

# 6

## Trabajar con objetos

# Trabajar con objetos: Introducción

Se accede a los objetos a través de referencias.

- Los objetos son versiones instanciadas de su clase.
- Los objetos constan de atributos y operaciones:
  - En Java, son campos y métodos.

# Acceso a objetos mediante una referencia



La cámara es como el objeto al que se accede a través de la referencia (control remoto).



El control remoto es como la referencia utilizada para acceder a la cámara (objeto).

fundacion@proydesa.org) has a  
this Student Guide.

## Class Shirt

```
public class Shirt {  
    public int shirtID = 0; // Default ID for the shirt  
    public String description =  
        "-description required-"; // default  
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 0.0; // Default price all items  
    // This method displays the details for an item  
    public void display() {  
        System.out.println("Item ID: " + shirtID);  
        System.out.println("Item description:" +  
            description);  
        System.out.println("Color Code: " + colorCode);  
        System.out.println("Item price: " + price);  
    } // end of display method  
} // end of class
```

# Trabajar con variables de referencia de objetos

Declaración:

```
Classname identifier;
```

Instanciación:

```
new Classname();
```

Este fragmento de código crea el objeto.

Asignación:

```
Object reference = new Classname();
```

Identificador del paso de declaración

Operador de asignación

Para realizar la asignación a una referencia, la creación y la asignación deben estar en la misma sentencia.

# Declaración e inicialización: Ejemplo

1

Declarar una referencia para el objeto.

2

Crear la instancia del objeto.

```
Shirt myShirt;
```

```
myShirt = new Shirt();
```

3

Asignar el objeto a la variable de referencia.



# Trabajar con referencias de objetos

Declarar e  
inicializar la  
referencia.

```
Shirt myShirt = new Shirt();
```

```
int shirtId = myShirt.shirtId;
```

Obtener el valor  
del campo  
shirtId del  
objeto.

```
myShirt.display();
```

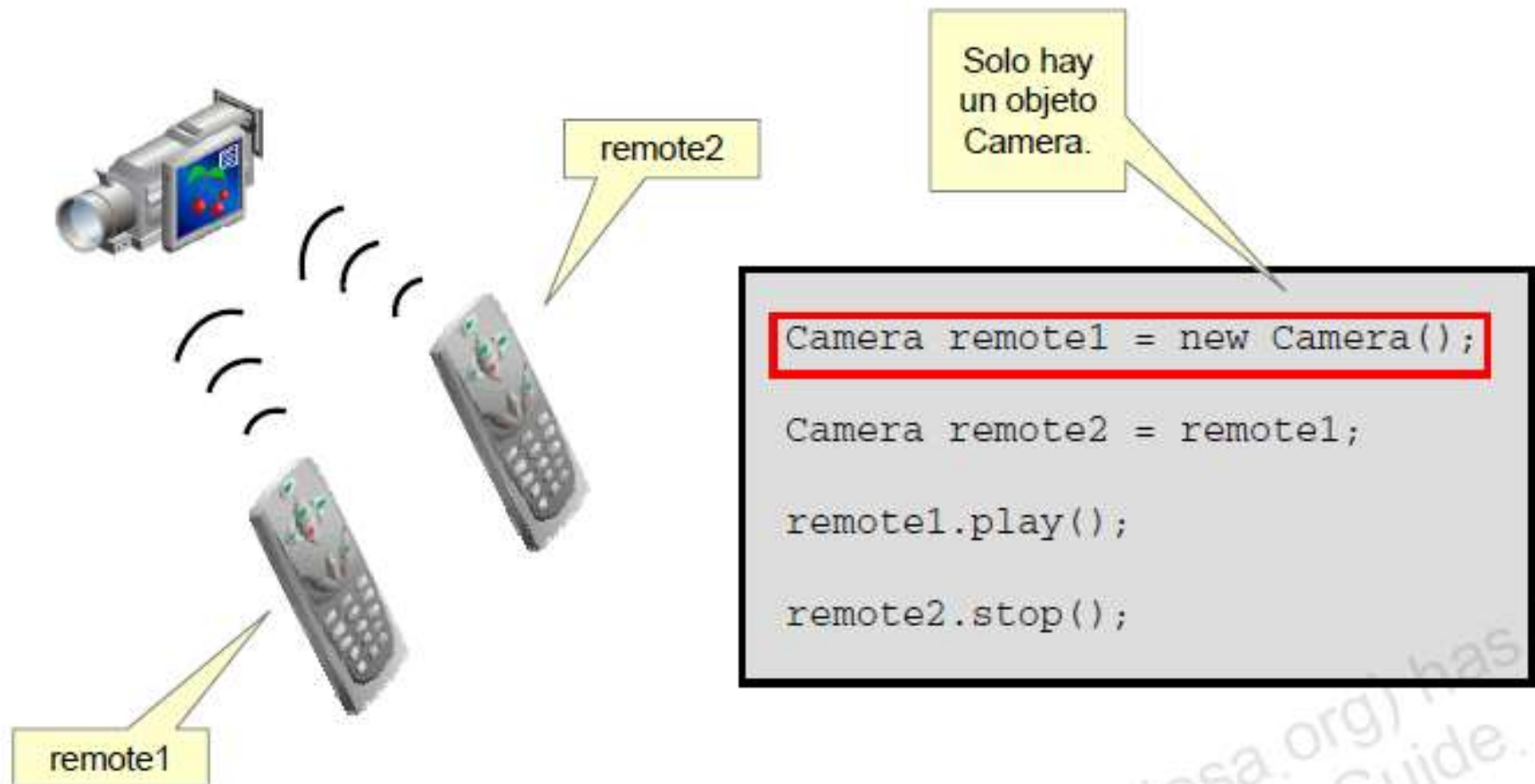
Llamar al  
método  
display()  
del objeto.

# Trabajar con referencias de objetos

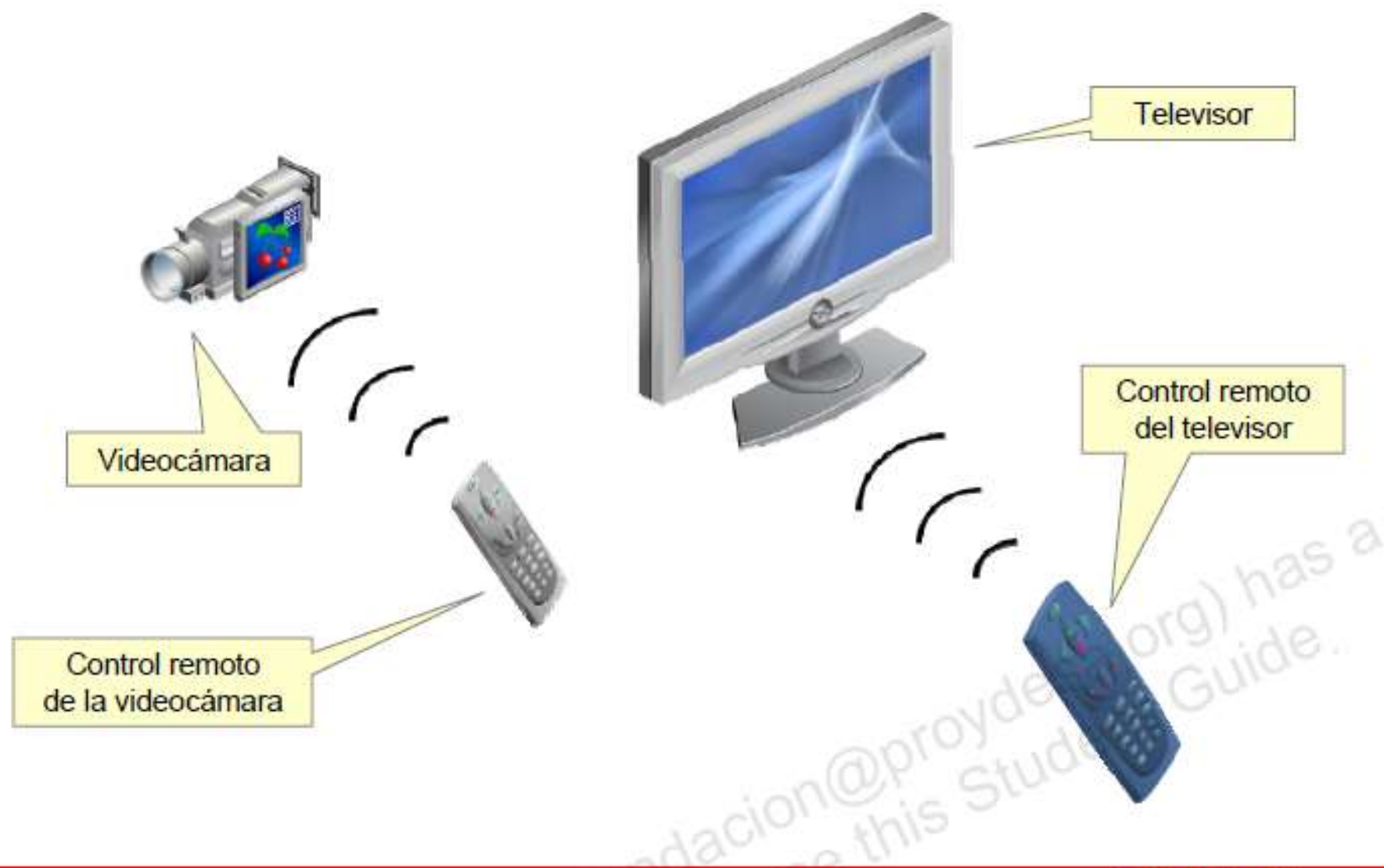




# Trabajar con referencias de objetos



# Referencias a diferentes objetos



# Referencias a diferentes tipos de objetos

El tipo de referencia es Shirt.

El tipo de objeto es Shirt.

```
Shirt myShirt = new Shirt();  
myShirt.display();
```

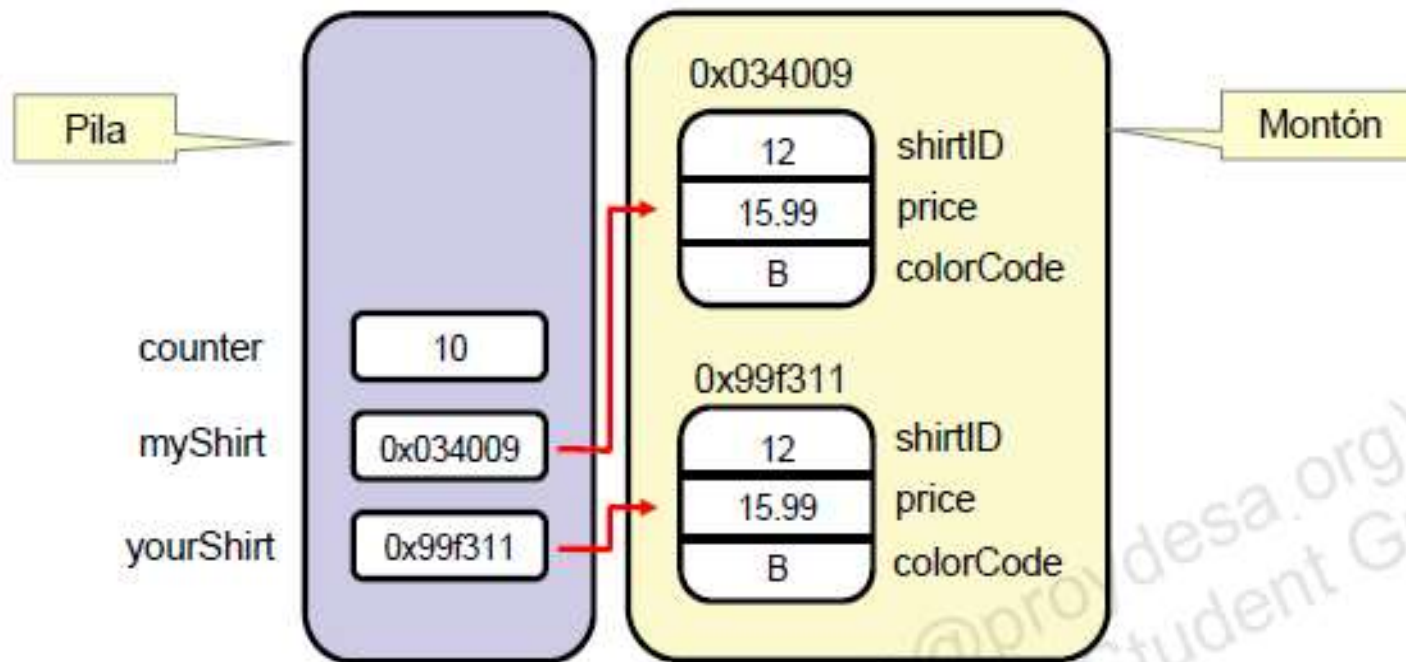
```
Trousers myTrousers = new Trousers();  
myTrousers.display();
```

El tipo de referencia es Trousers.

El tipo de objeto es Trousers.

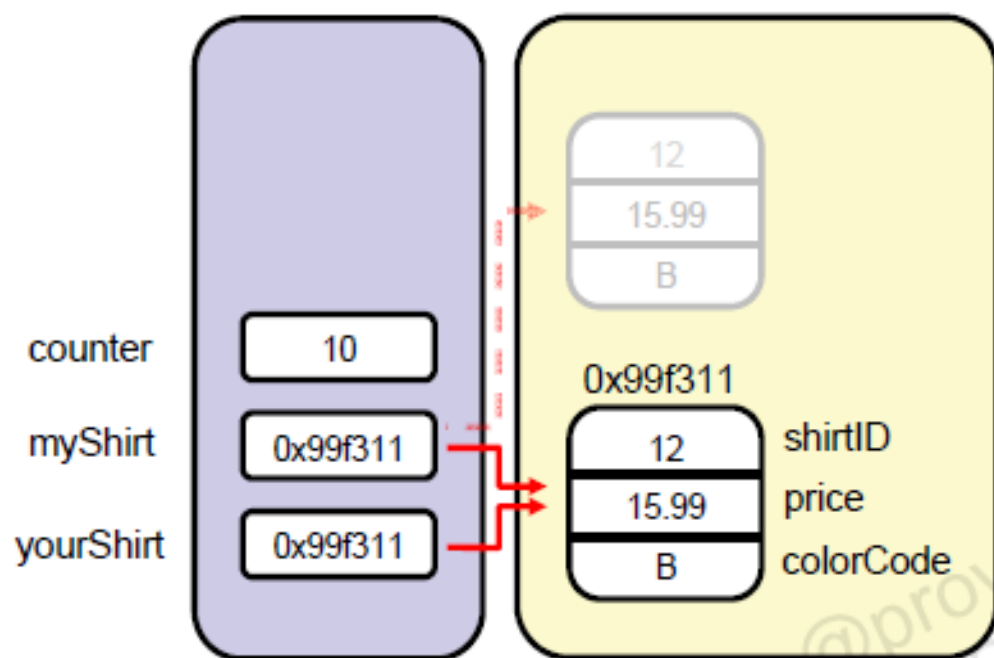
# Referencias y objetos en memoria

```
int counter = 10;  
Shirt myShirt = new Shirt();  
Shirt yourShirt = new Shirt();
```



## Asignación de una referencia a otra

```
myShirt = yourShirt;
```



## Dos referencias, un objeto

Fragmento de código:

```
Shirt myShirt = new Shirt();  
Shirt yourShirt = new Shirt();  
  
myShirt = yourShirt;  
  
myShirt.colorCode = 'R';  
yourShirt.colorCode = 'G';  
  
System.out.println("Shirt color: " + myShirt.colorCode);
```

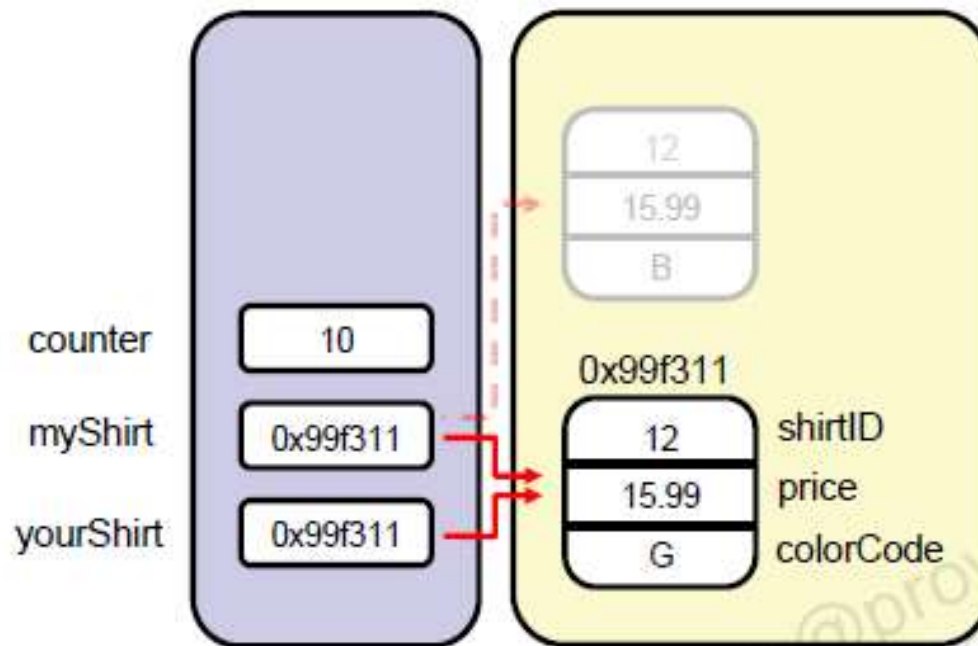
Salida del fragmento de código:

```
Shirt color: G
```



# Asignación de una referencia a otra

```
myShirt.colorCode = 'R';  
yourShirt.colorCode = 'G';
```



## Clase String

La clase String soporta alguna sintaxis no estándar.

- Se puede instanciar un objeto String sin utilizar la palabra clave `new`; se prefiere esto:

```
String hisName = "Fred Smith";
```

- Se puede utilizar la palabra clave `new`, pero *no* se recomienda:

```
String herName = new String("Anne Smith");
```

- Un objeto String es inmutable; su valor no se puede cambiar.
- Un objeto String se puede utilizar con el símbolo del operador de concatenación de cadenas (+) para la concatenación.

## Concatenación de cadenas

Cuando utiliza un literal de cadena en el código Java, se instancia y se convierte en una referencia String.

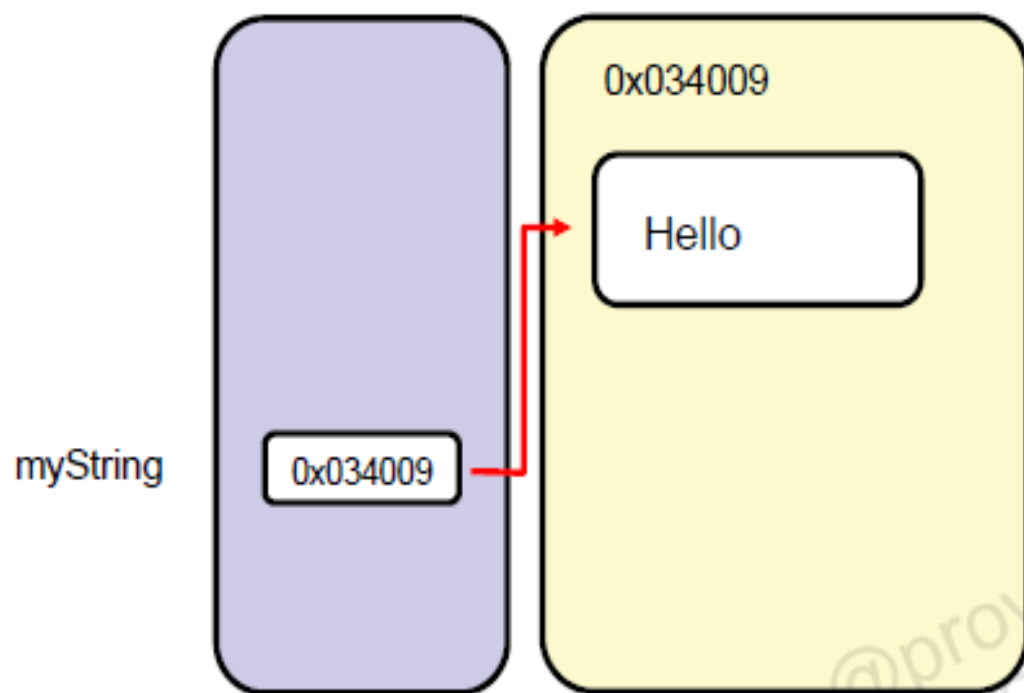
- Concatenar cadenas:

```
String name1 = "Fred"  
theirNames = name1 + " and " +  
               "Anne Smith";
```

- La concatenación crea una nueva cadena y la referencia String `theirNames` apunta ahora a esta nueva cadena.

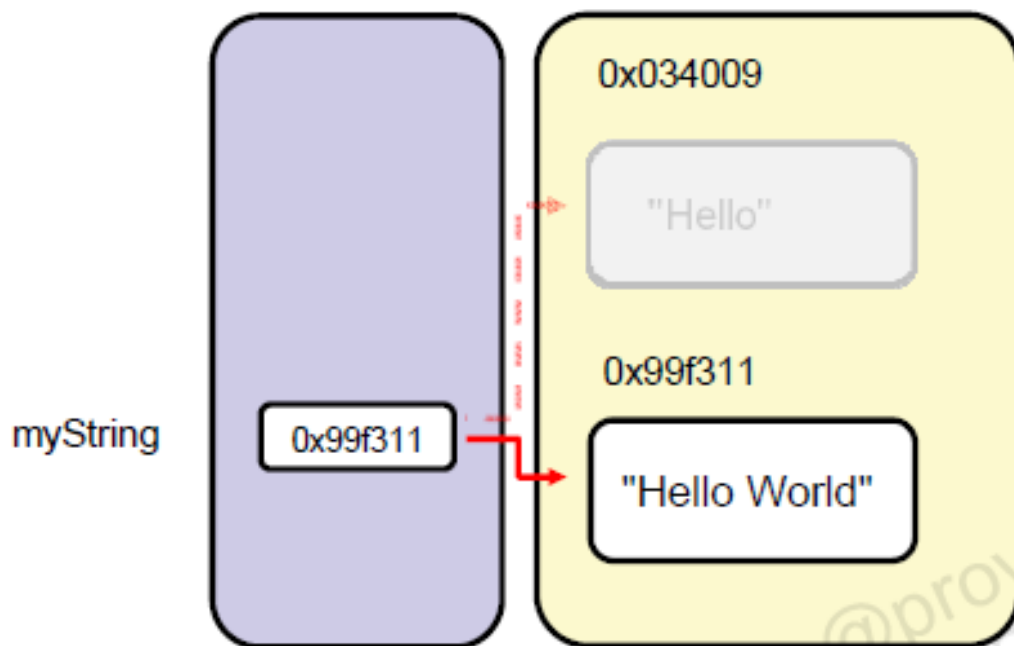
# Concatenación de cadenas

```
String myString = "Hello";
```



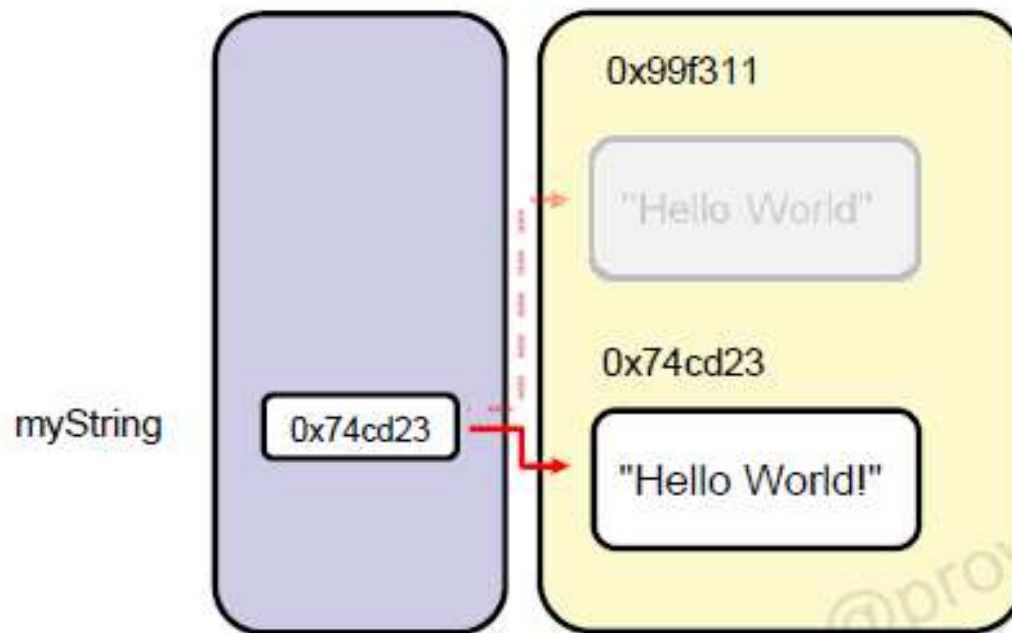
## Concatenación de cadenas

```
String myString = "Hello";  
myString = myString.concat(" World");
```



# Concatenación de cadenas

```
String myString = "Hello";  
myString = myString.concat(" World");  
myString = myString + "!"
```





## Llamadas al método String con valores de retorno primitivos

Una llamada a método puede devolver un único valor de cualquier tipo.

- Ejemplo de un método de tipo primitivo `int`:  
`String hello = "Hello World";`  
`int stringLength = hello.length();`

Adicion@proydesa.org) has a  
this Student Guide.

## Llamadas al método String con valores de retorno de objeto

Llamadas a método que devuelven objetos:

```
String greet = " HOW ".trim();
```

```
String lc = greet + "DY".toLowerCase();
```

O bien

```
String lc = (greet + "DY").toLowerCase();
```

Foundation@proydesa.org) has a  
this Student Guide.

# Llamadas a métodos que necesitan argumentos

Las llamadas a métodos pueden necesitar transferir uno o más argumentos:

- Transferir un primitivo

```
String theString = "Hello World";  
String partString = theString.substring(6);
```

- Transferir un objeto

```
boolean endWorld =  
    "Hello World".endsWith("World");
```

adacion@proydesa.org) has a  
this Student Guide.

# Documentación de la API de Java

Consta de un juego de páginas web.

- Muestra todas las clases de la API
  - Descripciones de la función de la clase
  - Lista de constructores, métodos y campos de la clase
- Gran cantidad de hiperenlaces para mostrar las interconexiones entre las clases y facilitar la búsqueda
- Disponible en el sitio web de Oracle en:  
<http://download.oracle.com/javase/7/docs/api/index.html>



# Documentación de la plataforma Java SE 7

Aquí puede seleccionar All Classes o un paquete concreto.

En este panel se muestran detalles sobre la clase seleccionada.

Según lo que seleccione, aquí se muestran las clases de un paquete concreto o todas las clases.

The screenshot displays the Java Platform Standard Ed. 7 documentation interface. On the left, a sidebar contains a tree view with 'All Classes' and 'Packages' sections. The 'Packages' section is expanded, showing a list of packages including 'java.applet', 'java.awt', and 'java.awt.color'. Below this, a list of classes is shown, with 'String' selected. The main content area on the right displays the details for the 'Class String' in the 'java.lang' package. It shows the inheritance hierarchy (java.lang.Object, java.lang.String), the implemented interfaces (Serializable, CharSequence, Comparable<String>), and the class declaration: 'public final class String extends Object implements Serializable, Comparable<String>, CharSequence'. A description at the bottom states: 'The String class represents character strings. All string literals in Java programs, implemented as instances of this class'.

Java™ Platform Standard Ed. 7

All Classes

Packages

- java.applet
- java.awt
- java.awt.color

String

StringWriter

StringBuffer

StringBufferInputStream

StringBuffer

StringCharacterIterator

StringContent

StringHolder

StringIndexOutOfBoundsException

StringMonitor

StringMonitor/Bean

StringNameHelper

StringReader

StringReader

Overview Package **Class** Use Tree Deprecated Index

Prev Class Next Class Frames: No Frames

Summary: Nested | Field | Constructor | Method Detail: Field | Constructor | Method

java.lang

## Class String

java.lang.Object

java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The String class represents character strings. All string literals in Java programs, implemented as instances of this class

# Documentación de la plataforma Java SE 7

Al desplazarse hacia abajo, se muestra más información de la clase String.

**Java™ Platform  
Standard Ed. 7**

All Classes

**Packages**

- java.applet
- java.awt
- java.awt.color

String

StringBuffer

StringBufferInputStream

StringBuilder

StringCharacterIterator

StringContent

StringHolder

StringIndexOutOfBoundsException

StringMonitor

StringMonitorMBean

StringNameHelper

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};  
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");  
String cde = "cde";  
System.out.println("abc" + cde);  
String c = "abc".substring(2,3);  
String d = cde.substring(1, 2);
```



# Plataforma Java SE 7: Resumen del método

Method Summary	
Methods	
Modifier and Type	Method and Description
<b>char</b>	<b>charAt(int index)</b> Returns the char value at the specified index.
int	<b>codePointAt(int index)</b> Returns the character (Unicode code point) at the specified index.
int	<b>codePointBefore(int index)</b> Returns the character (Unicode code point) before the specified index.
int	<b>codePointCount(int beginIndex, int endIndex)</b> Returns the number of code points in the specified text.
int	<b>compareTo(String str)</b> Compares two strings lexicographically.
int	<b>compareToIgnoreCase(String str)</b> Compares two strings lexicographically, ignoring case differences.
String	<b>concat(String str)</b> Concatenates the specified string to the end of this string.

Tipo del método (tipo que se devuelve).

Tipo del parámetro que se debe transferir al método.

Nombre del método.

# Plataforma Java SE 7: Detalles del método

Haga clic aquí para obtener la descripción detallada del método.

```
int indexOf(String str)
Returns the index within this string of the first occurrence of the
specified substring.

int indexOf(String str, int fromIndex)
Returns the index within this string of the first occurrence of the
specified substring, starting at the specified index.
```

Se muestran más detalles sobre los parámetros y el valor de retorno en la lista de método.

Descripción detallada del método `indexOf()`.

## indexOf

```
public int indexOf(String str)
```

Returns the index within this string of the first occurrence of the specified substring.

The returned index is the smallest value *k* for which:

```
this.startsWith(str, k)
```

If no such value of *k* exists, then -1 is returned.

### Parameters:

*str* - the substring to search for.

### Returns:

the index of the first occurrence of the specified substring, or -1 if there is no such occurrence.

## Métodos `System.out`

Para encontrar todos los detalles de `System.out.println()`, considere lo siguiente:

- `System` es una clase (en `java.lang`).
- `out` es un campo de `System`.
- `out` es un tipo de referencia que permite llamar a `println()` en el tipo de objeto al que hace referencia.

Para buscar la documentación:

1. Vaya a la clase `System` y busque el tipo del campo `out`.
2. Vaya a la documentación de dicho campo.
3. Revise los métodos disponibles.

# Documentación sobre `System.out.println()`

java.lang  
**Class System**  
java.lang.Object  
java.lang.System

public final class System  
extends Object

**Field Summary**

**Fields**

Modifier and Type	Field and Description
static <code>PrintStream</code>	<code>err</code> The "standard" error output stream.
static <code>InputStream</code>	<code>in</code> The "standard" input stream.
static <code>PrintStream</code>	<code>out</code> The "standard" output stream.

Algunos de los  
métodos de  
`PrintStream`

El campo `out` de `System` es  
de tipo `PrintStream`.

void `print(Object obj)`  
Prints an object.

void `print(String s)`  
Prints a string.

`PrintStream` `printf(Locale l, String format, Object... args)`  
A convenience method to write a formatted string to this output.

void `println(double x)`  
Prints a double and then terminate the line.

void `println(float x)`  
Prints a float and then terminate the line.

void `println(int x)`  
Prints an integer and then terminate the line.

void `println(long x)`  
Prints a long and then terminate the line.

void `println(Object x)`  
Prints an Object and then terminate the line.

void `println(String x)`  
Prints a String and then terminate the line.

## Uso de los métodos `print()` y `println()`

- Método `println`:  
`System.out.println(data_to_print);`
- Ejemplo:  
`System.out.print("Carpe diem ");`  
`System.out.println("Seize the day");`
- Este método muestra lo siguiente:  
Carpe diem Seize the day



## Clase StringBuilder

StringBuilder proporciona una alternativa variable a String.

StringBuilder:

- Es una clase normal. Utilice `new` para instanciarla.
- Tiene un amplio juego de métodos para agregar, insertar y suprimir.
- Tiene muchos métodos para devolver una referencia al objeto actual. No hay ningún costo de instanciación.
- Se puede crear con la capacidad inicial que mejor se adapte a las necesidades.

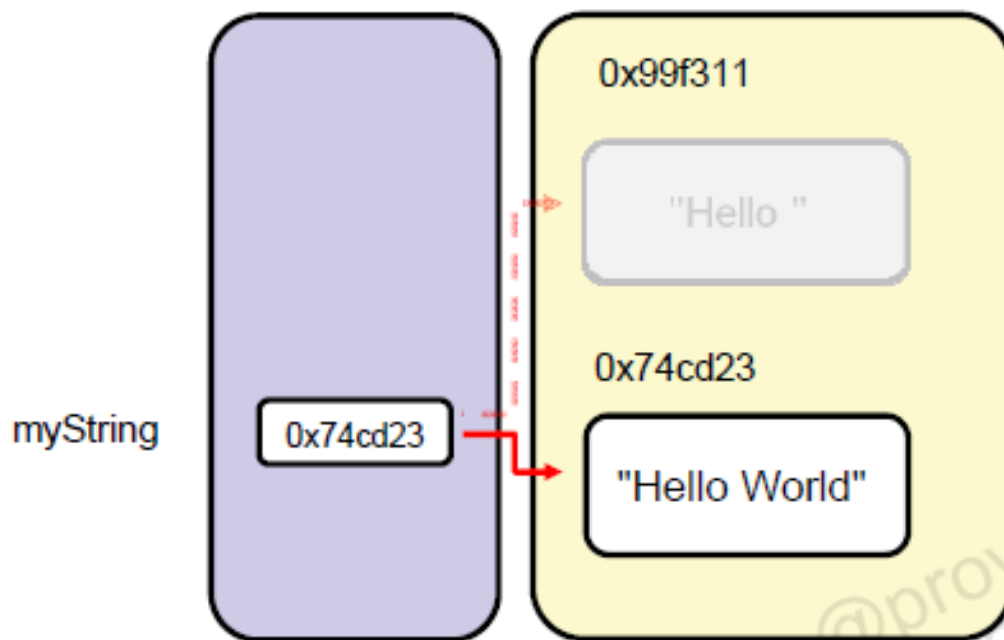
String sigue siendo necesaria porque:

- Su uso puede ser más seguro que un objeto inmutable.
- Una clase de la API puede necesitar una cadena.
- Tiene muchos más métodos no disponibles en StringBuilder.



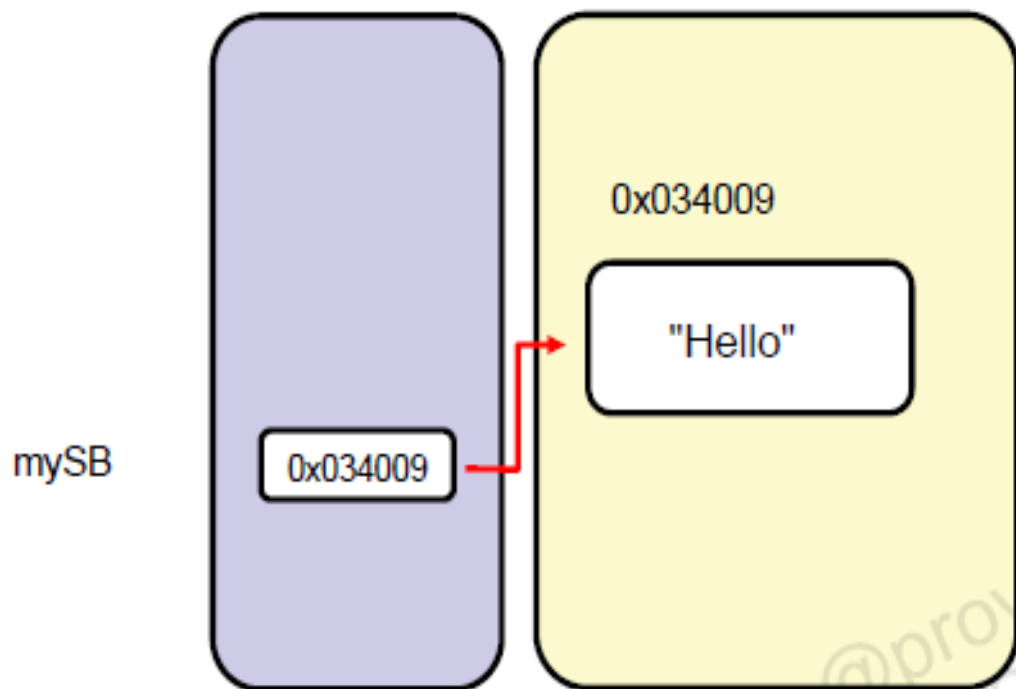
## Ventajas de StringBuilder sobre String para la concatenación (o adición)

```
String myString = "Hello";  
myString = myString.concat(" World");
```



# StringBuilder: Declaración e instanciación

```
StringBuilder mySB = new StringBuilder("Hello");
```



## Adición de StringBuilder

```
StringBuilder mySB = new StringBuilder("Hello");  
mySB.append(" World");
```

