Manual de usuario: Programa “Recursión”

Estructura de Datos

Víctor de la Cueva

Daniel Roa

A01021960

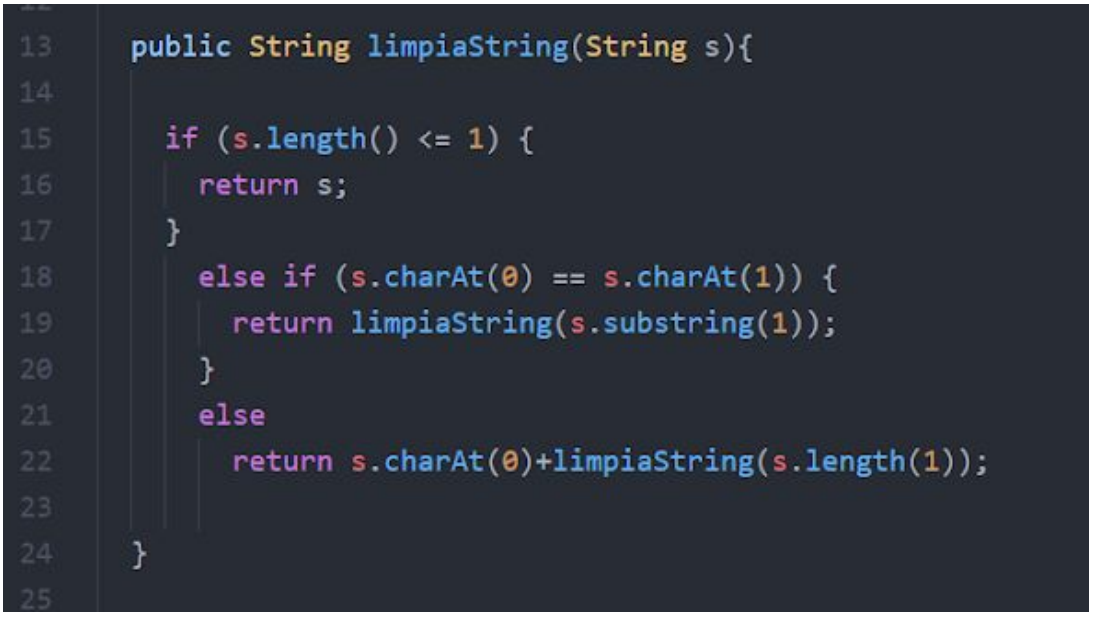
14 de enero del 2019

Explicación

El usuario deberá tener preparado una clase tipo *Main* en el cual se transfieren las operaciones que se desean hacer, en otras palabras, para que la sección correcta del código sea utilizada, es necesario insertar los datos solicitados para que los procesos puedan correr correctamente.

Funciones del Software

Este software (o *API*) tiene cuatro funciones principales que pueden ser accedidas desde el software tipo *Main* del usuario, estas tienen variables propias para que los procesos correctos puedan correr y no se encuentre con algún error al utilizar los datos otorgados.



```
13  public String limpiaString(String s){
14
15    if (s.length() <= 1) {
16      return s;
17    }
18      else if (s.charAt(0) == s.charAt(1)) {
19        return limpiaString(s.substring(1));
20      }
21      else
22        return s.charAt(0)+limpiaString(s.length(1));
23
24  }
25
```

(Fig. 1)

En la imagen llamada “Fig. 1”, se puede apreciar el código utilizado para cumplir una función en la que se inserta una combinación de caracteres que puedan estar siendo repetidos dentro de este *String* y esta sección se encarga de limpiarlo, es decir, se encarga de remover caracteres que se repitan si es que hay caracteres repetidos juntos. Utiliza una función llamada *substring* que se encarga de guardar los caracteres que ya fueron revisados. Es de suma importancia que el usuario utilice una variable tipo *String* para el funcionamiento correcto de este software.

Dudas y aclaraciones

Daniel Roa
a01021960@itesm.mx

```java
public int cuentaSubString(String s, String sub){

  int A = s.length();
  int B = sub.length();

  if (A == 0 ||A < B) {
    return 0;
  }

  else if (s.substring(0, B).equals(sub)) {
    return cuentaSubString(s.substring(B - 1), sub) + 1;
  }
  else
    return cuentaSubString(s.substring(B - 1), sub);

}//fin cuentaSubString
```

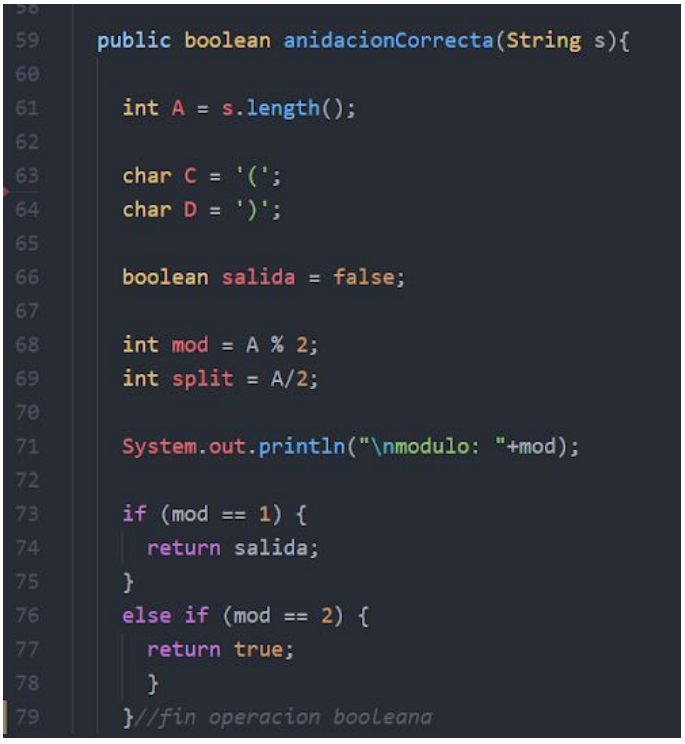
Fig. 2

En la imagen superior, llamada “Fig. 2”, en esta imagen se puede notar cómo para este proceso es vital que se utilizan dos variables de tipo *String*, el principal (llamado **s**) es donde el usuario insertará una selección de letras que se podrán encontrar dentro de la palabra original y esta misma sección imprimirá la cantidad de veces que se repite esa combinación de letras o la palabra insertada. En caso de que la palabra no se encuentre dentro del *String*, el sistema imprimirá un 0.

```java
public int sumaDigitos(int n){

  int total;

  total = n % 10; //modulo del numero insertado por 10

  if (n <= 0) {
    return n;
  }//fin IF
    else{
      return total + sumaDigitos(n/10);
    }//fin ELSE

}//fin sumaDigitos
```

Fig. 3

En la imagen posterior, llamada “Fig. 3”, se encuentra la primera (y única) operación de este software que depende de variables de tipo *Enteros* (o *int*). En esta sección, el usuario deberá insertar un valor entero. Aquí, el número será introducido a una operación de tipo *módulo*, en el cual el residuo de una división será archivada y, posteriormente, sumada con los demás dígitos que queden de la operación, estos, se sumarán y el código imprimirá el total de la suma de los números restantes.



```
59  public boolean anidacionCorrecta(String s){
60
61    int A = s.length();
62
63    char C = '(';
64    char D = ')';
65
66    boolean salida = false;
67
68    int mod = A % 2;
69    int split = A/2;
70
71    System.out.println("\nmodulo: "+mod);
72
73    if (mod == 1) {
74      return salida;
75    }
76    else if (mod == 2) {
77      return true;
78      }
79    }//fin operacion booleana
```

Fig. 4

La última operación encontrada en este sistema, es una operación que detecta si la cantidad de paréntesis abiertos va a la par con la cantidad de paréntesis cerrados. Este funciona a base de una variable de tipo *String* que insertará el usuario y, posteriormente, imprimirá una valor booleana que dirá si los paréntesis si están vinculados correctamente, es decir, que la cantidad de paréntesis abiertos sea igual a la de los paréntesis cerrados. En caso contrario, se imprimirá una declaración FALSE para demostrar la falta de paréntesis.