Programación

Act06_Cadenas_de_caracteres

Francisco José García Cutillas | 1FPGS_DAM

Índice

Ejercicio A01

Implementa y utiliza un método que reciba una cadena de texto y devuelva un array en el que en la primera posición de este indique el número de 'a' que existen, en la segunda el número de 'e', tercera 'i', cuarta 'o' y quinta 'u'. No debe distinguir entre mayúsculas y minúsculas (ojo con las tildes). Si recibe la frase "Guillermo Palazón" devolverá el array [2,1,1,2,1]

```java
package com.mycompany.garciacutillasfranciscojoseactut06cadenascaracteres;

/**
 *
 * @author Fran
 */
public class GarciaCutillasFranciscoJoseActUt06CadenasCaracteres {

    public static void main(String[] args) {

     Ejercicios ej = new Ejercicios();

     ej.ejercicio1();

    }
}
```

```java
package com.mycompany.garciacutillasfranciscojoseactut06cadenascaracteres;

import java.util.Arrays;

/**
 *
 * @author Fran
 */
public class Ejercicios {

    Metodos fun = new Metodos();

    public void ejercicio1() {

        String entrada = "AÁl2aá  eÉé-+*E oOOóÓdsuuÚú iíÍI";

        System.out.println( x: Arrays.toString( a: fun.cuentaLetras(entrada)));

    }

}
```

```java
package com.mycompany.garciacutillasfranciscojoseactut06cadenascaracteres;

import java.util.Arrays;

/**
 *
 * @author Fran
 */
public class Metodos {

    public int[] cuentaLetras(String entrada) {

        int resultado[] = new int[5]; //Array que va a devolver el método
        char arrayTemp[] = new char[entrada.length()]; //Array temporal con el que vamos a hacer las operaciones en el método
        arrayTemp = entrada.toCharArray(); //Convertimos el String de entrada en un array de char

        //Recorremos el array en busca de las vocales y las contamos
        for (int i = 0; i < arrayTemp.length; i++) {

            switch (arrayTemp[i]) {

                case 'a', 'á', 'A', 'Á':

                    resultado[0]++;
                    break;

                case 'e', 'é', 'E', 'É':

                    resultado[1]++;
                    break;

                case 'i', 'í', 'I', 'Í':

                    resultado[2]++;
                    break;

                case 'o', 'ó', 'O', 'Ó':

                    resultado[3]++;
                    break;

                case 'u', 'ú', 'U', 'Ú':

                    resultado[4]++;
                    break;

            }

        }

        return resultado;

    }

}
```

```
--< com.mycompany:GarciaCutillasFranciscoJoseActUt06CadenasCaracteres >--
Building GarciaCutillasFranciscoJoseActUt06CadenasCaracteres 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[4, 4, 4, 5, 4]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.347 s
Finished at: 2023-02-05T11:56:14+01:00
```

Ejercicio A02

Implementa y utiliza un método que reciba tres cadenas de texto y devuelva un código a partir de las mismas. Este código se formará uniendo los 3 primeros caracteres de la primera y tercera palabra y los dos últimos de la segunda. Por ejemplo si recibe (“JUAN”, ”MARÍA”, ”LUCAS”) devolverá JUAÍALUC.

```
        ej.ejercicio2();

    }
}
```

```
public void ejercicio2(){

    String cadena1 = "JUAN";
    String cadena2 = "MARÍA";
    String cadena3 = "LUCAS";

    System.out.println( x: fun.concatena(cadena1, cadena2, cadena3));

}
```

```
public String concatena(String cadena1, String cadena2, String cadena3) {

    String salida = ""; //Cadena que va a devolver el método

    //Caso de error. Cadenas 1 y 3 menores de 3 caracteres o cadena 2 menor de 2
    if ((cadena1.length() < 3 || cadena2.length() < 2 || cadena3.length() < 3)) {

        salida = "Error, la cadena 1 y 3 no pueden tener menos de tres caracteres y la 2 no puede tener menos de 2.";

        //Si no hay error
    } else {

        salida = cadena1.substring( beginIndex: 0, endIndex: 3) + cadena2.substring(cadena2.length() - 2, endIndex: cadena2.length())
                + cadena3.substring( beginIndex: 0, endIndex: 3);

    }

    return salida;

}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
JUAÍALUC
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.348 s
Finished at: 2023-02-05T12:22:24+01:00
```

```
String cadena1 = "Pepe";
String cadena2 = "A";
String cadena3 = "Isa";
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
Error, la cadena 1 y 3 no pueden tener menos de tres caracteres y la 2 no puede tener menos de 2.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.331 s
Finished at: 2023-02-05T12:50:22+01:00
------------------------------------------------------------------------
```

Ejercicio A03

Implementa y utiliza un método que devuelva un array de caracteres a partir de tres palabras que se vayan a pasar por parámetro. Este array de caracteres se formará de tal manera. Primero se generará un código tal que el del ejercicio A02, pero ahora todos los caracteres de estas palabras se obtienen desde el principio de la palabra y el número de caracteres que obtenemos será aleatorio (ojo: debemos tener en cuenta la longitud de la palabra concreta). Por ejemplo si recibe ("JUAN", "MARÍA", "LUCAS") y para la primera palabra devuelve un 2, para la segunda un 3 y para la tercera un 1 el código sería JUMARL, por lo que el array a devolver será ['J','U','M','A','R','L').

```
        ej.ejercicio3();

    }
```

```
public void ejercicio3(){

    String cadena1 = "JUAN";
    String cadena2 = "MARÍA";
    String cadena3 = "LUCAS";

    System.out.println( x: Arrays.toString( a: fun.arrayCharAleatorio(cadena1, cadena2, cadena3)));

}
```

```
public char[] arrayCharAleatorio(String cadena1, String cadena2, String cadena3) {

    /*Cadena temporal para realizar las operaciones. Aquí vamos a sacar el resultado de generar la concatenación
    de las tres cadenas, habiendo obtenido de cada una de ellas los caracteres aleatoriamente. Puede darse el caso
    de que no salga ningún carácter de alguna cadena puesto que el aleatorio es entre 0 y longitud de cadena*/
    String temp = cadena1.substring( beginIndex: 0, (int) (Math.random() * ((cadena1.length() - 0) + 1)) + 0)
            + cadena2.substring( beginIndex: 0, (int) (Math.random() * ((cadena2.length() - 0) + 1)) + 0)
            + cadena3.substring( beginIndex: 0, (int) (Math.random() * ((cadena3.length() - 0) + 1)) + 0);

    //Creación del array salida con la longitud de la cadena aleatoria anterior
    char salida[] = new char[temp.length()];

    //Paso de la cadena a un array de char para que lo devuelva el método
    salida = temp.toCharArray();

    return salida;

}
```

```
Building GarciaCutillasFranciscoJoseActUt06CadenasCaracteres 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[M, A, R, Í, L, U]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.331 s
Finished at: 2023-02-05T13:32:27+01:00
```

```
Building GarciaCutillasFranciscoJoseActUt06CadenasCaracteres 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[L, U]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.344 s
Finished at: 2023-02-05T13:34:19+01:00
------------------------------------------------------------------------
```

```
Building GarciaCutillasFranciscoJoseActUt06CadenasCaracteres 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[J, M, L]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.346 s
Finished at: 2023-02-05T13:34:42+01:00
------------------------------------------------------------------------
```

Ejercicio A04

Implementa y utiliza un método que reciba un array de cadenas de texto y devuelva un array de enteros con las longitudes de dichas cadenas y devuelva al final en un elemento independiente el índice en el que se encontraba la cadena de mayor longitud.

```
ej.ejercicio4();
```

```
public void ejercicio4(){

    String arrayEntrada[] = {"Hola, me llamo Antonio", "Pepe jugaba", "Programación"};

    System.out.println( x: Arrays.toString( a: fun.longitudCadena(arrayEntrada)));

}
```

```
public int[] longitudCadena(String arrayEntrada[]) {

    int salida[] = new int[arrayEntrada.length + 1]; //Array de salida
    int longMayor = arrayEntrada[0].length(); //Variable para guardar la longitud mayor
    salida[salida.length - 1] = 0; // Suponemos que el mayor es el del índice 0

    //Recorremos el array para contar la longitud de las cadenas, buscamos la mayor y guardamos su índice
    for (int i = 0; i < arrayEntrada.length; i++) {

        salida[i] = arrayEntrada[i].length();

        if (arrayEntrada[i].length() > longMayor) {

            salida[salida.length - 1] = i;
            longMayor = arrayEntrada[i].length();

        }

    }

    return salida;

}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[22, 11, 12, 0]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.323 s
Finished at: 2023-02-07T17:03:45+01:00
------------------------------------------------------------------------
```

Ejercicio A05

Implementa y utiliza un método que reciba una cadena de texto y reciba un array con dos cadenas de texto, la primera debe contener la primera mitad de la cadena original y la segunda la segunda mitad. Por ejemplo, si recibe “La chica de nieve” devolverá [“La chica “,”de nieve”]

```
ej.ejercicio5();
```

```
public void ejercicio5(){

    String entrada = "La chica de nieve";

    System.out.println( x: Arrays.toString( a: fun.mitadCadena( cadenaEntrada: entrada)));

}
```

```
public String[] mitadCadena(String cadenaEntrada) {

    String salida[] = new String[2]; //Array salida

    //Primer elemento del array salida, desde el índice 0 hasta la mitad de la cadena de entrada
    salida[0] = cadenaEntrada.substring( beginIndex: 0, cadenaEntrada.length() / 2);

    //Segundo elemento del array salida, desde el índice mitad de cadena de entrada hasta el índice final
    salida[1] = cadenaEntrada.substring(cadenaEntrada.length() / 2, endIndex: cadenaEntrada.length());

    return salida;

}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[La chica,  de nieve]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.319 s
Finished at: 2023-02-07T17:22:44+01:00
------------------------------------------------------------------------
```

Ejercicio A06

Implementa y utiliza un método que reciba una cadena de texto y devuelva un array con la letra que más se ha repetido (sea vocal o consonante), la vocal que más se ha repetido (aunque coincida con la anterior) y la vocal que menos se ha repetido. Además, el método debe tener un parámetro true/false para indicar si vamos a diferenciar o no por mayúsculas/minúsculas.

```
ej.ejercicio6();
```

```
public void ejercicio6() {

    String entrada = "OaeaOotoOTTattUiIHtHiHHeIH43i4o5";

    System.out.println( x: Arrays.toString( a: fun.cuentaCaracteres( cadenaEntrada: entrada,  distingueMayMin: true)));

}
```

```
public char[] cuentaCaracteres(String cadenaEntrada, boolean distingueMayMin) {

    //Array salida
    char salida[] = new char[3];

    //Creamos un array temporal de char para operar con él en el método
    char arrayTemp[] = new char[cadenaEntrada.length()];

    //Pasamos la cadena de entrada a un array de char
    arrayTemp = cadenaEntrada.toCharArray();

    //Caso en el que distigamos entre mayúsculas y minúsculas
    if (distingueMayMin) {

        salida[0] = cuentaLetraVocalCons( arrayEntrada: arrayTemp);
        salida[1] = cuentaVocalMax( arrayEntrada: arrayTemp);
        salida[2] = cuentaVocalMin( arrayEntrada: arrayTemp);

        //Sin distinción entre mayúsculas y minúsculas
    } else {

        //Primero pasamos todos los caracteres a minúscula, para buscar después
        for (int i = 0; i < arrayTemp.length; i++) {

            arrayTemp[i] = Character.toLowerCase(arrayTemp[i]);

        }

        salida[0] = cuentaLetraVocalCons( arrayEntrada: arrayTemp);
        salida[1] = cuentaVocalMax( arrayEntrada: arrayTemp);
        salida[2] = cuentaVocalMin( arrayEntrada: arrayTemp);

    }

    return salida;

}
```

```java
//Este método devuelve la letra que más se ha repetido. Si se da el caso de que haya dos letras que se repitan las mismas
//veces, devuelve la primera que haya visto
public char cuentaLetraVocalCons(char arrayEntrada[]) {

    //Primeramente suponemos que la letra más repetida, vocal más repetida y menos repetida, va a ser la
    //primera del array
    char letraRep = arrayEntrada[0];

    //También vamos a usar un contador para saber las veces que se repite cada letra y un contador temporal
    int contadorLetra = 0;
    int conTemp = 0;

    //Recorremos el array de char para contar las letras que más se repiten
    for (int i = 0; i < arrayEntrada.length; i++) {

        //Primero vemos que el caracter sea una letra
        if (Character.isLetter(arrayEntrada[i])) {

            //Recorremos de nuevo para comparar con el índice i
            for (int j = 0; j < arrayEntrada.length; j++) {

                //Si la letra coincide con la del índice "i" suma al contador temporal
                if (arrayEntrada[j] == arrayEntrada[i]) {

                    conTemp++;

                }

            }

            //Si al finalizar la búsqueda está más veces que la anterior, sobreescribe la nueva
            if (conTemp > contadorLetra) {

                contadorLetra = conTemp;
                letraRep = arrayEntrada[i];

            }

            //Reinicializamos contador temporal para la siguiente vuelta
            conTemp = 0;

        }

    }

    return letraRep;

}
```

```java
//Este método devuelve la vocal que más se repite de un array de char de entrada
public char cuentaVocalMax(char arrayEntrada[]) {

    char salida = ' ';

    //Inicializamos las variables al primer valor del array
    salida = arrayEntrada[0];

    //Contadores
    int contVocalMax = 0;
    int conTempMax = 0;

    //Recorremos el array en busca de las vocales
    for (int i = 0; i < arrayEntrada.length; i++) {

        //Tenemos que discriminar sólo vocales
        if (Character.isLetter(arrayEntrada[i]) && esVocal(arrayEntrada[i])) {

            for (int j = 0; j < arrayEntrada.length; j++) {

                if (arrayEntrada[j] == arrayEntrada[i]) {

                    conTempMax++;

                }

            }

            if (conTempMax > contVocalMax) {

                contVocalMax = conTempMax;
                salida = arrayEntrada[i];

            }

            //Reinicializamos contador
            conTempMax = 0;

        }

    }

    return salida;

}
//Este método devuelve la vocal que menos se repite de un array de entrada
public char cuentaVocalMin(char arrayEntrada[]) {

    char salida = ' ';

    //Creamos un array para guardar las vocales y otro para su conteo
    char vocalesMin[] = {'a', 'e', 'i', 'o', 'u'};
    char vocalesMay[] = {'A', 'E', 'I', 'O', 'U'};
    int numVocalesMin[] = new int[5];
    int numVocalesMay[] = new int[5];

    //Recorremos el array para contar las vocales
    for (int i = 0; i < arrayEntrada.length; i++) {

        //Caso en el que las vocales sean minúsculas
        if (esVocal(arrayEntrada[i]) && Character.isLowerCase(arrayEntrada[i])) {

            switch (arrayEntrada[i]) {

                case 'a':
                    numVocalesMin[0]++;
                    break;
                case 'e':
                    numVocalesMin[1]++;
                    break;
                case 'i':
                    numVocalesMin[2]++;
                    break;
                case 'o':
                    numVocalesMin[3]++;
                    break;
                case 'u':
                    numVocalesMin[4]++;
                    break;

            }
```

```
            //Caso en el que las vocales sean mayúsculas
        } else if (esVocal(arrayEntrada[i]) && Character.isUpperCase(arrayEntrada[i])) {

            switch (arrayEntrada[i]) {

                case 'A':
                    numVocalesMay[0]++;
                    break;
                case 'E':
                    numVocalesMay[1]++;
                    break;
                case 'I':
                    numVocalesMay[2]++;
                    break;
                case 'O':
                    numVocalesMay[3]++;
                    break;
                case 'U':
                    numVocalesMay[4]++;
                    break;

            }

        }

    }
```

```
    //Ahora vamos a buscar la que menos se repite
    int contVocMay = 0;
    int contVocMin = 0;
    int indiceMin = 0;
    int indiceMay = 0;

    //Minúsculas
    for (int i = 0; i < numVocalesMin.length; i++) {

        if (contVocMin == 0 && numVocalesMin[i] > 0) {

            contVocMin = numVocalesMin[i];
            indiceMin = i;

        } else if (numVocalesMin[i] < contVocMin && numVocalesMin[i] > 0) {

            contVocMin = numVocalesMin[i];
            indiceMin = i;

        }

    }

    //Mayúsculas
    for (int i = 0; i < numVocalesMay.length; i++) {

        if (contVocMay == 0 && numVocalesMay[i] > 0) {

            contVocMay = numVocalesMay[i];
            indiceMay = i;

        } else if (numVocalesMay[i] < contVocMay) {

            contVocMay = numVocalesMay[i];
            indiceMay = i;

        }

    }
```

```
        //Si la vocal que menos se repite es mayúscula
        if (contVocMay < contVocMin && contVocMay > 0) {

            salida = vocalesMay[indiceMay];

            //Si la vocal que menos se repite es minúscula
        } else {

            salida = vocalesMin[indiceMin];

        }

        return salida;
    }

    //Método para saber si un carácter es vocal
    public boolean esVocal(char entrada) {

        char caracter = ' ';
        boolean salida = false;

        //Primero lo pasamos a minúscula
        caracter = Character.toLowerCase( ch: entrada);

        switch (caracter) {

            case 'a', 'e', 'i', 'o', 'u':

                salida = true;
                break;

        }

        return salida;

    }

}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[H, O, U]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.333 s
Finished at: 2023-02-08T16:36:52+01:00
```

```
public void ejercicio6() {

    String entrada = "OaeaOotoOTTattUiIHtHiHHeIH43i4o5";

    System.out.println( x: Arrays.toString( a: fun.cuentaCaracteres( cadenaEntrada: entrada,  distingueMayMin: false)));

}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[o, o, u]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.327 s
Finished at: 2023-02-08T16:46:32+01:00
```

```
public void ejercicio6() {

    String entrada = "aee";

    System.out.println( x: Arrays.toString( a: fun.cuentaCaracteres( cadenaEntrada: entrada,  distingueMayMin: false)));

}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ GarciaCutillasFranciscoJoseActUt06CadenasCaracteres ---
[e, e, a]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.334 s
Finished at: 2023-02-08T16:48:22+01:00
```