



MANEJO DE CADENAS



2

Objetivo(s)

- Aprender sobre las cadenas constantes
- Aprender los constructores de cadenas
- Aprender los métodos comunes
- Entender la inmutabilidad de las cadenas
- Aprender a convertir números en cadenas

Clase String

3

- Un objeto de tipo cadena representa una secuencia de caracteres.
- La clase String pertenece al paquete `java.lang`, que no requiere importación.
- Como cualquier clase, String tiene métodos y constructores.
- A diferencia de otras clases, String tiene operadores `+` y `+=` que se utilizan para concatenación.

Cadenas literales

4

- Son objetos anónimos de la clase String
- Se definen por medio de dobles comillas, ejemplo:
"Esto es una cadena constante"
- Pueden ser asignadas a variables del tipo cadena.
- Pueden enviarse a métodos y constructores como parámetros.
- Tienen métodos que pueden ser invocados.

Inmutabilidad

6

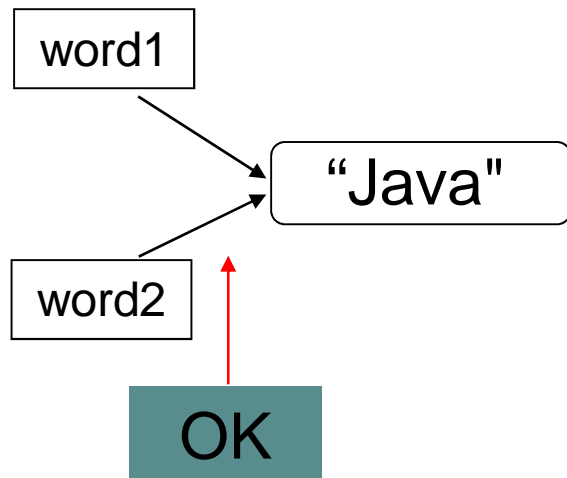
- Una vez creada, una cadena no puede ser modificada. Ninguno de sus métodos la modifica.
- Un objeto con esta propiedad es llamado "immutable"
- Esta característica es conveniente cuando se tienen múltiples referencias al objeto.

Ventajas de la inmutabilidad

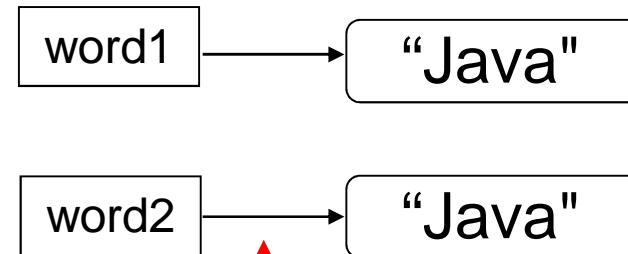
7

Usa menos memoria.

```
String word1 = "Java";  
String word2 = word1;
```



```
String word1 = "Java";  
String word2 = new String(word1);
```

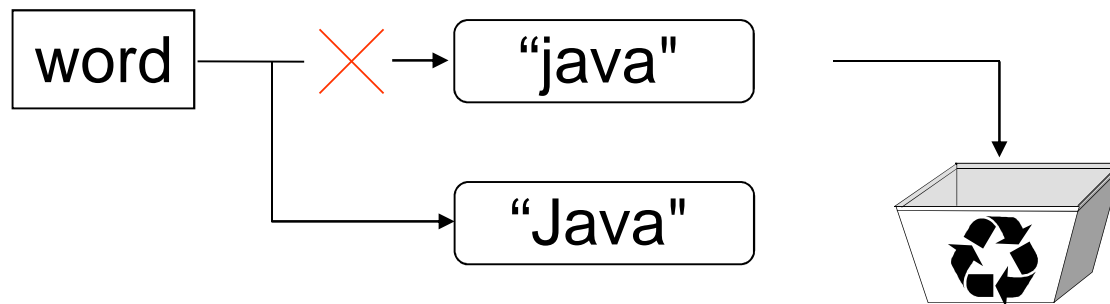







Desventajas de la inmutabilidad

8

- Se necesitan crear nuevas cadenas y tirar las anteriores incluso por cambios pequeños.

```
String word = "java";  
char ch = Character.toUpperCase(word.charAt (0));  
word = ch + word.substring (1);
```




```
public class EjemploCambios {  
    public static void main (String [] args){  
        String word = "java";  
        System.out.println (word);  java  
  
        char inicial = word.charAt (0);  
        System.out.println (inicial);  j  
  
        char mayuscula = Character.toUpperCase(inicial);  
        System.out.println (mayuscula);  J  
  
        String subcadena = word.substring (1);  
        System.out.println (subcadena);  ava  
  
        word = mayuscula + word.substring (1);  
        System.out.println (word);  Java  
    }  
}
```

Cadenas vacías

10

- Una cadena vacía no tiene caracteres. Su longitud es 0.

```
String word1 = "";  
String word2 = new String();
```

Cadenas vacías



- No es lo mismo que una cadena no inicializada.

```
private String errorMsg;
```

**errorMsg
es null**



Constructor sin argumento

11

- El constructor sin argumento crea una cadena vacía.

```
String empty = new String();
```

- Es más común inicializar una cadena con la cadena vacía. (Comunmente usado para reinicializar la cadena)

```
String empty = ""; //nothing between quotes
```

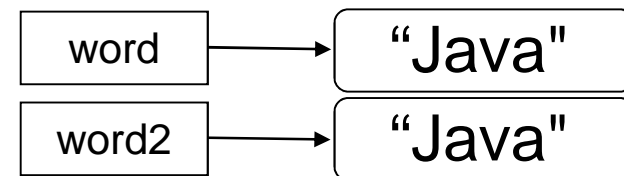
Constructores que copian.

12

- Un constructor que recibe una cadena, crea una copia de la misma. No es muy común.
- No es lo mismo que una asignación.

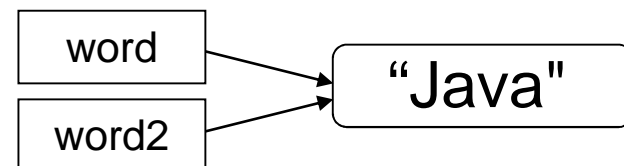
Cada variable apunta a una cadena diferente.

```
String word = new String("Java");  
String word2 = new String(word);
```



Ambas variables apuntan a la misma cadena en memoria.

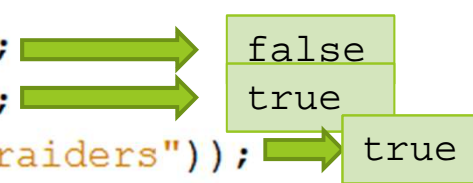
```
String word = "Java";  
String word2 = word;
```



```
public class EjemploCopia {  
  
    public static void main (String [] args){  
        String palabra = "Java";  
        String palabra2 = new String (palabra);  
        System.out.println ("Palabra: "+palabra);  
        System.out.println ("Palabra2: "+palabra2);  
  
        if (palabra == palabra2){  
            System.out.println ("Es la misma cadena");  
        }  
        else{  
            System.out.println ("Son diferentes");  
        }  
    }  
}
```

```
public class EjemploCopia {  
  
    public static void main (String [] args){  
        String palabra = "Java";  
        String palabra2 = palabra;  
        System.out.println ("Palabra: "+palabra);  
        System.out.println ("Palabra2: "+palabra2);  
  
        if (palabra == palabra2){  
            System.out.println ("Es la misma cadena");  
        }  
        else{  
            System.out.println ("Son diferentes");  
        }  
    }  
}
```

```
public class EjemploEquals {  
    public static void main (String [] args){  
  
        String equipo = "Raiders";  
  
        System.out.println(equipo.equals("raiders"));  
        System.out.println(equipo.equals("Raiders"));  
        System.out.println(equipo.equalsIgnoreCase("raiders"));  
    }  
}
```



System.out.println(equipo.equals("raiders"));	false
System.out.println(equipo.equals("Raiders"));	true
System.out.println(equipo.equalsIgnoreCase("raiders"));	true

Otros constructores

16

Se puede convertir un arreglo de caracteres en una cadena. Y viceversa.

```
char[] letters = {'J', 'a', 'v', 'a'};  
String word = new String(letters); // "Java"  
  
char [] caracteres = word.toCharArray ();
```


Métodos — length, charAt

17

`int length();`

- Regresa el número de caracteres en la cadena.

`char charAt(i);`

- Regresa el caracter en la posición i.

La posición de los caracteres en la cadena inicia en 0 – tal como en los arreglos.

Regresa:

`"Problem".length();`

7

`"Window".charAt (2);`

'n'

Métodos — substring

18

Regresa una nueva cadena copiando caracteres de una existente..

- `String subs = word.substring (i, k);`
 - ▣ Regresa la subcadena de caracteres comprendida entre los índices *i* y *k-1*
- `String subs = word.substring (i);`
 - ▣ Regresa la subcadena a partir del *i*-ésimo caracter y hasta el final.

television
↑ ↑
i *k*

television
↑
i

```
"television".substring (2,5);  
"immutable".substring (2);  
"bob".substring (9);
```

Returns:

→ "lev"
→ "mutable"
→ "" (empty string)

```
public class EjemploMetodos {  
    public static void main (String [] args){  
  
        String television = "television";  
  
        System.out.println (television.length());  
        System.out.println (television.charAt(3));  
        System.out.println (television.substring (2,5));  
        System.out.println (television.substring (5));  
  
    }  
}
```

Métodos — Concatenación


20

```
String word1 = "re";
```

```
String word2 = "think";
```


```
String word3 = "ing";
```

```
int num = 2;
```

□ `String result = word1 + word2;`  rethink

□ `String result = word1.concat(word2);`  rethink

□ `result += word3;`  rethinking

□ `result += num;`  rethinking2

```
public class EjemploIndexOf {
```

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

```
        012345678901234567890123456
```

```
        String name = "President George Washington";
```

```
        System.out.println (name.indexOf('P'));
```

0

```
        System.out.println (name.indexOf('e'));
```

2

```
        System.out.println (name.indexOf("George"));
```

10

```
        System.out.println (name.indexOf('e', 3));
```

6

```
        System.out.println (name.indexOf("Bob"));
```

-1

```
        System.out.println (name.lastIndexOf('e'));
```

15

```
    }
```

```
}
```

Métodos — Comparación

22

```
int diff = word1.compareTo(word2);  
    regresa la "diferencia" word1 - word2
```

```
int diff = word1.compareToIgnoreCase(word2);  
    regresa la diferencia word1 - word2,  
    sin fijarse en mayúsculas o minúsculas
```

Comparison Examples

23

```
//negative differences
```

```
diff = "apple".compareTo("berry");//a before b
```

```
diff = "Zebra".compareTo("apple");//Z before a
```

```
diff = "dig".compareTo("dug");//i before u
```

```
diff = "dig".compareTo("digs");//dig is shorter
```

```
//zero differences
```

```
diff = "apple".compareTo("apple");//equal
```

```
diff = "dig".compareToIgnoreCase("DIG");//equal
```

```
//positive differences
```

```
diff = "berry".compareTo("apple");//b after a
```

```
diff = "apple".compareTo("Apple");//a after A
```

```
diff = "BIT".compareTo("BIG");//T after G
```

```
diff = "huge".compareTo("hug");//huge is longer
```

Métodos — trim

24

```
String word2 = word1.trim ();
```

Elimina los espacios de la cadena al inicio y al final.

Regresa la cadena sin espacios.

```
String word1 = “ Hi Bob “;  
String word2 = word1.trim();
```

```
//word2 is “Hi Bob” – no spaces on either end  
//word1 is still “ Hi Bob “ – with spaces
```


Methods — replace

25

`String word2 = word1.replace(oldCh, newCh);`
regresa una cadena reemplazando los caracteres
oldCh por **newCh** en **word1**

```
String word1 = "rare";  
String word2 = "rare".replace('r', 'd');
```

```
//word2 es "dade",  
//word1 es "rare"
```

Methods — Changing Case

26

```
String word2 = word1.toUpperCase();
```

```
String word3 = word1.toLowerCase();
```

Regresa una cadena convirtiendo los caracteres a mayúsculas o a minúsculas

```
String word1 = "HeLLo";
```

```
String word2 = word1.toUpperCase();//"HELLO"
```

```
String word3 = word1.toLowerCase();//"hello"
```

```
//word1 is still "HeLLo"
```

Replacements

27

- Para modificar la cadena debe hacerse una asignación.

```
word1 = word1.toUpperCase();
```

- Error común:

```
word1.toUpperCase();
```

word1
remains
unchanged

Numbers to Strings

28

Existen varias maneras de convertir números en cadenas.

1. `String s = "" + num;`
2. `String s = Integer.toString (i);`
`String s = Double.toString (d);`
3. `String s = String.valueOf (num);`

Preguntas de participación:



29

1. ¿A qué paquete pertenece la clase String?
2. ¿Qué tiene la clase String que otras clases no tienen?
3. ¿Qué representa el texto que se encuentra entre comillas?
4. ¿Qué regresaría el método "Rumplestiltskin".length()?
5. ¿Qué es un objeto inmutable?



Review (cont):

30

6. ¿Por qué los objetos inmutables son más eficientes?
7. ¿Porqué los objetos inmutables son menos eficientes?
8. ¿Cómo se declara una cadena vacía?
9. ¿“Bob” + “ “ + “Smith” se llama ____ ?
10. ¿Cómo creo una copia de una cadena?



Review (cont'd):



31

```
String city = "Bloomington";
```

¿Qué regresa :

11. `city.charAt (2)?`
12. `city.substring(2, 4)?`
13. `city.lastIndexOf('o')?`
14. `city.indexOf(3)?`
15. ¿Qué hace el método `trim`?



Review (cont):

32

16. `"sam".equals("Sam")` ¿regresa?
17. ¿Qué tipo de dato regresa `"sam".compareTo("Sam")`?
18. ¿Qué se almacenará en `s`?
`s = "mint".replace('t', 'e');`
19. ¿Qué le hace `s.toUpperCase()` a `s`?
20. Di una manera simple de convertir un número en cadena.

