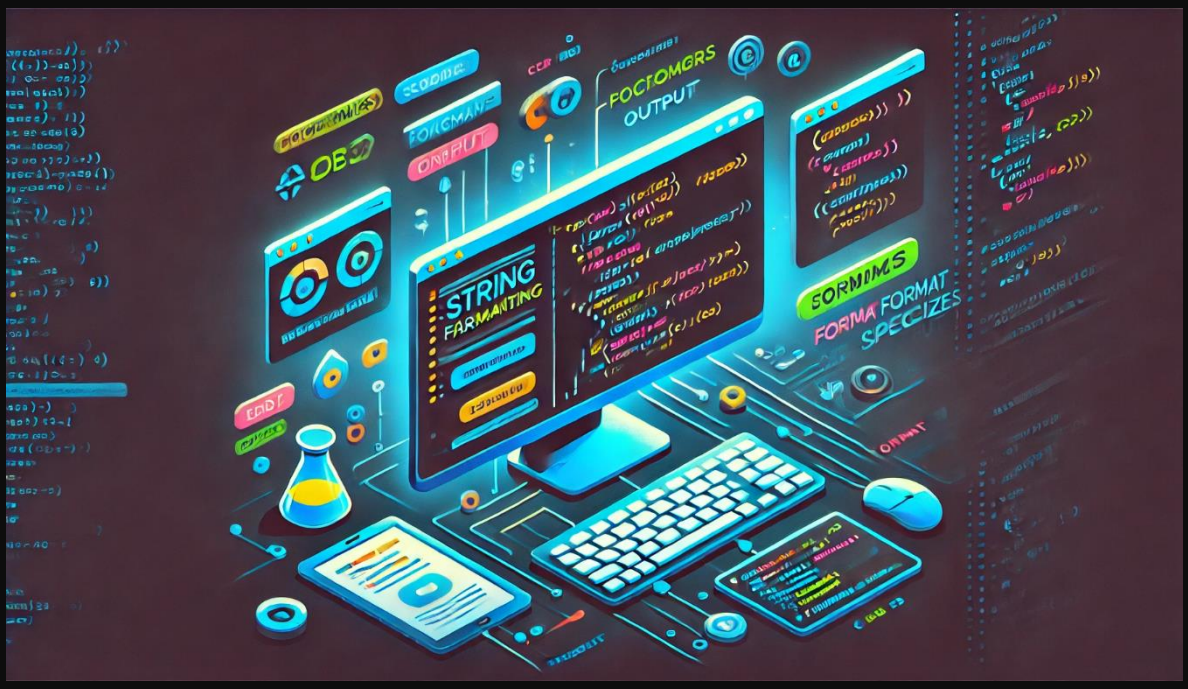
Formateo de Cadenas en Java



Lección: Formateo de Números y Cadenas en Java

En esta lección, aprenderemos a dar formato a números y cadenas en Java utilizando tanto cadenas con cadenas simples, así como bloques de texto (text blocks). Veremos cómo usar métodos como `String.format` y `System.out.printf` para formatear la salida y cómo incluir saltos de línea y otros caracteres especiales en las cadenas.

1. Formateo Básico de Números y Cadenas

Uso de `String.format`

`String.format` permite crear una nueva cadena con formato especificado.

```
public class FormateoEjemplo {
    public static void main(String[] args) {
        var nombre = "Juan";
        var edad = 35;
        var salario = 12345.6789;

        // Formateo con comillas dobles
        var mensaje =
            String.format("Nombre: %s, Edad: %d, Salario: %.2f", nombre, edad, salario);
        System.out.println(mensaje);
    }
}
```

Uso de `System.out.printf`

`System.out.printf` permite formatear la salida directamente en la consola.

```
public class FormateoEjemplo {
    public static void main(String[] args) {
        var nombre = "Juan";
        var edad = 35;
        var salario = 12345.6789;

        // Formateo con comillas dobles
        System.out.printf("Nombre: %s, Edad: %d, Salario: %.2f%n", nombre, edad, salario);
    }
}
```

2. Uso de Bloques de Texto (Text Blocks)

Los bloques de texto se introdujeron en Java 13 como una vista previa y se estabilizaron en Java 15. Permiten escribir cadenas de texto multilínea de manera más fácil y legible.

Ejemplo con Text Blocks

```
public class FormateoEjemplo {
    public static void main(String[] args) {
        var nombre = "Juan";
        var edad = 35;
        var salario = 12345.67;

        // Formateo con text block
        var mensaje = """
                Nombre: %s
                Edad: %d
                Salario: %.2f
                """.formatted(nombre, edad, salario);
        System.out.println(mensaje);
    }
}
```

3. Caracteres Especiales y Saltos de Línea

Uso de Caracteres Especiales en Formateo

Los caracteres especiales como el salto de línea (`\n`), el tabulador (`\t`), y otros, se pueden usar para mejorar la legibilidad de la salida.

```
public class FormateoEjemplo {
    public static void main(String[] args) {
        var nombre = "Juan";
        var edad = 35;
        var salario = 12345.67;

        // Uso de caracteres especiales
        var mensaje =
             String.format("Nombre:\t%s%nEdad:\t%d%nSalario:\t%.2f", nombre, edad, salario);
        System.out.println(mensaje);
    }
}
```

4. Ejemplos Complejos con Bloques de Texto

Formateo Complejo con Text Blocks

```
public class FormateoEjemplo {
    public static void main(String[] args) {
        var nombre = "Juan";
        var edad = 35;
        var salario = 12345.67;

        // Formateo complejo con text block
        var mensaje = """
                Información del Empleado:
                -------------------------
                Nombre:   %s
                Edad:     %d
                Salario:  %.2f
                -------------------------
                """.formatted(nombre, edad, salario);
        System.out.println(mensaje);
    }
}
```

Resumen de Especificadores de Formato

- `%s`: Cadena de texto.
- `%d`: Entero decimal.
- `%f`: Número de punto flotante.
- `%n`: Nueva línea independiente de la plataforma.
- `%t`: Fecha y hora (requiere especificadores adicionales para el formato específico).

Ejercicio Práctico

Escribe un programa que solicite el nombre, la edad y el salario de un usuario, y luego imprima esta información utilizando tanto comillas dobles como bloques de texto, incluyendo formateo adecuado para los números y saltos de línea.

Solución del Ejercicio

```
import java.util.Scanner;

public class FormateoEjercicio {
    public static void main(String[] args) {
        var consola = new Scanner(System.in);

        // Solicitar datos del usuario
        System.out.print("Ingrese su nombre: ");
        var nombre = consola.nextLine();
        System.out.print("Ingrese su edad: ");
        var edad = Integer.parseInt(consola.nextLine());
        System.out.print("Ingrese su salario: ");
        var salario = Double.parseDouble(consola.nextLine());

        // Formateo con comillas dobles
        var mensaje1 =
```

```
                String.format("Nombre: %s%nEdad: %d%nSalario: %.2f", nombre, edad, salario);
        System.out.println(mensaje1);

        // Formateo con text block
        var mensaje2 = """
                Información del Usuario:
                ------------------------
                Nombre:   %s
                Edad:     %d
                Salario:  %.2f
                ------------------------
                """.formatted(nombre, edad, salario);
        System.out.println(mensaje2);
    }
}
```

Ejemplo Adicional: Asegurar Formato de 4 Dígitos y 2 Dígitos en Punto Flotante

Para asegurar que un número entero tenga siempre 4 dígitos (rellenando con ceros a la izquierda si es necesario) y que un número de punto flotante tenga siempre 2 dígitos decimales, utilizamos el siguiente formato:

Uso de `String.format`

```
public class FormateoNumeros {
    public static void main(String[] args) {
        var numero = 42;
        var valor = 123.456;

        // Asegurar 4 dígitos para el número entero
        var numeroFormateado = String.format("%04d", numero);
        // Asegurar 2 dígitos decimales para el número de punto flotante
        var valorFormateado = String.format("%.2f", valor);

        System.out.println("Número formateado (4 dígitos): " + numeroFormateado);
        System.out.println("Valor formateado (2 dígitos decimales): " + valorFormateado);
    }
}
```

Uso de `System.out.printf`

```
public class FormateoNumeros {
    public static void main(String[] args) {
        var numero = 42;
        var valor = 123.456;

        // Asegurar 4 dígitos para el número entero
        System.out.printf("Número formateado (4 dígitos): %04d%n", numero);
        // Asegurar 2 dígitos decimales para el número de punto flotante
        System.out.printf("Valor formateado (2 dígitos decimales): %.2f%n", valor);
    }
}
```

Uso de Text Blocks

```
public class FormateoNumeros {
    public static void main(String[] args) {
        var numero = 42;
        var valor = 123.456;
```

```
        // Formateo con text block
        var mensaje = """
                Número formateado (4 dígitos): %04d
                Valor formateado (2 dígitos decimales): %.2f
                """.formatted(numero, valor);
        System.out.println(mensaje);
    }
}
```

Resumen de Especificadores de Formato

- `%s`: Cadena de texto.
- `%d`: Entero decimal.
- `%04d`: Entero decimal con al menos 4 dígitos, rellenado con ceros a la izquierda si es necesario.
- `%f`: Número de punto flotante.
- `%.2f`: Número de punto flotante con 2 dígitos decimales.
- `%n`: Nueva línea independiente de la plataforma.
- `%t`: Fecha y hora (requiere especificadores adicionales para el formato específico).

Desglose del Especificador de Formato `%04d`

1. **`%`**:
   - Este símbolo indica el comienzo de un especificador de formato. Todos los especificadores de formato en `String.format` y `System.out.printf` comienzan con `%`.
2. **`0`**:
   - Este dígito indica que el número debe ser rellenado con ceros si no alcanza la longitud mínima especificada.
   - En lugar de rellenar con espacios (que es el comportamiento predeterminado), los ceros se añadirán a la izquierda del número para asegurar que la longitud total sea la especificada.
3. **`4`**:
   - Este dígito especifica la longitud mínima del campo, es decir, el número debe tener al menos 4 dígitos.
   - Si el número tiene menos de 4 dígitos, se rellenará con ceros a la izquierda hasta alcanzar la longitud de 4 caracteres. El valor se puede adecuar a la cantidad de dígitos que se desean rellenar.
4. **`d`**:
   - Este carácter indica que el tipo de dato que se va a formatear es un entero decimal.

Conclusión

Esta lección cubre cómo formatear números y cadenas en Java utilizando `String.format`, `System.out.printf`, y bloques de texto (text blocks), proporcionando ejemplos claros y concisos para mejorar la legibilidad y el manejo de la salida en tus programas Java.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx