

Programación Orientada a Objetos

```
var Carousel = {  
  init : function(options, el){  
    var base = this;  
  
    base.$elem = $(el);  
  
    // options passed via js override options passed via data attributes  
    base.options = $.extend({}, $.fn.owlCarousel.options, base.$elem.data(), options);  
  
    base.userOptions = options;  
    base.loadContent();  
  },  
  
  loadContent : function(){  
    var base = this;  
  
    if (typeof base.options.beforeInit === "function") {  
      base.options.beforeInit.apply(this, [base.$elem]);  
    }  
  
    if (typeof base.options.jsonPath === "string") {  
      var url = base.options.jsonPath;  
  
      function getData(data) {  
        if (base.options.jsonSuccess === "function") {  
          base.options.jsonSuccess.apply(this, [data]);  
        }  
      }  
    }  
  }  
};
```



Tener en cuenta...

- Laboratorio 1

Sábado 27 de Agosto

- ✓ Parejas
- ✓ Revisión a par con el monitor.
- ✓ Los horarios se seleccionan en Moodle
- ✓ ¿BlueJ?

Lectura S4



LECTURA

BARKER, JACQUIE "BEGINNING JAVA OBJECTS: FROM CONCEPTS TO CODE". APRESS. 2005.
SEGUNDA EDICIÓN.

4. Object Interaction



Esquina

CODE CODE THE UNIT TEST FIRST. G01. G02.
TESTING ALL CODE MUST HAVE UNIT TESTS. G01. G02.

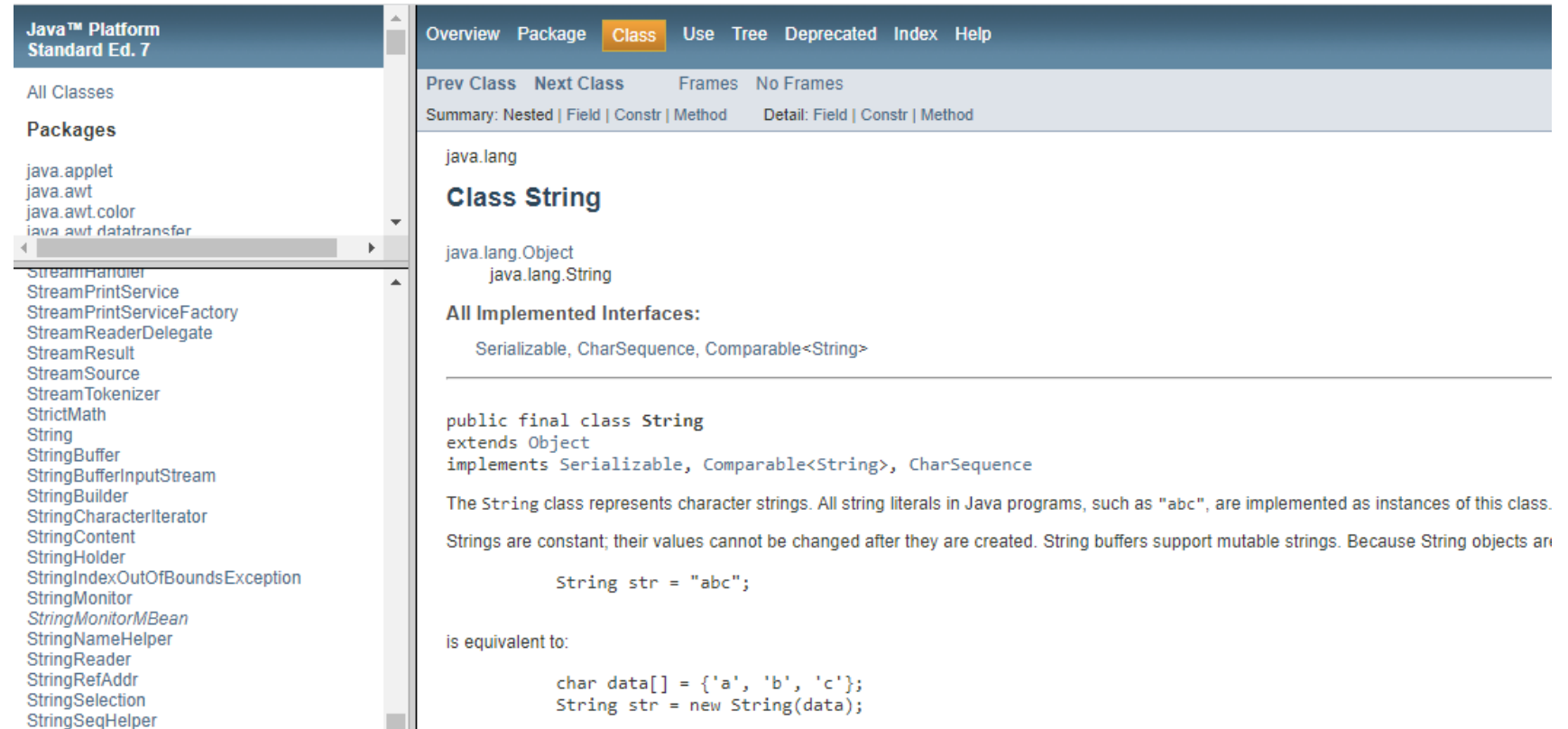
== Vs. Equals()

```
public class Test {  
    public static void main(String[] args)  
    {  
        String s1 = new String("HELLO");  
        String s2 = new String("HELLO");  
        System.out.println(s1 == s2);  
        System.out.println(s1.equals(s2));  
    }  
}
```

Output ?

API *Interfaz de programación de aplicaciones*

Biblioteca de funciones, procedimientos Y/o protocolos que permite la interacción entre aplicaciones



The screenshot displays the Java Platform Standard Ed. 7 API documentation for the `String` class. The left sidebar shows the package hierarchy, with `java.lang` selected. The main content area shows the `String` class, which extends `Object` and implements `Serializable`, `Comparable<String>`, and `CharSequence`. The class is described as representing character strings, and it is noted that string literals in Java programs are implemented as instances of this class. The documentation also shows the class is constant and that string buffers support mutable strings. The code snippet shows the declaration of a `String` variable and its initialization with a character array.

Java™ Platform Standard Ed. 7

All Classes

Packages

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer
- StreamHandler
- StreamPrintService
- StreamPrintServiceFactory
- StreamReaderDelegate
- StreamResult
- StreamSource
- StreamTokenizer
- StrictMath
- String
- StringBuffer
- StringBufferInputStream
- StringBuilder
- StringCharacterIterator
- StringContent
- StringHolder
- StringIndexOutOfBoundsException
- StringMonitor
- StringMonitorMBean
- StringNameHelper
- StringReader
- StringRefAddr
- StringSelection
- StringSeqHelper

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | 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, 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

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

is equivalent to:

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

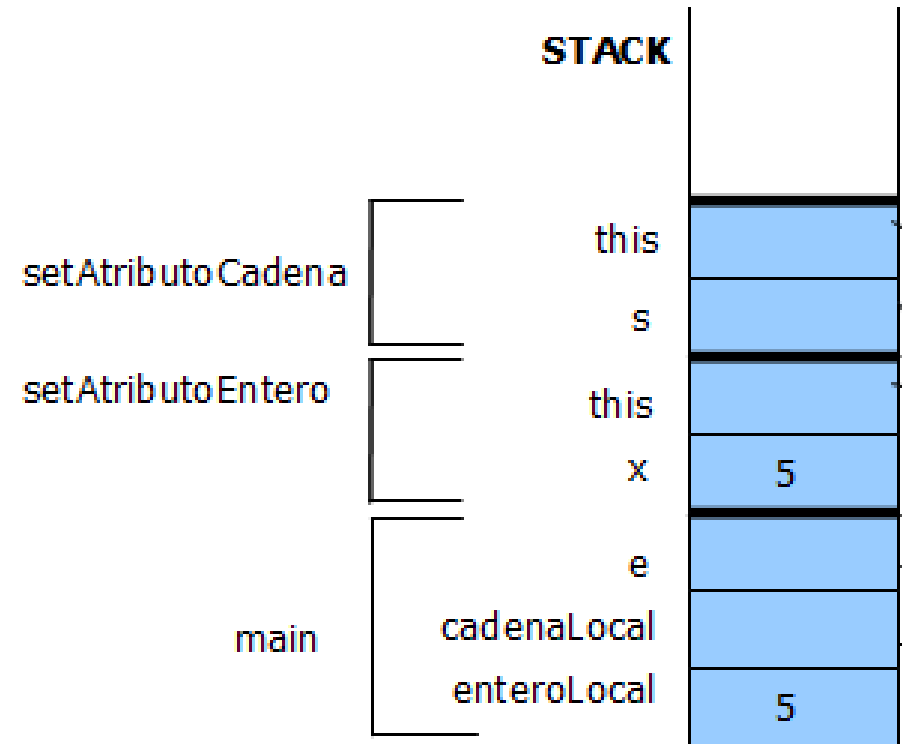
MEMORIA

STACK [*PILA*]

Pila de invocaciones

Cada vez que se invoca una función, un bloque de memoria se reserva en la parte superior de la pila para variables locales

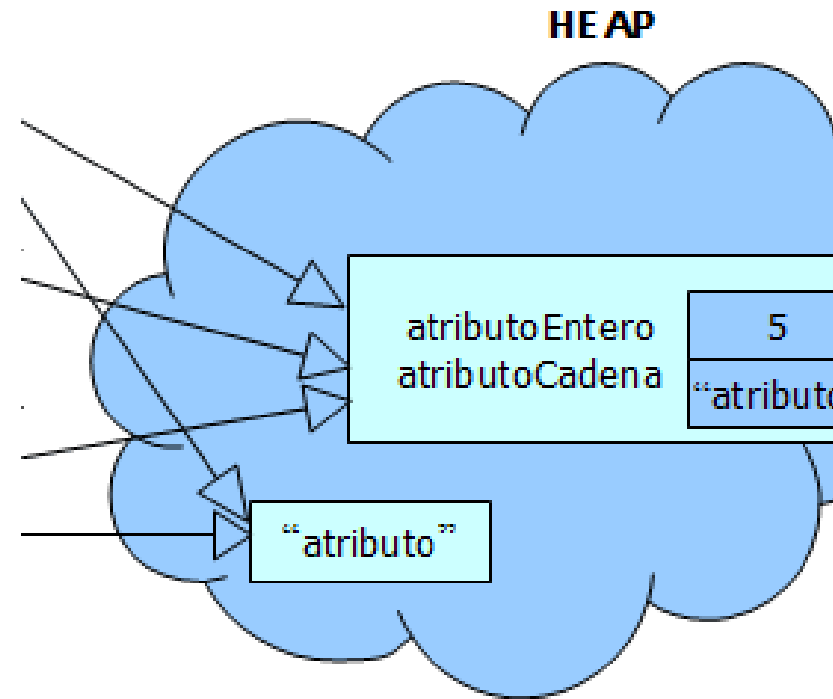
LIFO (last in first out)



MEMORIA

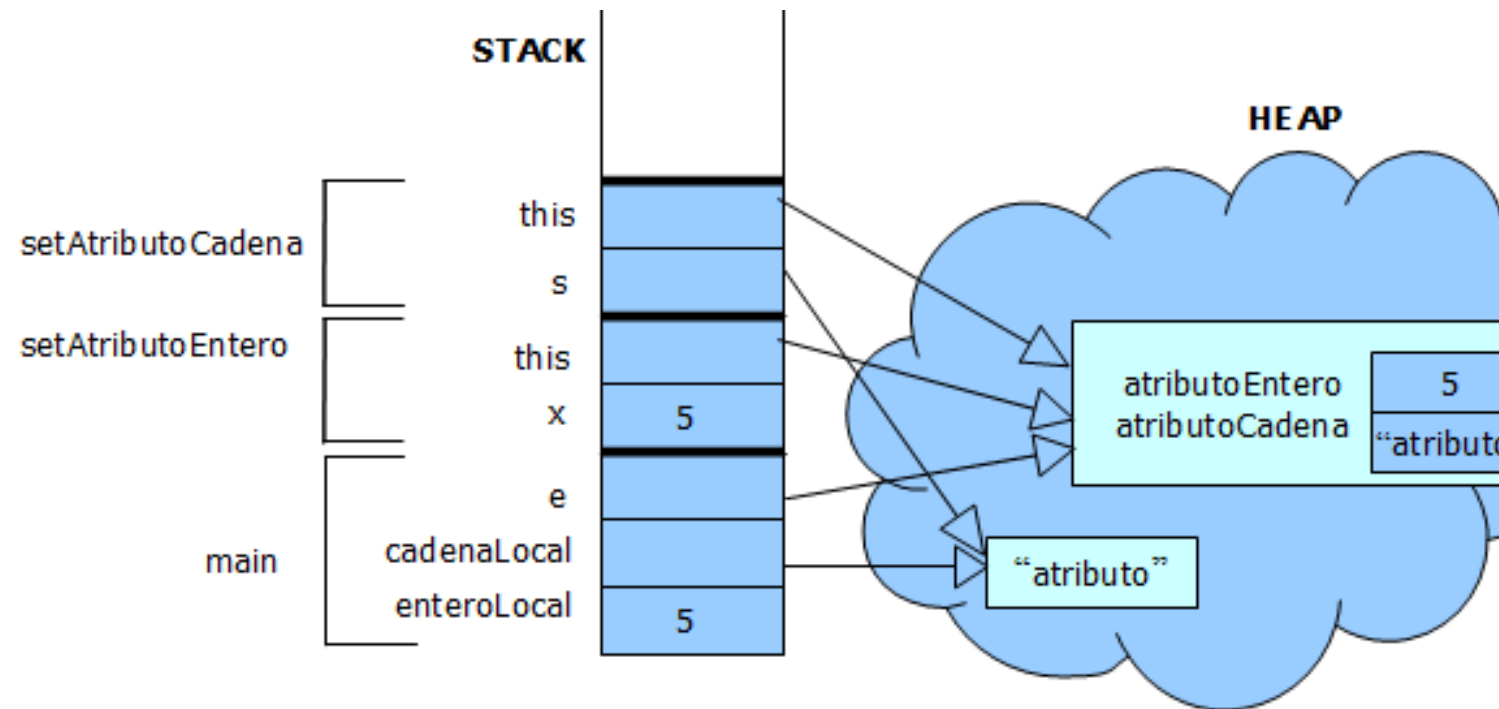
HEAP [*CÚMULO - CONJUNTO*]

Memoria para la asignación dinámica. No tiene un orden o ubicación en específico



MEMORIA

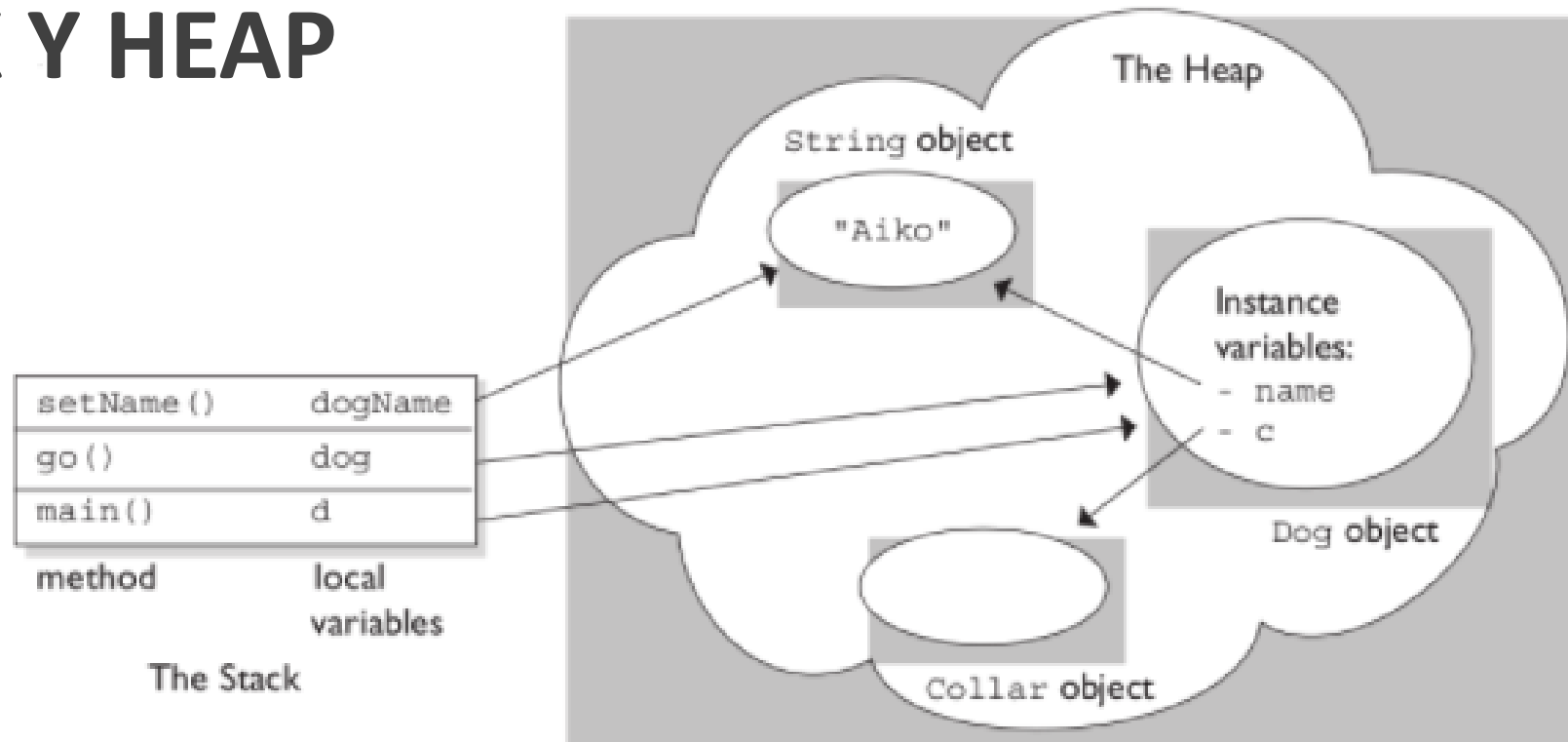
STACK Y HEAP



Tomado de: [http://www.javahispano.org/certificacion/2011/9/27/stack-y-heap.html#:~:text=El%20Stack%20\(Pila\)%20se%20utiliza,y%20retorno%20de%20los%20m%C3%A9todos.&text=El%20Heap%20\(Mont%C3%ADculo\)%20almacena%20objetos%20y%20sus%20variables%20de%20instancia](http://www.javahispano.org/certificacion/2011/9/27/stack-y-heap.html#:~:text=El%20Stack%20(Pila)%20se%20utiliza,y%20retorno%20de%20los%20m%C3%A9todos.&text=El%20Heap%20(Mont%C3%ADculo)%20almacena%20objetos%20y%20sus%20variables%20de%20instancia).

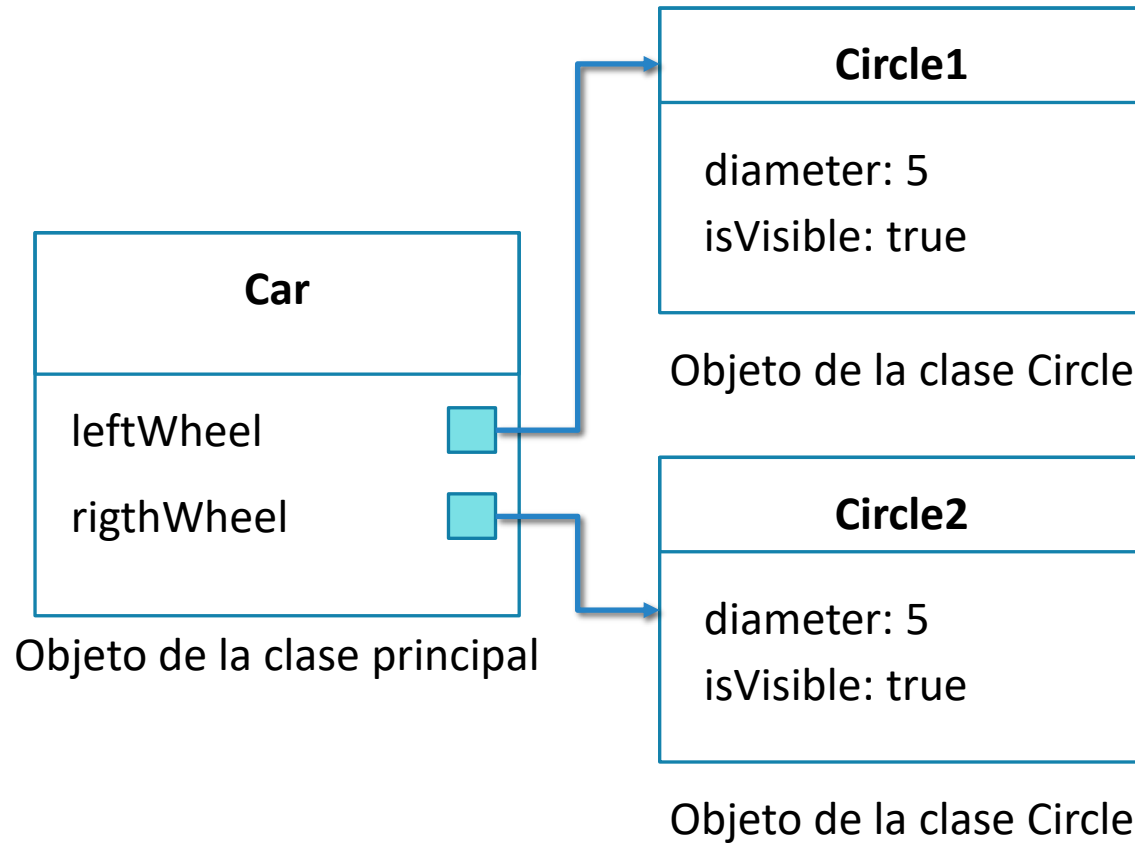
MEMORIA

STACK Y HEAP



Tomado de: <https://vikashazrati.wordpress.com/2007/10/01/quicktip-java-basics-stack-and-heap/>

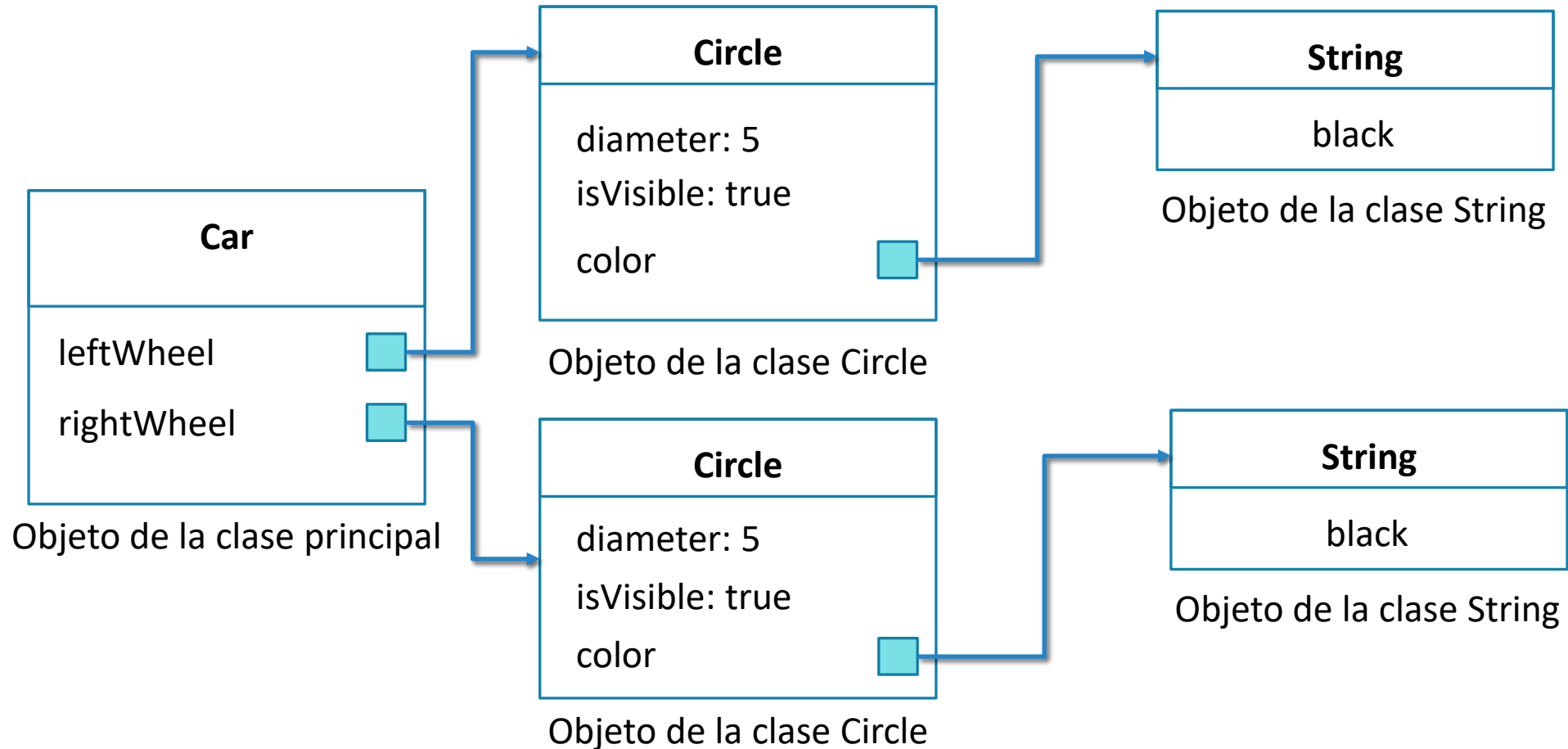
Mapa de memoria



```
public class Circle{  
    public static final double PI=3.1416;  
  
    private int diameter;  
    private int xPosition;  
    private int yPosition;  
    private String color;  
    private boolean isVisible;
```

¿color?

Mapa de memoria



Mapa de memoria

¿Si es una colección?

