FLYWEIGHT PATTERN


```
public class Jugador {

  String nombre;
  int numero;
  byte[] camiseta;
}
```

Problema: puedo tener muchos jugadores con camisetas casi iguales

“

Use sharing to support large numbers of fine-grained objects efficiently.

— *GoF*

”

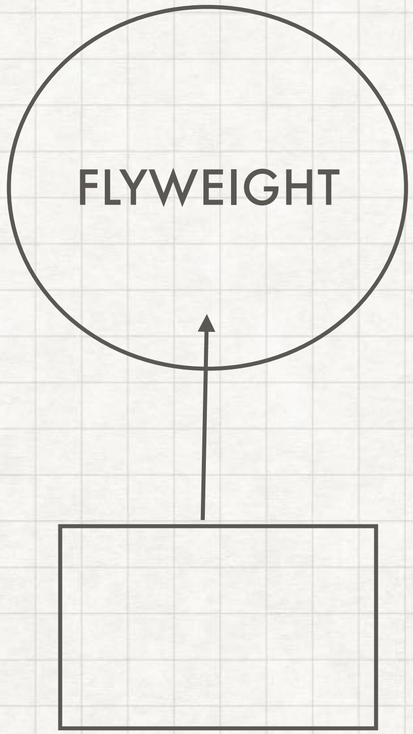
Objeto

Estado intrínseco

No es propio del objeto específico:
puede ser compartido entre distintas instancias

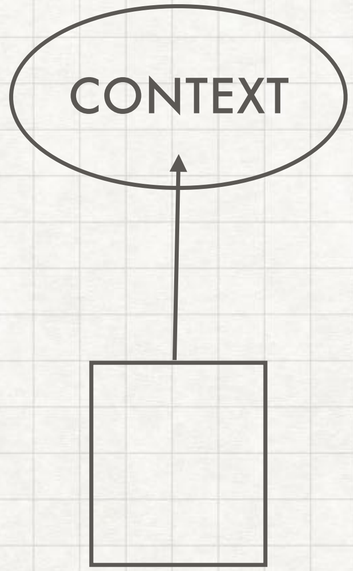
Estado extrínseco

Particular de cada instancia:
NO puede ser compartido con las otras



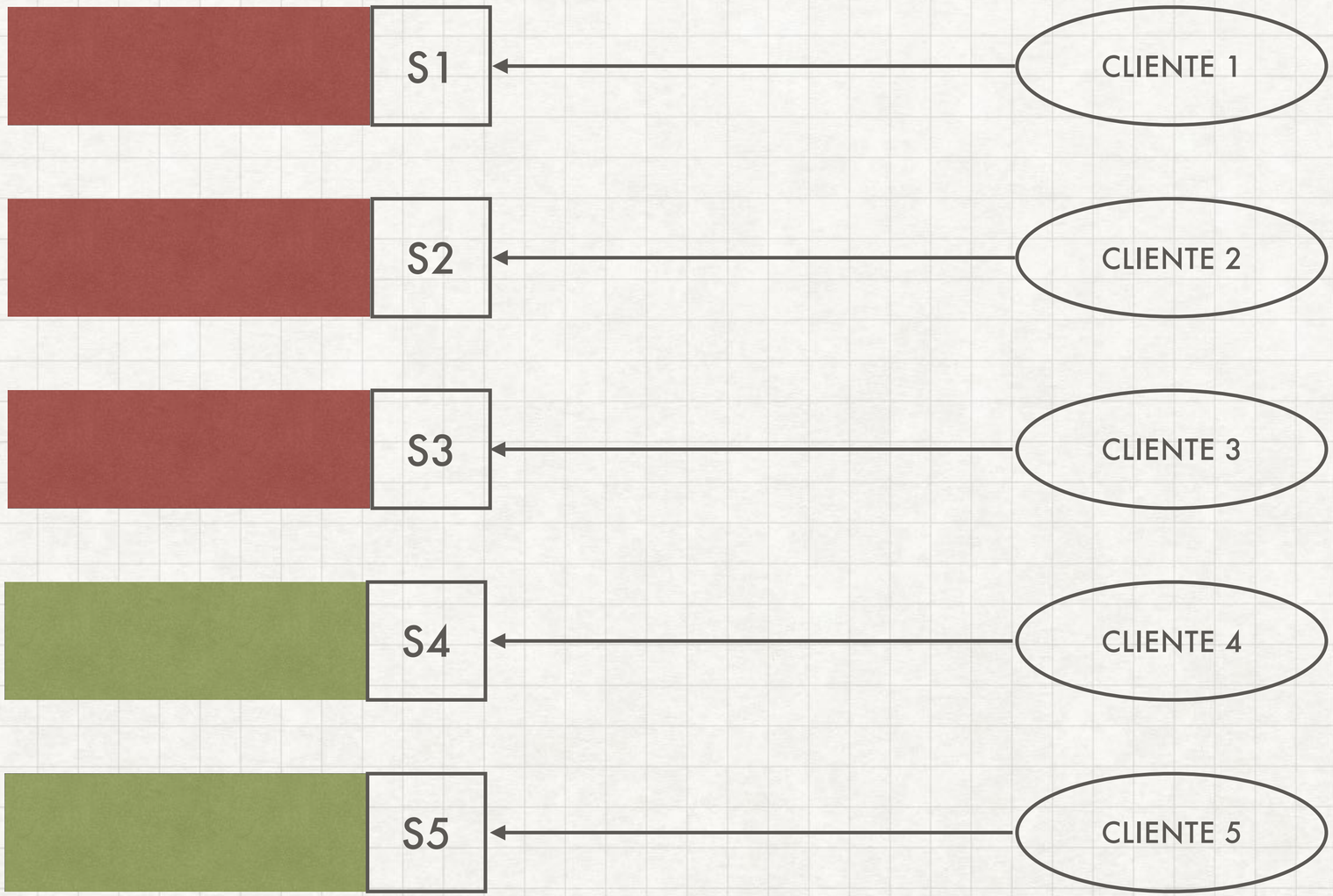
Estado intrínseco

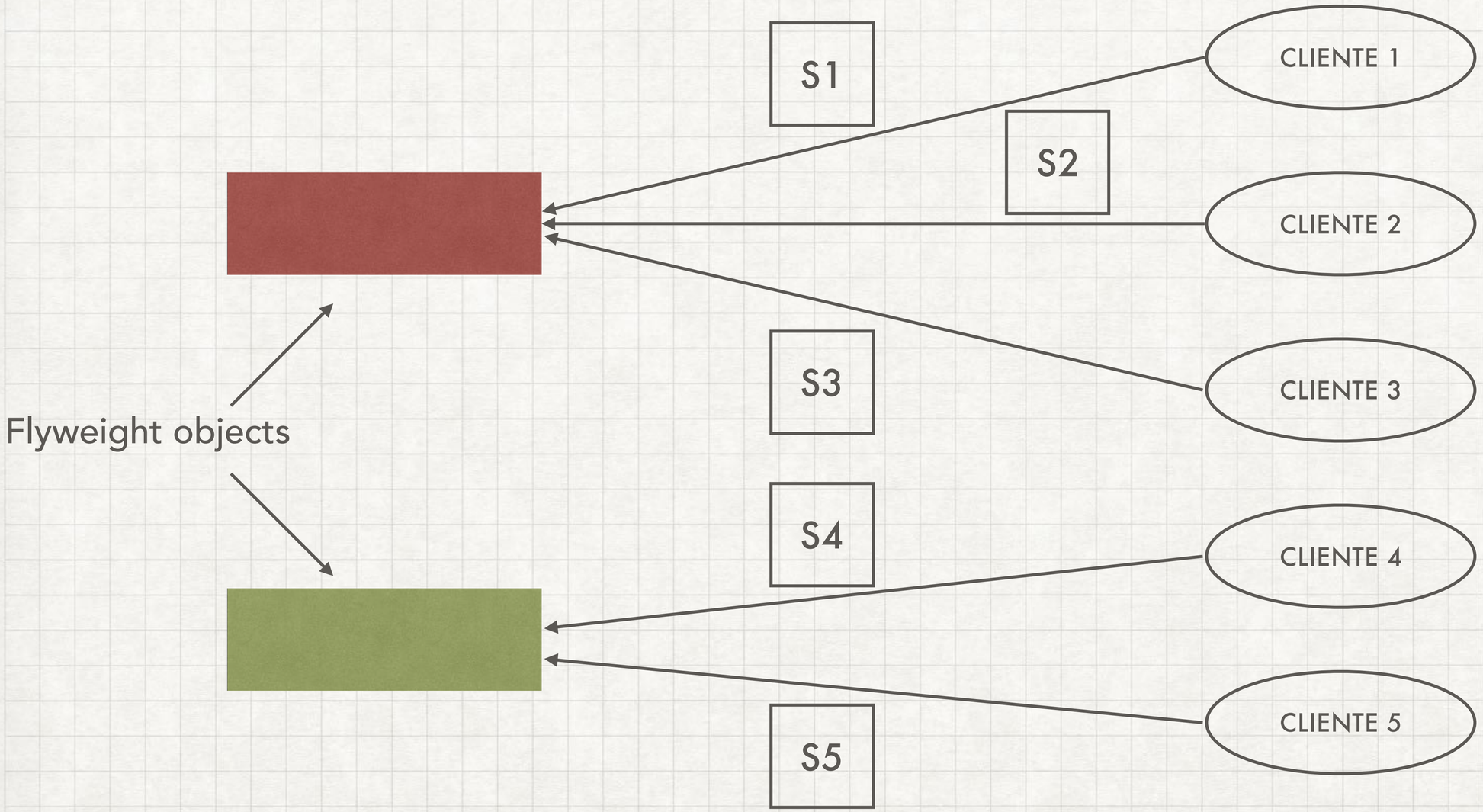
No es propio del objeto específico:
puede ser compartido entre distintas instancias

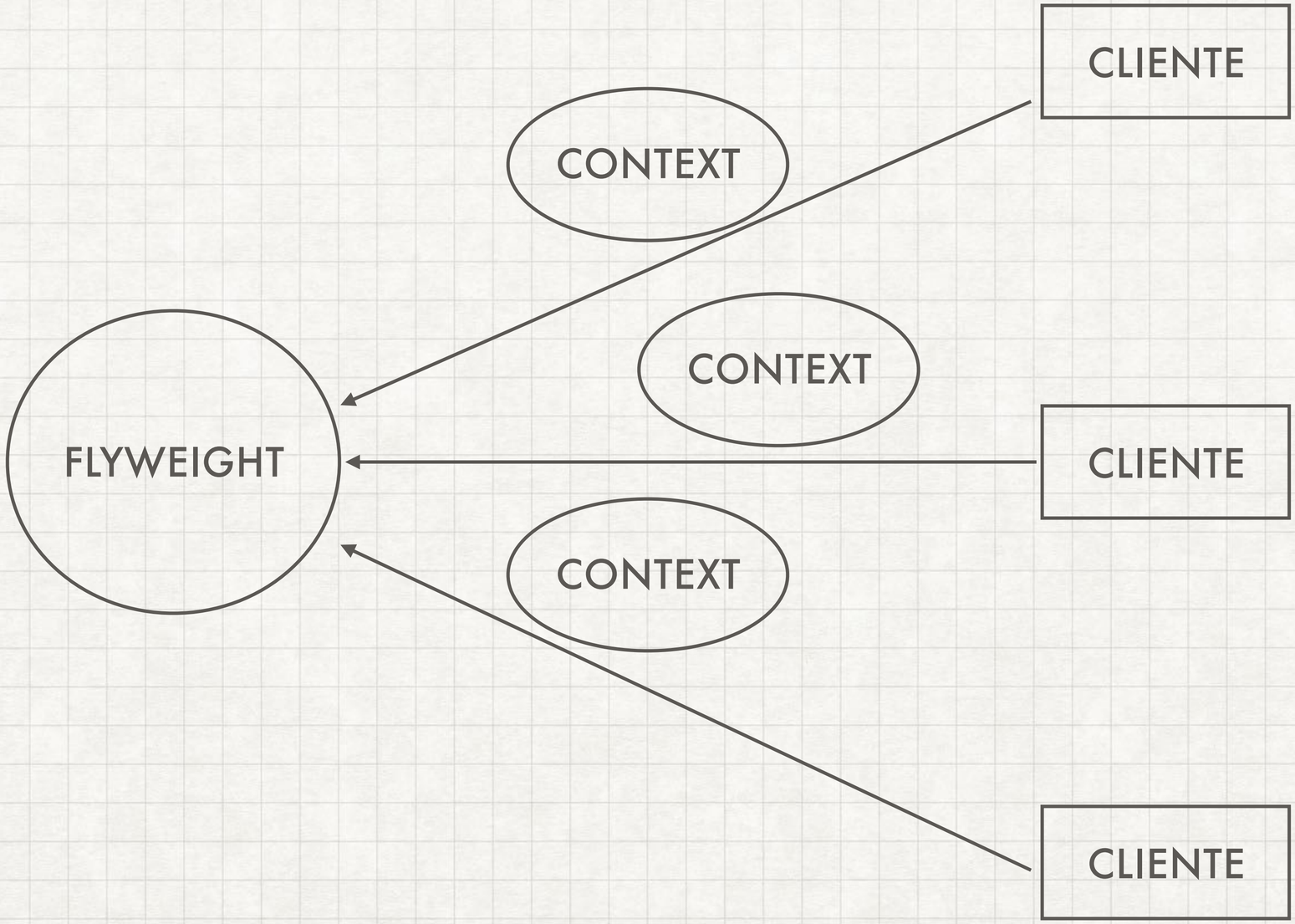


Estado extrínseco

Particular de cada instancia:
NO puede ser compartido con las otras


S1
S2
S3
S4
S5
CLIENTE 1
CLIENTE 2
CLIENTE 3
CLIENTE 4
CLIENTE 5


S1
S2
S3
S4
S5
CLIENTE 1
CLIENTE 2
CLIENTE 3
CLIENTE 4
CLIENTE 5
Flyweight objects


CLIENTE
CONTEXT
CONTEXT
FLYWEIGHT
CLIENTE
CONTEXT
CLIENTE

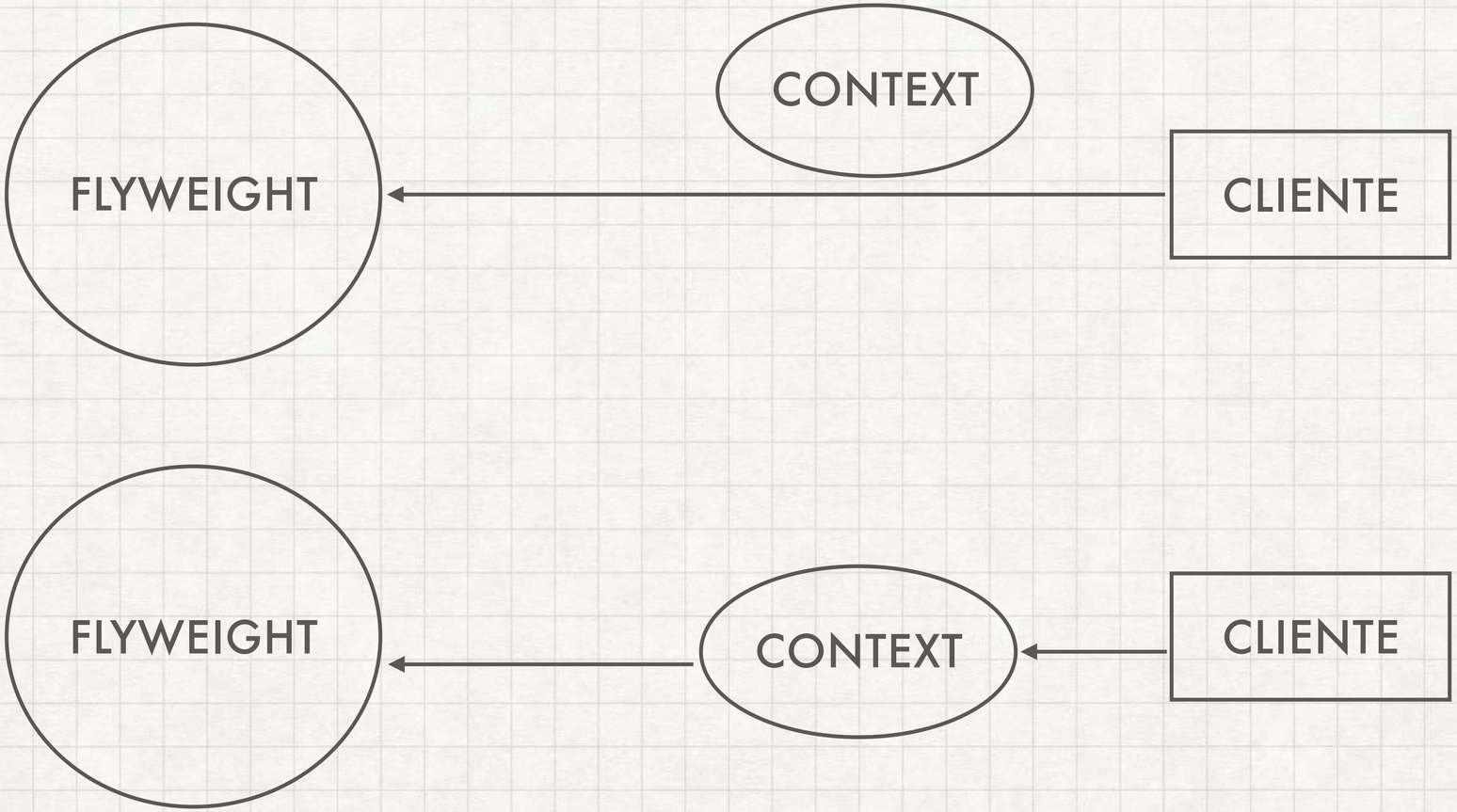
Problema: creación de los objetos

```
Flyweight f1 = new Flyweight();
Flyweight f2 = new Flyweight();
```

El cliente no puede instanciar directamente los objetos:

¡No habría reutilización!

Necesitamos encapsularlo en un factory con la lógica de cuando es necesario crear un flyweight nuevo y cuando se puede reusar uno.


FLYWEIGHT
CONTEXT
CLIENTE
FLYWEIGHT
CONTEXT
CLIENTE

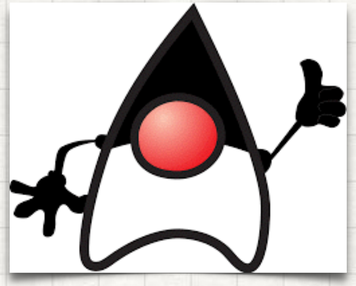

```
public class Jugador {

  String nombre;
  int numero;
  byte[] camiseta;
}
```

Problema: puedo tener muchos jugadores con camisetas casi iguales


```
public class Equipo {

  byte[] camiseta;

  byte[] getCamisetaJugador (int numero, String nombre) {
     // combina la imagen con el numero y nombre
  }

}
```

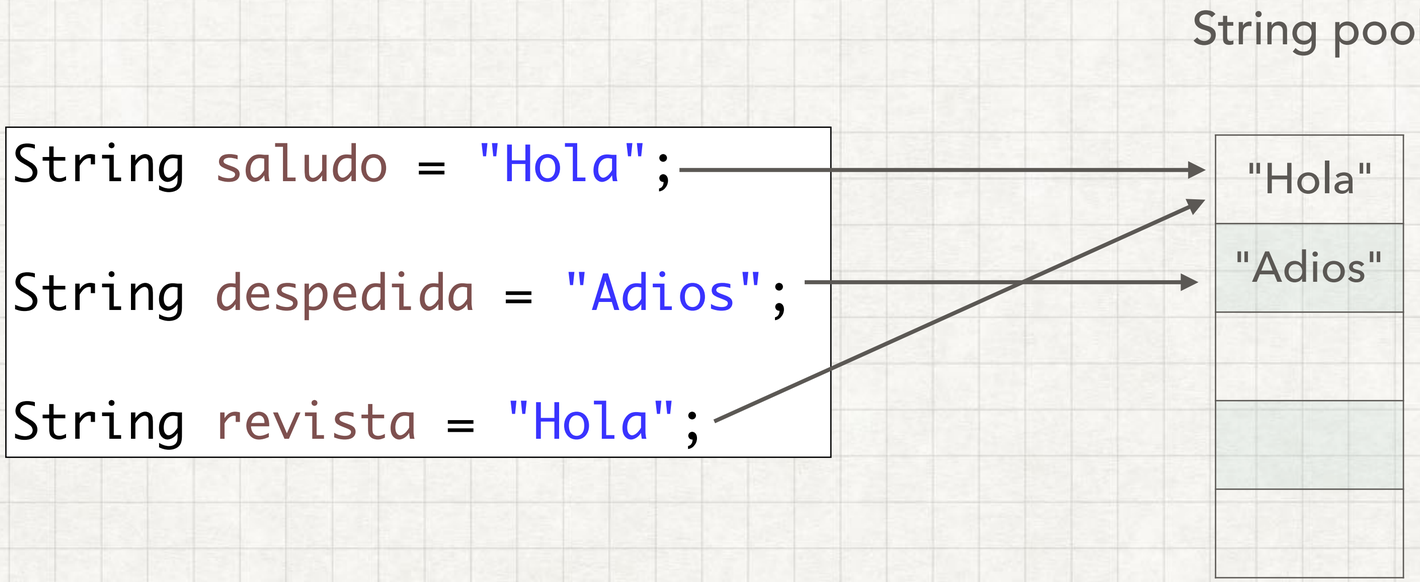
Todos los miembros de un equipo comparten el byte[]


```
public class Jugador {

  String nombre;
  int numero;

  Equipo equipo;

  byte[] getCamiseta() {
     return equipo.getCamisetaJugador(numero, nombre);
  }
}
```

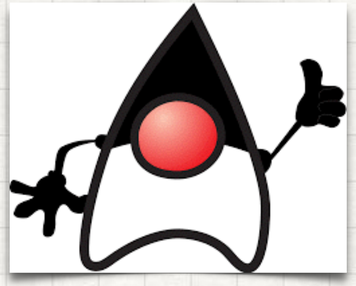


String interning

"Técnica que consiste en guardar una sola copia por cada cadena distinta"



¿Que requiere esta técnica para funcionar correctamente?



String interning : consecuencias

- Los String han de ser *immutables*
- Los métodos de modificación (concat, replace ...) devuelven un nuevo String
- No se deben crear con el operador new
- Podemos usar == en lugar de equals (¡si no hay new String()!)
- APIs de seguridad usan char[] en lugar de String

FLYWEIGHT PATTERN