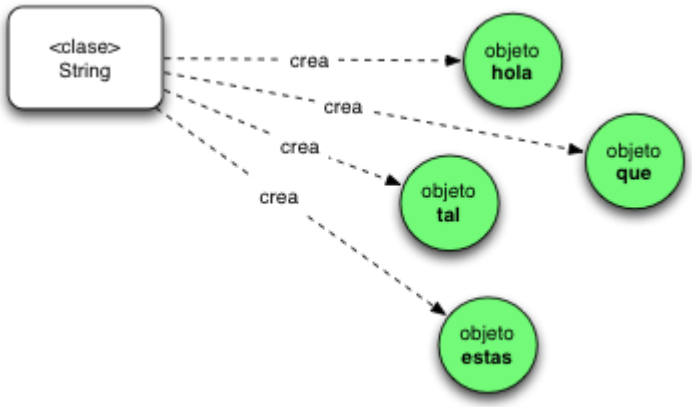
Hay pocas clases con las que trabajemos mas a menudo que la clase String . Sin embargo mucha gente no conoce como funciona esta clase a detalle lo cual puede llevarnos a malentendidos .Vamos a ver un código muy sencillo.

```
package com.arquitecturajava;

public class Principal {
 public static void main(String[] args) {

 String cadena="";
 long numero1= System.currentTimeMillis();
 for(int i=0;i<10000;i++) {

 cadena+= "hola" + "que" +"tal" +"estas";
 }
 long numero2=System.currentTimeMillis();
 System.out.println(numero2-numero1);
 }

}
```

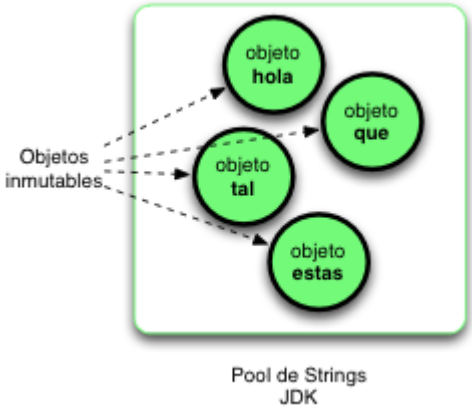
String y Bubles

En este bloque de código ejecutamos un sencillo buble for que concatena cadenas y calculamos el tiempo que tarda en hacerlo el procesador. Sorpresivamente el procesador tarda casi un segundo en ejecutar este código tan sencillo. En principio unicamente estamos construyendo objetos.



String Inmutable

Sin embargo los objetos de tipo String tienen una peculiaridad son “Inmutables” esto quiere decir que una vez hemos construido el objeto, este no se puede modificar .En el caso de las cadenas son almacenadas e un pool de Strings para su posterior uso.



Si revisamos el código de dentro del bucle for podremos nos parecerá relativamente sencillo .

```
for(int i=0;i<10000;i++) {
```

```
cadena+= "hola" + "que" +"tal" +"estas";
 }
```

En la linea se construyen 4 objetos y se concatenan entre ellos .Sin embargo si lo revisamos a detalle partiendo de que los Strings son objetos “Inmutables” nos daremos cuenta que se crean los siguientes objetos en memoria.

1. “hola”
2. “que”
3. “tal”
4. “estas”
5. “tal estas”
6. “que tal estas”
7. “hola que tal estas”

Tenemos un pequeño problema ya que ya no son unicamente 4 objetos sino que son 7 .

Si realizamos el buble 10000 veces tendremos 70000 objetos en memoria . Ahora bien si miramos un poco mas a detalle nos daremos cuenta que todos estos objetos solo se crean una vez y luego se almacenan en un pool de “String” .

Así pues parece que al final solo tenemos 7 objetos en todo el buble . Sin embargo ¿como es que va tan lento? . Bueno realmente tenemos unicamente 7 objetos en la primera iteración del buble. Si pasamos a la segunda iteración nos encontraremos algo así.

StringBuffer

Así que tenemos un objeto adicional mas que se genera. No parece mucho pero en un buble de 10000 son 10000 objetos nuevos (ya que todos son distintos) ademas de copiar el texto del objeto anterior al nuevo . Ahora si que podemos ver el problema de una forma bastante mas clara. Para solucionarlo en vez de usar en este buble un objeto de tipo String usaremos un StringBuffer.

```
package com.arquitecturajava;

public class Principal02 {

public static void main(String[] args) {
```

```
 StringBuffer cadena=new StringBuffer();
 long numero1= System.currentTimeMillis();
 for(int i=0;i<10000;i++) {

 cadena.append("hola");
cadena.append("que");
cadena.append("tal");
cadena.append("estas");
}
 long numero2=System.currentTimeMillis();
 System.out.println(numero2-numero1);
 }

}
```

La diferencia fundamental es que el objeto StringBuffer se crea la primera vez y va acumulando el texto.Por lo tanto nos ahorraremos el crear 10000 objetos adicionales. Realizada la modificación nos encontraremos con que el rendimiento se dispara. En la mayoría de las ocasiones String funciona mejor que StringBuffer .Es en estas situaciones peculiares en las que el rendimiento es peor . Mas adelante abordaremos la clase StringBuilder que es complementaria a las dos anteriores.