PROGRAMACIÓN UD-6

CADENAS DE CARACTERES

ÍNDICE

[CARACTERES Y CADENAS 3](#_Toc122072773)

[CHARACTER 3](#_Toc122072774)

[CLASE STRING 4](#_Toc122072775)

[BÚSQUEDA 5](#_Toc122072776)

[COMPROBACIONES 5](#_Toc122072777)

[CONVERSIONES Y SEPARACIÓN EN PARTES 5](#_Toc122072778)

# CARACTERES Y CADENAS

Hasta ahora con los tipos primitivos sólo disponíamos del tipo char para trabajar con caracteres, pero su uso es un poco limitado.

Las cadenas de caracteres las definimos como un número determinado de caracteres, que pueden contener caracteres especiales como saltos de línea para formar párrafos distintos.

Las cadenas se escriben entre comillas dobles “cadena2, mientras que los caracteres se escriben entre comillas simples ‘c’.

Tabla

Descripción generada automáticamenteUn carácter se puede escribir como un literal carácter chrCaracter = ‘a’, como su código en decimal chrCaracter = 97 o mediante su valor en hexadecimal chrCaracter = ‘\u0061’ (4 dígitos precedidos de \u).

Existen también caracteres especiales que se pueden escribir directamente con una barra invertida (\).

En los últimos estándares de Unicode, los caracteres ocupan 3 bytes. En Java el tipo char solo admite dos bytes.

# CHARACTER

Existe una clase envoltorio llamada Character que permite realizar una serie de funciones que pueden facilitar el trabajo.

* Character.isDigit(‘a’) – Devuelve true si el carácter es un dígito.
* Character.isLetter(‘a’) – Devuelve true si el carácter es una letra.
* Character.isLetterOrDigit(‘a’) – Devuelve true el carácter es un alfanumérico.
* Character.isLowerCase(‘a’) – Devuelve true el carácter está en minúscula.
* Character.isUpperCase(‘a’) – Devuelve true el carácter está en mayúscula.
* Character.isWhitespace(‘a’) – Devuelve true si el carácter es un espacio en blanco (barra espaciadora, retorno de carro, nueva línea, tabulador…).
* Character.toLowerCase(‘a’) – Convierte el carácter a minúscula, si no lo es lo deja tal cual.
* Character.toUpperCase(‘a’) – Convierte el carácter a mayúscula, si no lo es lo deja tal cual.

Se pueden realizar también operaciones matemáticas sobre un tipo carácter, para que seleccione un carácter anterior o posterior, por ejemplo,

* System.out.println(‘h’ + ‘A’ – ‘a’) – Devuelve una H.
* System.out.println(‘e’ – ‘2’) – Devuelve una c.

# CLASE STRING

Las cadenas son secuencias de caracteres que se manipulan mediante la clase String.

Los literales de tipo String vendrán delimitados por comillas dobles “Hola Mundo”. La definición de una variable de tipo String es similar a la de los tipos básicos, String strCadena = “Hola Mundo”.

Es posible inicializar la cadena como un objeto más String strCadena = new String(“Hola Mundo”);

Cuando aplicamos el método valueOf(), de la clase String, sobre cualquier variable de tipo primito obtendremos una cadena que representa dicho objeto. Con los objetos se puede usar toString().

Para comparar cadenas no puedo utilizar los operadores ==, >, <… es necesario usar métodos:

* equals – compara si son iguales dos cadenas strCadenal.equals(strCadena2);
* equalsIgnoreCase – Igual que equals pero ignorando mayúsculas y minúsculas
* strCadenal.equa1sIgnoreCase(strCadena2);
* regionMatches – Indica si se encuentra una cadena en otra. Se pasan como parámetros (ignora mayúsculas, minúsculas, posición en la cadena que invoca, segunda cadena, posición en la segunda, numero de caracteres a comparar). El primer parámetro es opcional. String strSaludo = “Hola Mundo”, strPrueba = “Donde está Mundo”; strSaludo.regionMatches(5,strPrueba,11,5); devolverá true.
* compareTo – compara alfabéticamente dos cadenas, devolviendo: 0 si son iguales, <0 si la cadena que hace la llamada es menor y >0 si la cadena que hace la llamada es mayor. strCadena1.compareTo(strCadena2);
* compareToIgonreCase – Igual que el anterior, pero sin tener en cuenta mayúsculas y minúsculas.
* concat – Permite concatenar cadenas, aunque es raro su uso porque el operador + hace la misma función. strCadena1.concat(strCadena2);
* charAt – Devuelve el carácter indicado de una cadena. El primer carácter de una cadena es el 0.
* substring – devuelve una subcadena de la cadena original. Se puede indicar desde donde queremos que arranque la subcadena y que llegue hasta el final strCadena1.substring(4); También es posible indicar el principio y el final de la subcadena strCadena1.substring(4, 3);
* strip, stripLeading, stripTrailing – se utiliza para eliminar los espacios en blanco del principio y final de una cadena en el primer caso, sólo del principio en el segundo o solo del final en el tercero.
* length – indica el número de caracteres de una cedena. strCadena.length(); Se suele utilizar en bucles con una función charAt.

# BÚSQUEDA

Dentro de una cadena es posible buscar caracteres o cadenas, en un sentido u otro y a partir de una posición indicada. Estas funciones devuelven el índice donde se ha localizado o -1 si no se encuentra.

* indexOf (carácter | cadena [, posición\_inicial]) – Busca la primera ocurrencia de la cadena o el carácter. Si se indica la posición inicial, empezará desde esta. La posición inicial se suele utilizar si queremos buscar otro carácter tras el primero.
* lastIndexOf (carácter | cadena [, posición\_inicial]) – Busca la última ocurrencia de la cadena o el carácter.

# COMPROBACIONES

Dentro de una cadena es posible utilizar métodos que me devolverán un booleano (verdadero, falso) al hacer una comprobación (¿está vacía la cadena?, ¿contiene esta subcadena?, ¿empieza o termina por?...)

* isEmpty – Indica si la cadena está vacía. strNombreCadena.isEmpty(); Devuelve true si está vacía y false si está llena.
* contains – Busca si dentro de la cadena contiene la subcadena pasada, si es así devuelve true, sino devolverá false. strNombreCadena.contains(strNombreSucadena);
* startsWith – Devuelve true si la cadena comienza por la subcadena pasada. Si se tiene la posición de inicio, mirará desde esa posición. strNombreCadena.startsWith(strNombreSubcadena [, intPosicionInicio]);
* endsWith – Igual que el caso anterior pero ahora mira por donde termina la cadena. strNombreCadena.endsWith(strNombreSubcadena)

# CONVERSIONES Y SEPARACIÓN EN PARTES

* replace – Devuelve la cadena pasada, donde se han sustituido todas las ocurrencias del carácter original por el sustituto. strNuevaCadena = strNombreCadena.replace(chrCaracterOrigen, chrCaracterSustituto); En lugar de caracteres se pueden introducir cadenas a sustituir strNuevaCadena = strNombreCadena.replace(strCadenaOrigen, strCadenaSustituta);
* split – Devuelve un array de String que contiene las cadenas entre separadores. strArrCadenas = strNombreCadena.split(chrCaracterSeparador); Si es un carácter especial usar \\ antes del carácter.
* toCharArray – Devuelve un array de caracteres con los caracteres de la cadena que hace la llamada. chrArrCaracteres = strNombreCadena.toCharArray();
* valueOf – Devuelve de un array de caracteres el String formado por todos los caracteres del array. strNuevaCadena = String.valueOf(chrArrCaracteres[, intInicio, intNumCaract]); Se puede indicar desde donde se empieza con intInicio y cuantos caracteres queremos coger con intNumCaract.