Manipular cadenas con String

Fundamentos String

- Un objeto de la clase String en una cadena de caracteres inmutable, no se pueden modificar
- Los métodos que operan con String devuelven una copia de la cadena modificada
- Se pueden crear:

```
String n1=new String("mi cadena");
```

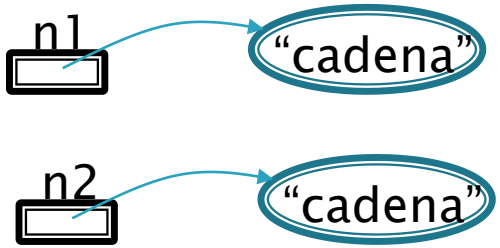
- O de forma simplificada:

```
String n1="mi cadena";
```

Operador == con objetos

- Se utiliza para comprobar la igualdad con tipos primitivos
- En variables de tipo objeto compara referencias, no los objetos:

```
String n1=new String("cadena");
String n2=new String("cadena");
//el resultado es falso
if(n1==n2){

}
```



- Al apuntar a objetos diferentes, las referencias son diferentes

Igualdad de cadenas

- Para comparar dos cadenas de caracteres utilizamos el método equals():

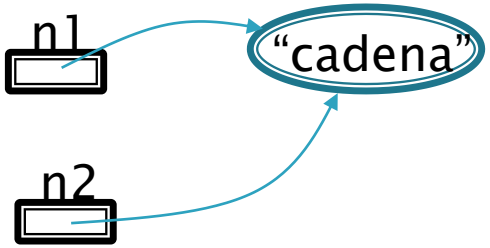
```
String n1=new String("cadena");
String n2=new String("cadena");
//el resultado es verdadero
if(n1.equals(n2)){

}
```

- El método equals() distingue mayúsculas y minúsculas, para ignorar la diferencia, utilizamos equalsIgnoreCase()

Pool de cadenas

- Java utiliza un pool de literales de cadenas de caracteres para optimizar memoria:

```
String n1="cadena";
String n2="cadena";
//el resultado es verdadero
//apuntan al mismo objeto
if(n1==n2){
}
```



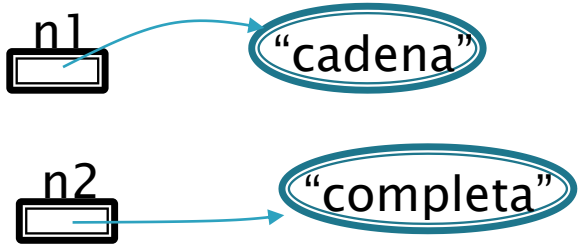
- Al asignar un literal de cadena, no se crea un nuevo objeto, se comprueba si existe en el pool y si es así se devuelve una referencia al objeto existente. Si no existe, se crea y se graba en el pool.

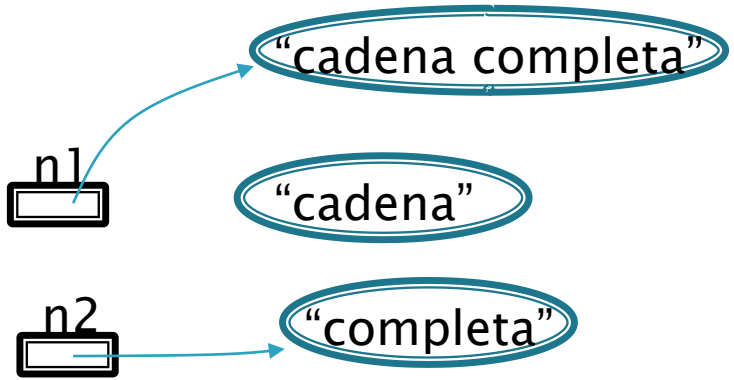
Inmutabilidad de objetos String

- Un objeto String representa una cadena de caracteres inmutable, es decir, no se puede modificar.

- En la concatenación, no se modifica ningún objeto existente, se crea uno nuevo:

```
String n1="cadena";
String n2=" completa";
n1=n1+n2;
```





Métodos String (I)

- int length(). Devuelve la longitud de la cadena

- String toLowerCase(), toUpperCase(). Devuelven la cadena convertida a minúsculas y mayúsculas, respectivamente

```
String n1="cadena";
System.out.println(n1.toUpperCase()); //muestra: CADENA
System.out.println(n1); //muestra: cadena, no ha cambiado
```

- String substring(int a, int b). Devuelve un trozo de cadena comprendido entre las posiciones a y b-1

```
String n1="esto es un texto";
System.out.println(n1.substring(3,9)); //muestra: o es u
```

Métodos String (II)

- **char charAt(int pos). Devuelve el carácter que ocupa la posición indicada**

```
String n1="esto es un texto";
System.out.println(n1.charAt(0)); //muestra: e
System.out.println(n1.charAt(20));  //StringIndexOutOfBoundsException
```

- **int indexOf(String cad). Devuelve la posición de la cadena parámetro. Si no existe, devuelve -1**

```
String n1="esto es un texto";
System.out.println(n1.indexOf("un")); //muestra: 8
```

- **String replace(CharSequence c1, CharSequence c2). Devuelve la cadena resultante de reemplazar la subcadena c1 por c2.**

```
String n1="esto es un texto";
System.out.println(n1.replace("es","de")); //muestra: deto de un texto
```

Métodos String (III)

- boolean startsWith(String s), endsWith(String s). Indica si la cadena empieza o termina, respectivamente, por el texto recibido:

```
String n1=“esto es un texto”;
System.out.println(n1.endsWith(“to”)); //muestra: true
System.out.println(n1.startsWith(“eso”)); //muestra: false
```

- String trim(). Devuelve la cadena resultante de eliminar espacios al principio y al final de la misma

```
String n1=“  cade prueba nueva  ”;
System.out.println(n1.trim().length()); //muestra: 17
```

- String concat(String s). Mismo efecto que aplicar el operador +

- boolean isEmpty(). Devuelve *true* si es una cadena vacía. Equivale:

```
cad.equals(“”)
```