notación científica o decimal (lo más corto)	`"%g"` o `"%G"`	Flotantes simples o dobles. El número se mostrará como decimal o en notación científica dependiendo de lo que sea mas corto.	`double i = 10.5;`
			`String d = String.format("Resultado: %g", i);`
			`System.out.println(d);`
				Resultad 10.500

Ahora que ya hemos visto alguna de las conversiones existentes (las más importantes), veamos algunos modificadores que se le pueden aplicar a las conversiones, para ajustar como queremos que sea la salida. Los modificadores se sitúan entre el carácter de escape ("`%`") y la letra que indica

el tipo de conversión (d, f, g, etc.).

Podemos especificar, por ejemplo, el número de caracteres que tendrá como mínimo la salida de una conversión. Si el dato mostrado no llega a ese ancho en caracteres, se rellenará con espacios (salvo que se especifique lo contrario):

El hecho de que esté entre corchetes significa que es opcional. Si queremos por ejemplo que la salida genere al menos 5 caracteres (poniendo espacios delante) podríamos ponerlo así:

```
String.format("%5d", 10);
```

Se mostrará el "10" pero también se añadirán 3 espacios delante para rellenar. Este tipo de modificador se puede usar con cualquier conversión.

Cuando se trata de conversiones de tipo numéricas con decimales, solo para tipos de datos que admitan decimales, podemos indicar también la precisión, que será el número de decimales mínimos que se mostrarán:

```
%[Ancho][.Precisión]Conversión
```

Como puedes ver, tanto el ancho como la precisión van entre corchetes, los corchetes no hay que ponerlos, solo indican que son modificaciones opcionales. Si queremos, por ejemplo, que la salida genere 3 decimales como mínimo, podremos ponerlo así:

```
String.format("%.3f", 4.2f);
```

Como el número indicado como parámetro solo tiene un decimal, el resultado se completará con ceros por la derecha, generando una cadena como la siguiente: "`4,200`".

Una cadena de formato puede contener varios especificadores de formato y varios argumentos. Veamos un ejemplo de una cadena con varios especificadores de formato:

```
String producto = "Lavadora";

int cantidad = 10;

double precio = 302.4;

double total = cantidad * precio;

String output = String.format("Producto: %s; Unidades: %d;
Precio por unidad: %.2f €; Total: %.2f €",
                producto, cantidad, precio, total);

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

Cuando el orden de los argumentos es un poco complicado, porque se reutilizan varias veces en la cadena de formato los mismos argumentos, se puede recurrir a los índices de argumento. Se trata de especificar la posición del argumento a utilizar, indicando la posición del argumento (el primer argumento sería el 1 y no el 0) seguido por el símbolo del dólar ("`$`"). El índice

se ubicaría al comienzo del especificador de formato, después del porcentaje, por ejemplo:

```
int i = 10;
int j = 20;
String salida = String.format("%1$d multiplicado por %2$d (%1$d x %2$d) es %3$d", i, j, i * j);
System.out.println(salida);
```

El ejemplo anterior mostraría por pantalla la cadena "1`0 multiplicado por 20 (10 x 20) es 200`". Los índices de argumento se pueden usar con todas las conversiones, y es compatible con otros modificadores de formato (incluida la precisión).