COMPARACION DE CADENAS

Veamos distintas formas de comparar y de ordenar cadenas de texto en java, según lo que nos interese en cada momento.

Comparar con ==

En java se pueden comparar cadenas de texto de varias maneras.

La primera y más directa, es comparando con el ==. Esta comparación no compara las cadenas, sino si la instancia es la misma. Así, por ejemplo

```
String cadena1 = new String("Hola");
String cadena2 = new String("Hola");
if (cadena1 == cadena2)
{
   ...
}
```

da false, las cadenas no son iguales, ya que son dos instancias -dos news- distintos.

El compilador de java es listo y si hacemos

```
String cadena1 = "Hola";
String cadena2 = "Hola";
if (cadena1 == cadena2)
{
   ...
}
```

daría true, ya que el compilador ve que tiene dos veces la misma cadena y crea una única instancia para ella -sólo hace internamente un new-.

Comparar con equals()

Para comparar realmente las cadenas y no si son o no la misma instancia, se usa el método equals().

```
String cadena1 = new String("Hola");
String cadena2 = new String("Hola");
if (cadena1.equals(cadena2))
{
   ...
}
```

esto daría true siempre que ambas cadenas tengan el mismo texto dentro. Es posible hacer cosas como esta

```
String cadena2 = new String("Hola");
if ("Hola".equals(cadena2))
{
   ...
}
```

es decir, podemos usar directamente el método equals() sobre una cadena de texto literal, sin necesidad de tener variable. Esto es útil para evitar comparar primero si la cadena es null antes de llamar a equals(). Es decir, evita hacer esto

```
String cadena2 = new String("Hola");
if (cadena2!=null)
   if (cadena2.equals("Hola"))
   {
      ...
   }
```

Comparar sin importar mayúsculas (ignore case)

Para comparar dos cadenas sin importar si los caracteres están en mayúscula o minúscula, se utiliza el método equalsIgnoreCase() de la clase String

```
String a = "hola";
String b = "HOLA";

// son iguales
if (a.equalsIgnoreCase(b)) {
   System.out.println("a y b son iguales");
}
```

Ordenar cadenas con compareTo()

Si lo que queremos es comparar cadenas para ordenarlas, una opción es usar el método compareTo() de la clase String. Este método devuelve 0 si ambas cadenas tienen el mismo contenido, negativo si el String es menor -va antes- que el parámetro que se le pasa y positivo si es mayor. Es decir

```
if (cadena1.compareTo(cadena2) == 0)
   System.out.println("cadena1 y cadena2 son iguales");
else
if (cadena1.compareTo(cadena2) < 0)
  System.out.println ("cadena1 va antes que cadena2");
else
if (cadena1.compareTo(cadena2) > 0)
  System.out.println("cadena2 va después que cadena1");
```

Ordenar con Collator

El problema de compareTo() es que compara según estén ordenados los caracteres en la tabla de caracteres que se usen. Por ello, una A mayúscula va antes que una a minúscula y esta va antes que una á con acento.

Si queremos un orden alfabético real, dependiendo del idioma en el que estemos, debemos usar la clase Collator. Esta clase sí realiza una comparación alfabética tal cual la entendemos los humanos.

Podemos configurarla para que trabaje igual que el compareTo() -es decir, 'á', 'A' y 'a' son distintas-, para que considere A y a iguales entre sí, pero distintas de á o bien para que considere que las tres son iguales.

Una constantes define este comportamiento

Collator.PRIMARY considera distintas dos letras si la letra base es distinta. Es decir, serían iguales A, a y á, puesto que en todos casos la letra base es la a.

Collator.SECUNDARY considera iguales mayúsculas y minúsculas, pero distintas si tienen acento.

Collator.TERCIARY considera distintas las tres letras.

Collator.IDENTICALLY sólo las considera iguales si su código por debajo. Por ejemplo, se puede codificar internamente una á con acento de varias formas. Con este nivel, se considerarían distintas, aunque ambas se vean como á con acento.

Un ejemplo de código

```
Collator comparador = Collator.getInstance();
comparador.setStrength(Collator.PRIMARY);
// Estas dos cadenas son iguales
System.out.println(comparador.compare("Hóla", "hola"));

comparador.setStrength(Collator.SECUNDARY);
// Ahora son distitnas por el acento
System.out.println(comparador.compare("Hóla", "hola"));
// pero estas otras dos son iguales
System.out.println(comparador.compare("Hola", "hola"));

comparador.setStrength(Collator.TERCIARY);
// Ahora son distintas por la mayúscula
System.out.println(comparador.compare("Hola", "hola"));
// pero estas sí son iguales
System.out.println(comparador.compare("hola", "hola"));
```