



# Java Foundations

**3-3**

**Datos textuales**

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

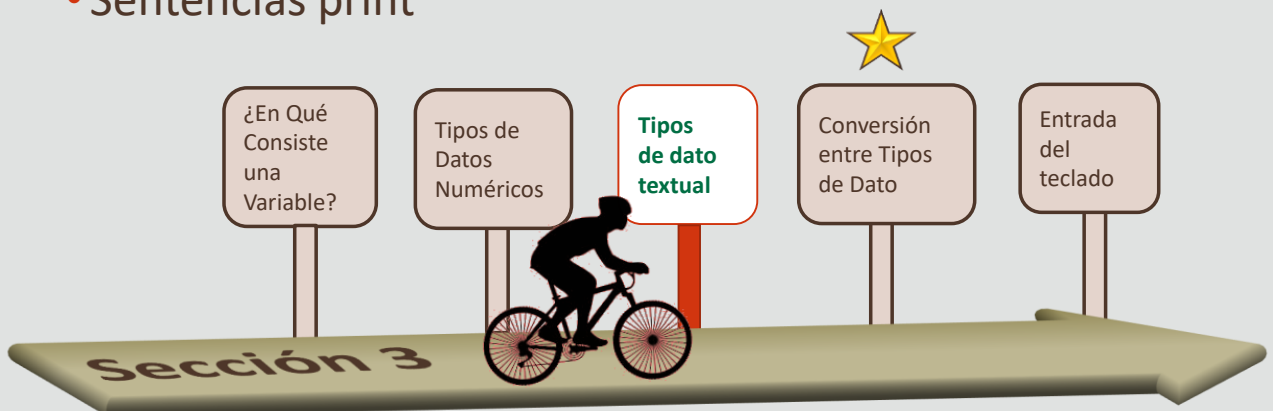
# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Usar el tipo de dato char
  - Utilizar cadenas
  - Concatenar cadenas
  - Comprender secuencias de escape
  - Comprender mejor las sentencias print



# Temas

- **Caracteres y cadenas**
- Concatenación de cadenas
- Combinación de cadenas y números
- Sentencias print



**ORACLE**  
Academy

JFo 3-3  
Datos textuales


Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

## Tipo primitivo textual

- El único tipo de dato textual primitivo es `char`
- Se utiliza para un único carácter (16 bits)

- Ejemplo:

```
-char shirtSize = 'M';
```



Las comillas simples se pueden utilizar con valores literales de `char`

Se trata de otro tipo de dato que se utiliza para almacenar y manipular datos como caracteres únicos. El tipo primitivo `char` tiene un tamaño de 16 bits. Al asignar un valor literal a una variable `char`, debe colocar el carácter entre comillas simples, como se muestra en el código que aparece en el ejemplo.

## Concatenación de caracteres

- Puede unir caracteres para crear frases
- Aquí le mostramos una forma poco eficaz de hacerlo
- Se necesita una línea de código para cada letra de la frase

```
char letter1 = 'H';  
char letter2 = 'e';  
char letter3 = 'l';  
char letter4 = 'l';  
char letter5 = 'o';  
//Long sentences would be painful to code  
System.out.println(letter1 +letter2 +letter3 +letter4 +letter5);
```

## Concatenación eficaz de caracteres

- A continuación se muestra una mejor forma de hacerlo
  - Solo se necesita una línea para toda la frase:

```
String greeting = "Hello World!";  
//Notice the double quotes  
System.out.println(greeting);
```

## Caracteres frente a Cadenas

- Los tipos char son para caracteres únicos
  - Use comillas simples



```
char shirt1Size = 'S';  
char shirt2Size = 'M';  
char shirt3Size = 'L';
```

- Los tipos char no pueden admitir varios caracteres



```
char shirt4Size = 'XL';  
char shirt5Size = "XXL";
```



# Caracteres frente a Cadenas

- Las cadenas pueden admitir varios caracteres
  - Utilice comillas dobles



```
String shirt6Size = "XXXL";
```

# Primitivos

Tipo	Longitud	Datos
boolean	1 bits	true / false
byte	8 bits	Enteros
short	16 bits	Enteros
int	32 bits	Enteros
long	64 bits	Enteros
float	32 bits	Números de coma flotante
double	64 bits	Números de coma flotante
char	16 bits	Caracteres únicos

*¿Dónde están las cadenas?*

**ORACLE**  
Academy

JFo 3-3  
Datos textuales

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

10

# Investiguemos

- ¿Podemos descubrir más diferencias entre los tipos char y String?

```
char shirt3Size = 'L';  
String shirt6Size = "XXXL";
```

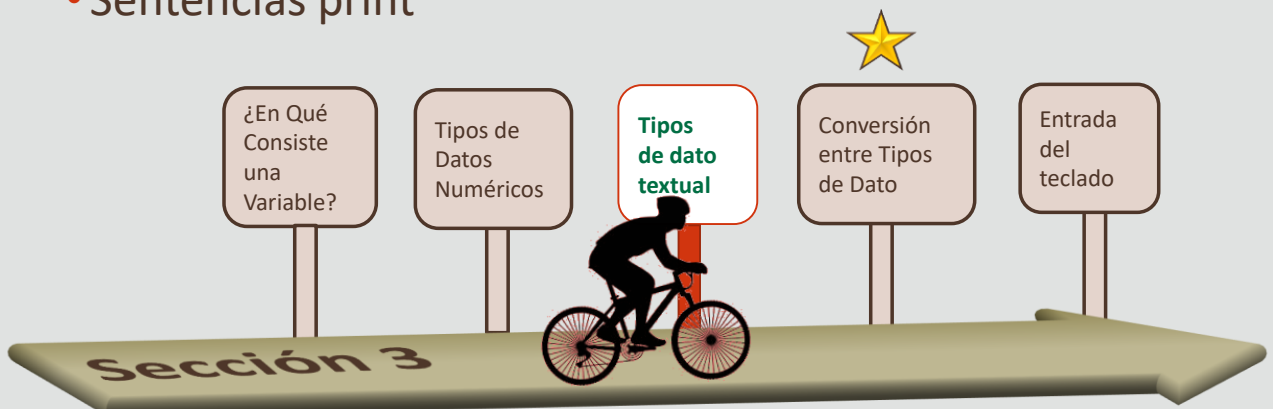
- 1. char cambia a azul
  - char es una palabra clave para un tipo de dato primitivo
  - Las palabras clave se vuelven azules en NetBeans
- 2. String aparece en mayúsculas
  - Las cadenas son objetos, no datos primitivos
  - Los tipos de objetos se capitalizan por convención

# Las Cadenas Son Objetos

- Java viene con una clase String que ofrece información sobre:
  - Las propiedades de la cadena
  - Los comportamientos de la cadena
- Las cadenas son objetos especiales
  - Las cadenas se utilizan de forma diferente que la mayoría de los objetos
- Se facilita más información al respecto en futuras secciones:
  - Los objetos pueden tener valores primitivos como propiedades
  - Los objetos pueden tener objetos como propiedades, como las cadenas
  - Los objetos no se guardan de igual modo que los valores primitivos en la memoria

# Temas

- Caracteres y cadenas
- **Concatenación de cadenas**
- Combinación de cadenas y números
- Sentencias print



**ORACLE**  
Academy

JFo 3-3  
Datos textuales

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

13

# Declaración e Inicialización de Cadenas

- Declare y asigne los valores String como haría con cualquier otro valor primitivo

```
//One variable declared and initialized
int intVar = 300;
String stringVar = "Three Hundred";

//Many variables declared
int x, y, z;
String xString, yString, zString;

//A declared variable is initialized later
x = 1;
xString = "One";
```

## Variable de Cadena frente a Literales de Cadena

```
String stringVariable = "This is a String literal.>";
```

*Variable* *Literal*

- Se puede crear una cadena mediante la combinación de varios literales de cadena:

```
String combinedLiterals = "I want to" + " buy a shirt.>";
```

- Se puede crear una cadena combinando variables:

```
String var1 = "This shirt has";  
String var2 = " too many buttons.>";  
String combinedVariables = var1 + var2;
```

## Concatenación de cadenas

- La combinación de varias cadenas es lo que se denomina concatenación
- Las cadenas se pueden combinar mediante el operador +
  - stringVariable1 + stringVariable2
  - stringVariable1 + “String literal”
  - stringVariable1 + “String literal” + stringVariable2

```
String greet1 = "Hello";  
String greet2 = "World";  
String message1 = greet1 + " " + greet2 + "!";  
String message2 = greet1 + " " + greet2 + " " + 2020 + "!";
```

La combinación de varias cadenas se denomina "concatenación." Se pueden concatenar variables de cadenas entre sí. También puede concatenar un literal de cadena a una variable de cadena.

Puede concatenar cualquier número de variables de cadena y literales de cadena para alcanzar su objetivo.



# Resultados de la Concatenación de Cadenas

- Ejemplo de concatenación:

```
String greet1 = "Hello";  
String greet2 = "World";  
String message1 = greet1 + " " + greet2 + "!";  
String message2 = greet1 + " " + greet2 + " " + 2020 + "!";
```

- Puede concatenar cadenas en una sentencia de impresión:

```
System.out.println(message2);  
System.out.println(greet1 + " " + greet2 + " " + 2020 + "!");
```

- Resultado:

```
Hello World 2020!  
Hello World 2020!
```

En los ejemplos de la diapositiva, se muestran dos variaciones de datos de cadenas de impresión utilizando `System.out.println`.

En el primer ejemplo, se imprimirá la variable `message2` que vio en la diapositiva anterior.

En el segundo ejemplo, la expresión que contiene la concatenación de variables, así como los literales de la cadena, se puede utilizar con los paréntesis del método. El motor de tiempo de ejecución termina la concatenación antes de que se ejecute el método `println`.

Como puede ver, los resultados de ambas llamadas al método son los mismos.

## Situación del Ejercicio 1

- Recordemos el catálogo de ropa de Duke's Choice:
- En dicho ejemplo, había una clase ShoppingCart
- En este ejercicio, vamos a ver por encima algunos comportamientos y propiedades de ShoppingCart
- Propiedades de ShoppingCart:  Aparecen representadas como cadenas en este ejercicio
  - Quién es el propietario
  - Elementos que contiene
  - Mensaje/descripción del carro
- Comportamientos de ShoppingCart:
  - Imprime el mensaje



El ejemplo del catálogo de Duke's Choice se vio en la sección 2 de la lección 3.



## Ejercicio 1, Parte 1

- Importe y edite el proyecto `ShoppingCart01`
- Declare e inicialice la variable de cadena `custName`
- Declare e inicialice la variable de cadena `itemDesc`
- Declare una variable de cadena `message`



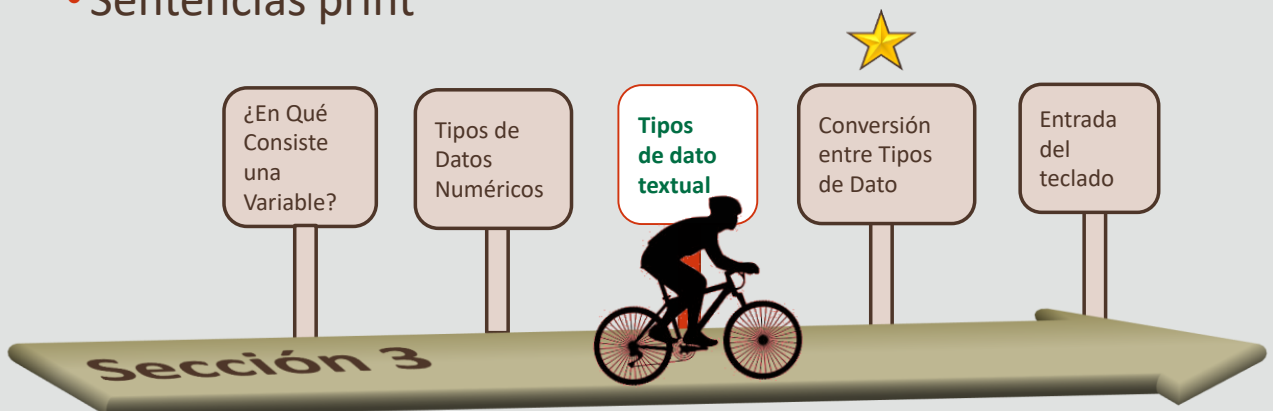
## Ejercicio 1, Parte 2

- Asigne a message un valor concatenado que incluya custName, itemDesc y una literal de cadena, que dé lugar a una frase completa:
  - (ejemplo: “Alex wants to purchase a Shirt”)
- Imprima el mensaje
- El programa debería generar un resultado similar a este:

```
Alex wants to purchase a Shirt
```

# Temas

- Caracteres y cadenas
- Concatenación de cadenas
- **Combinación de cadenas y números**
- Sentencias print



**ORACLE**  
Academy

JFo 3-3  
Datos textuales

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

21

## Combinación de cadenas y números

- Las cadenas pueden contener números:

```
String totalPrice = "Total: $" +3;  
System.out.println(totalPrice);    //Total: $3
```

- Pero tenga cuidado al intentar realizar cálculos:

```
String totalPrice = "Total: $" +3 +2 +1;  
System.out.println(totalPrice);    //Total: $321
```

- Utilice paréntesis para los números:

```
String totalPrice = "Total: $" +(3 +2 +1);  
System.out.println(totalPrice);    //Total: $6
```

## Situación del Ejercicio 2

- Pregunta: ¿cuánto deberán pagar los clientes, conforme van rellendo el carro?
- Tenemos que ver los productos del carro en más detalle para poder responder esta pregunta



## Situación del Ejercicio 2

- ShoppingCart debe conocer las propiedades siguientes:
  - Item price:
  - Tipo de impuesto sobre las ventas
  - Número de productos
  - Cálculo del precio total de todos los productos del carro
- ShoppingCart necesita los comportamientos siguientes:
  - Imprimir un mensaje con el total







## Ejercicio 2, Parte 1

- Importe y edite el proyecto `ShoppingCart02`
- Declare e inicialice los campos numéricos:
  - price (double)
  - tax (double)
  - quantity (int)
- Declare un double `totalPrice`:
  - Asigne un valor, que se debe calcular a partir del price, tax y quantity



## Ejercicio 2, Parte 2

- Cambie el mensaje para que incluya la cantidad
  - (ejemplo: “Alex wants to purchase 2 Shirts”)
- Imprima otro mensaje en el que se muestre el coste total
- El programa debería generar un resultado similar a este:

```
Alex wants to purchase 2 Shirts  
Total cost with tax is: $25.78
```

## Notas sobre el Ejercicio

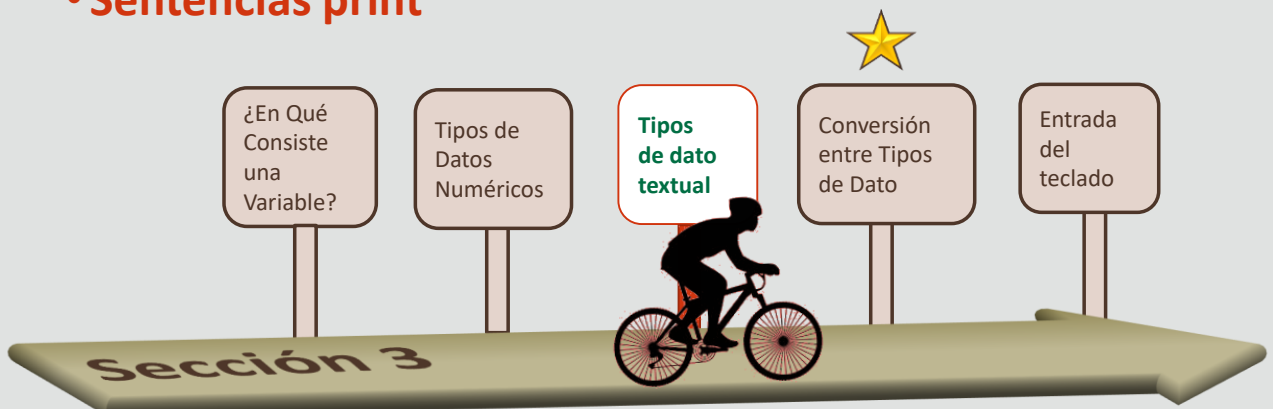
- No es una práctica recomendada representar las propiedades y los comportamientos de los objetos utilizando exclusivamente el método principal
- Pasamos esto por alto en esta sección para poder centrarnos en la manipulación de datos
- Nos esforzaremos por acatar mejor las normas en la próxima sección

¡Ahhhhh! ¿Por qué no sigue las reglas?



# Temas

- Caracteres y cadenas
- Concatenación de cadenas
- Combinación de cadenas y números
- **Sentencias print**



# Caracteres Especiales en las Cadenas

- ¿Recuerdas cuando imprimimos el gato?
- La doble barra invertida no se imprimía:
  - Solo se imprimían las barras invertidas sueltas
  - Why?

```
3 public class Text01 {
4     public static void main(String[] args) {
5         System.out.println("  /\ \      /\ \ ");
6         System.out.println(" /  /\ \____/  /\ \ ");
7         System.out.println(" /                    /\ \ ");
8         System.out.println("(  /\ \              /\ \ )");
9         System.out.println("====      V      =====");
10        System.out.println("===== ( _ | _ ) =====");
11        System.out.println(" (                ) ");
12        System.out.println(" (                ) ");
13    }
14 }
```

## Secuencia de Escape

- Un carácter precedido por una barra invertida se denomina secuencia de escape y tiene un significado especial para el compilador
- En la tabla de la siguiente diapositiva, aparecen secuencias de escape de Java

## Secuencia de Escape

Secuencia de Escape	Description
<code>\t</code>	Insertar un nuevo separador
<code>\b</code>	Insertar un retroceso
<code>\n</code>	Insertar una nueva línea
<code>\r</code>	Insertar un retorno de carro
<code>\f</code>	Insertar un salto de hoja
<code>\'</code>	Insertar un carácter de comilla simple
<code>\"</code>	Insertar un carácter de comillas dobles
<code>\\</code>	Insertar un carácter de barra invertida

## Secuencia de Escape: Ejemplo

- Si desea poner entre comillas una sección que ya está entre comillas, debe utilizar la secuencia de escape, `\`, en las comillas de dentro

– Para imprimir la frase...

```
The cat said "Meow!" to me.
```

– Debe escribir

```
System.out.println("The cat said \"Meow!\" to me.");
```



## Sentencias print

- Es posible que el texto que se escriba en una línea diferente, no se imprima en una nueva línea:

```
System.out.println("This is the first line."  
+ " This is NOT the second line.");
```

### Resultado:

```
This is the first line.This is NOT the second line.
```

- Las secuencias de escape le pueden ayudar a dar formato a los resultados:

```
System.out.println("This is the first line.\n"  
+ "This is the second line.");
```

### Resultado:

```
This is the first line.  
This is the second line.
```

## Más Sentencias de Impresión

- Hay dos métodos importantes de impresión:

```
System.out.println("Print Line method");  
System.out.print("Print method");
```

- println funciona como si estuviera poniendo \n automáticamente al final de la sentencia
- Las dos sentencias siguientes generan los mismos resultados:

```
System.out.println("Printing ");  
System.out.print("Printing \n");
```

## println() frente a print:

- println() crea una línea de forma automática:

```
System.out.println("This is the first line.");  
System.out.println("This is the second line.");
```

### Resultado:

This is the first line  
This is the second line

- print() no crea una línea de forma automática:

```
System.out.print("This is the first line.");  
System.out.print("This is NOT the second line.");
```

### Resultado :

This is the first line This is NOT the second line.

# Método Abreviado de NetBeans

Método de impresión	¿Con qué frecuencia voy a utilizar esto?
<code>System.out.println()</code>	A menudo
<code>System.out.print("@");</code>	No tan a menudo

- `System.out.println()` se utiliza con mucha frecuencia
- Pero su configuración requiere escribir mucho
- Netbeans ofrece un método abreviado:
  - Escriba `sout`

`sout`

- Presione Tab

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

# Imprimir Mucho Texto, Opción 1

- En función de lo que desee imprimir, puede que prefiera:
  - Dividir una sola sentencia de impresión en varias líneas en NetBeans:

```
System.out.println("This is the first line."
    + "This is the still the first line."
    + "It's just that the line is very long "
    + "and I can't see it all in NetBeans."
    + "\n" + "This is the second line."
    + "\n" + "This is the third line.");
```

–OR

## Imprimir Mucho Texto, Opción 2

- Utilizar muchas sentencias de impresión:

```
System.out.println("This is the first line.");  
System.out.println("This is the second line.");  
System.out.println("This is the third line.");  
System.out.println("This is the fourth line.");
```

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Usar el tipo de dato char
  - Utilizar cadenas
  - Concatenar cadenas
  - Comprender secuencias de escape
  - Comprender mejor las sentencias print



