



APRENDIZAJE EN LAS MARATONES DE PROGRAMACIÓN – UNIVERSIDAD DE LA AMAZONIA

SEMILLERO DE INVESTIGACIÓN EN PROGRAMACIÓN

I. CONCEPTUALIZACIÓN

Hay que resaltar que en estas competencias de programación suelen proponer ejercicios relacionados con el manejo de cadenas de caracteres, probabilidades, diversos métodos matemáticos, y con seguridad, estructuras de datos. Estos problemas postulados son en su mayoría relacionados con aspectos de la vida real, lo cual conlleva a la solución de problemas de diversos grados de complejidad.

1. String

1.1. Split

Este es un método de java que se emplea para convertir una cadena de String a un arreglo, de acuerdo a unos parámetros en su formación.

Por ejemplo, si se tiene una cadena como “Semillero de programación”, este método propio de java permite crear un arreglo de tipo String cuyas posiciones corresponden a cada una de las palabras separadas por un espacio en blanco, tal como se muestra a continuación:

```
public static void main(String[] args) {  
  
    String cadena = "Semillero de Programación";  
    String[] array = cadena.split(" ");  
    for (int i = 0; i < array.length; i++) {  
        System.out.println("Pos[" + i + "]: " + array[i]);  
    }  
}
```

El código anterior genera el siguiente resultado:



```
Output - Mar (run) X
run:
Pos[0]: Semillero
Pos[1]: de
Pos[2]: Programación
BUILD SUCCESSFUL (total time: 1 second)
```

En este se puede ver claramente cómo a partir de una cadena de caracteres se puede construir un arreglo con cada una de las palabras que la conforman. Hay que resaltar que la separación normalmente se lleva a cabo por cada espacio en blanco presente en la cadena, sin embargo, se puede realizar de acuerdo a la necesidad del usuario.

1.2. Substring

Si se tiene una cadena de caracteres y se requiere recorrerla término a término, por ejemplo para hacer ciertas verificaciones en cada una de sus posiciones, se emplea un método de *java* llamado 'substring', el cual se usa de la siguiente manera:

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

    String cadena = "Semillero";
    for (int i = 0; i < cadena.length(); i++) {
        System.out.println("Caracter " + i + ": " + cadena.substring(i, i + 1));
    }
}
```

En este ejemplo se asigna a una variable *cadena* la palabra "Semillero". Luego, se recorre con el substring y se muestra término a término de la misma.

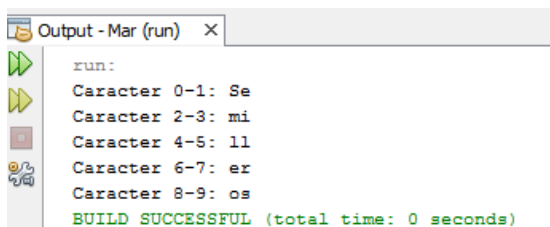
```
Output - Mar (run) X
run:
Caracter[0]: S
Caracter[1]: e
Caracter[2]: m
Caracter[3]: i
Caracter[4]: l
Caracter[5]: l
Caracter[6]: e
Caracter[7]: r
Caracter[8]: o
BUILD SUCCESSFUL (total time: 0 seconds)
```



Hay que resaltar que este método recibe como parámetro dos (2) valores; el primero, es la posición inicial de la cadena, y el segundo, la posición final. Estos valores pueden variar como en el siguiente ejemplo:

```
public static void main(String[] args) {  
    String cadena = "Semilleros";  
    for (int i = 0; i < cadena.length(); i++) {  
        System.out.println("Caracter " + i + "-" + (i + 1) + ": " + cadena.substring(i, i + 2));  
        i = i + 1;  
    }  
}
```

Y la salida:



```
run:  
Caracter 0-1: Se  
Caracter 2-3: mi  
Caracter 4-5: ll  
Caracter 6-7: er  
Caracter 8-9: os  
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.3. String.toCharArray

A diferencia del método anterior, con este podemos convertir una cadena a un arreglo cuyas posiciones serán cada uno de los caracteres que componen la cadena. Por ejemplo, la palabra “Maraton” se puede convertir en un arreglo de tipo *char*, cuyo tamaño será de exactamente el número de caracteres que tenga la cadena, en este caso 7.

```
public static void main(String[] args) {  
    String cadena = "Maraton";  
    char[] array = cadena.toCharArray();  
    for (int i = 0; i < array.length; i++) {  
        System.out.println("Pos[" + i + "]: " + array[i]);  
    }  
}
```

Y la salida de la siguiente manera:



```
Output - Mar (run) X
run:
Pos[0]: M
Pos[1]: a
Pos[2]: r
Pos[3]: a
Pos[4]: t
Pos[5]: o
Pos[6]: n
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.4. Arrays.sort

En ocasiones cuando es necesario organizar un arreglo ya sea de menor a mayor (en caso de arreglos con valores numéricos) o alfabéticamente (cuando el arreglo contiene caracteres), se puede emplear fácilmente un método de *java* para evitar otras opciones que tal vez serán menos efectivas. Este método pertenece a la clase *Arrays* y se llama *sort*. Hay que resaltar que el tipo del arreglo que esta función recibe como parámetro puede variar de acuerdo a la necesidad. Por ejemplo:

```
public static void main(String[] args) {
    String cadena = "Maraton";
    char[] array = cadena.toCharArray();

    Arrays.sort(array);

    for (int i = 0; i < array.length; i++) {
        System.out.println("Pos[" + i + "]: " + array[i]);
    }
}
```

Siguiendo con el ejemplo anterior, en esta oportunidad se desea organizar alfabéticamente el arreglo 'array' de tipo char.

```
run:
Pos[0]: M
Pos[1]: a
Pos[2]: a
Pos[3]: n
Pos[4]: o
Pos[5]: r
Pos[6]: t
```



Ahora un ejemplo con otro tipo de arreglo:

```
public static void main(String[] args) {  
  
    int[] array = new int[5];  
    array[0] = 3;  
    array[1] = 8;  
    array[2] = 1;  
    array[3] = 5;  
    array[4] = 2;  
  
    Arrays.sort(array);  
  
    for (int i = 0; i < array.length; i++) {  
        System.out.println("Pos[" + i + "]: " + array[i]);  
    }  
}
```

Se obtiene el siguiente resultado:

```
run:  
Pos[0]: 1  
Pos[1]: 2  
Pos[2]: 3  
Pos[3]: 5  
Pos[4]: 8
```

2. StringBuilder

2.1. Reverse

Como su nombre lo indica, este método se emplea para invertir o voltear una cadena. Usualmente cuando se requiere hacer esta operación con una cadena, se lleva a cabo de la siguiente manera:

```
public static void main(String[] args) {  
  
    String cadena = "Maraton", result = "";  
    for (int i = cadena.length() - 1; i >= 0; i--) {  
        result += cadena.substring(i, i + 1);  
    }  
    System.out.println(result);  
}
```

Y efectivamente funciona, pero esta opción implica un tiempo adicional de ejecución por el ciclo que contiene.