

Avance

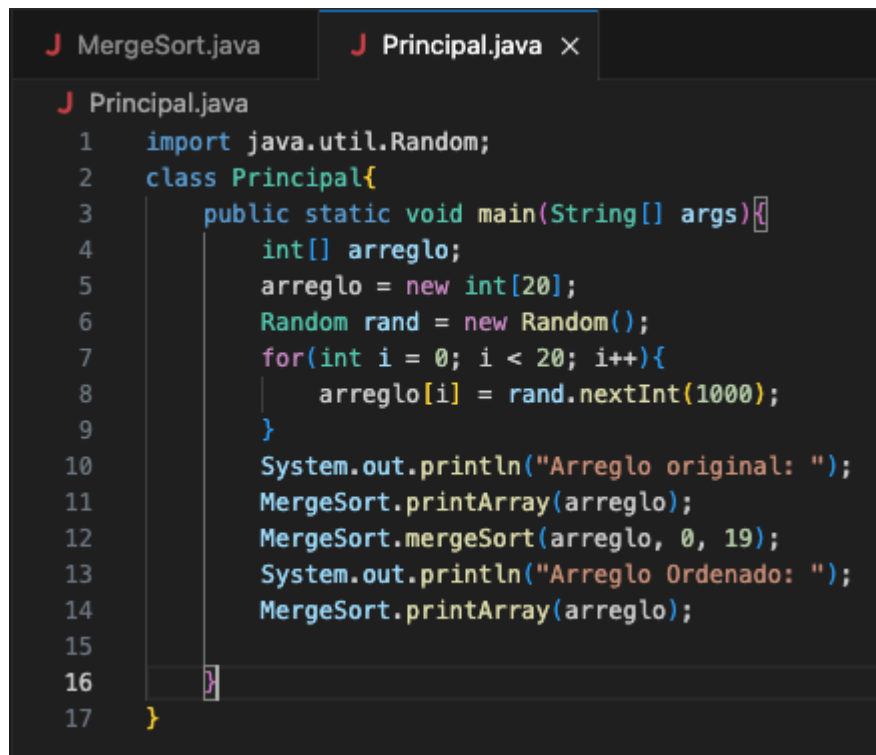
Práctica 3. Algoritmos de ordenamiento parte 3

Objetivo: El estudiante identificará la estructura de los algoritmos de ordenamiento MergeSort, Counting-sort y Radix-sort.

Objetivo de clase: El alumno implementará casos particulares de estos algoritmos para entender mejor su funcionamiento a nivel algorítmico.

Actividades para el Avance de la práctica

1. A diferencia del código visto en clase, el mergeSort proporcionado realiza la operación de merge de manera distinta.
En este nuevo código se utilizan dos listas auxiliares para la combinación, una teniendo los valores a la izquierda de la mitad, y otra con los valores a la derecha de la mitad, posteriormente se comparan los índices entre ambas listas y dependiendo del caso, se realizan cambios en la lista original.



```
J MergeSort.java J Principal.java x
J Principal.java
1 import java.util.Random;
2 class Principal{
3     public static void main(String[] args){
4         int[] arreglo;
5         arreglo = new int[20];
6         Random rand = new Random();
7         for(int i = 0; i < 20; i++){
8             arreglo[i] = rand.nextInt(1000);
9         }
10        System.out.println("Arreglo original: ");
11        MergeSort.printArray(arreglo);
12        MergeSort.mergeSort(arreglo, 0, 19);
13        System.out.println("Arreglo Ordenado: ");
14        MergeSort.printArray(arreglo);
15
16    }
17 }
```

2.

```
estudiante@Kosovo26 Diaz Antunez David Avance P3 % javac Principal.java
estudiante@Kosovo26 Diaz Antunez David Avance P3 % java Principal
Arreglo original:
194 94 250 415 821 199 580 358 364 349 144 861 851 639 165 675 344 968 789 410
Arreglo Ordenado:
94 144 165 194 199 250 344 349 358 364 410 415 580 639 675 789 821 851 861 968
estudiante@Kosovo26 Diaz Antunez David Avance P3 %
```

3. Sirven para la declaración de los arreglos auxiliares, delimitando los elementos a copiar dentro de los arreglos auxiliares.

4. Los arreglos L y R contienen las sublistas a comparar.

```
estudiante@Kosovo26 Diaz Antunez David Avance P3 % javac Principal.java
estudiante@Kosovo26 Diaz Antunez David Avance P3 % java Principal
Arreglo original:
33 16 92 81 46 4 19 53 99 72 95 65 19 64 4 56 57 29 53 67
Subarreglo izquierdo:33
Subarreglo derecho: 16
Subarreglo izquierdo:16 33
Subarreglo derecho: 92
Subarreglo izquierdo:81
Subarreglo derecho: 46
Subarreglo izquierdo:16 33 92
Subarreglo derecho: 46 81
Subarreglo izquierdo:4
Subarreglo derecho: 19
Subarreglo izquierdo:4 19
Subarreglo derecho: 53
Subarreglo izquierdo:99
Subarreglo derecho: 72
Subarreglo izquierdo:4 19 53
Subarreglo derecho: 72 99
Subarreglo izquierdo:16 33 46 81 92
Subarreglo derecho: 4 19 53 72 99
Subarreglo izquierdo:95
Subarreglo derecho: 65
Subarreglo izquierdo:65 95
Subarreglo derecho: 19
Subarreglo izquierdo:64
Subarreglo derecho: 4
Subarreglo izquierdo:19 65 95
Subarreglo derecho: 4 64
Subarreglo izquierdo:56
Subarreglo derecho: 57
Subarreglo izquierdo:56 57
Subarreglo derecho: 29
```

```
Subarreglo izquierdo:16 33 46 81 92
Subarreglo derecho: 4 19 53 72 99
Subarreglo izquierdo:95
Subarreglo derecho: 65
Subarreglo izquierdo:65 95
Subarreglo derecho: 19
Subarreglo izquierdo:64
Subarreglo derecho: 4
Subarreglo izquierdo:19 65 95
Subarreglo derecho: 4 64
Subarreglo izquierdo:56
Subarreglo derecho: 57
Subarreglo izquierdo:56 57
Subarreglo derecho: 29
Subarreglo izquierdo:53
Subarreglo derecho: 67
Subarreglo izquierdo:29 56 57
Subarreglo derecho: 53 67
Subarreglo izquierdo:4 19 64 65 95
Subarreglo derecho: 29 53 56 57 67
Subarreglo izquierdo:4 16 19 33 46 53 72 81 92 99
Subarreglo derecho: 4 19 29 53 56 57 64 65 67 95
Arreglo Ordenado:
4 4 16 19 19 29 33 46 53 53 56 57 64 65 67 72 81 92 95 99
```

5. Si, esto debido a que dentro del funcionamiento de estos métodos no se hace uso de algún atributo propio de la clase, ni se modifica el funcionamiento del método durante la ejecución del programa, la función mergeSort siempre ordenará listas no importa en qué punto del código estemos.

6. Considero que cada iteración se cumple cada vez que se combinan dos sublistas ya particionadas en la función merge.

```
2 3 12 14 38 50 50 58 59 61 63 68 71 73 77 79 88 92 94 98
estudiante@Kosovo26 Diaz Antunez David Avance P3 % javac Principal.java
estudiante@Kosovo26 Diaz Antunez David Avance P3 % java Principal
Arreglo original:
1 64 98 94 94 5 8 64 1 66 73 72 12 88 73 62 82 10 83 99
Iteracion
Subarreglo izquierdo: 1
Subarreglo derecho: 64
Iteracion
Subarreglo izquierdo: 1 64
Subarreglo derecho: 98
Iteracion
Subarreglo izquierdo: 94
Subarreglo derecho: 94
Iteracion
Subarreglo izquierdo: 1 64 98
Subarreglo derecho: 94 94
Iteracion
Subarreglo izquierdo: 5
Subarreglo derecho: 8
Iteracion
Subarreglo izquierdo: 5 8
Subarreglo derecho: 64
Iteracion
Subarreglo izquierdo: 1
Subarreglo derecho: 66
Iteracion
Subarreglo izquierdo: 5 8 64
Subarreglo derecho: 1 66
Iteracion
Subarreglo izquierdo: 1 64 94 94 98
Subarreglo derecho: 1 5 8 64 66
Iteracion
Subarreglo izquierdo: 73
```

```
Subarreglo derecho: 1 5 8 64 66
Iteracion
Subarreglo izquierdo: 73
Subarreglo derecho: 72
Iteracion
Subarreglo izquierdo: 72 73
Subarreglo derecho: 12
Iteracion
Subarreglo izquierdo: 88
Subarreglo derecho: 73
Iteracion
Subarreglo izquierdo: 12 72 73
Subarreglo derecho: 73 88
Iteracion
Subarreglo izquierdo: 62
Subarreglo derecho: 82
Iteracion
Subarreglo izquierdo: 62 82
Subarreglo derecho: 10
Iteracion
Subarreglo izquierdo: 83
Subarreglo derecho: 99
Iteracion
Subarreglo izquierdo: 10 62 82
Subarreglo derecho: 83 99
Iteracion
Subarreglo izquierdo: 12 72 73 73 88
Subarreglo derecho: 10 62 82 83 99
Iteracion
Subarreglo izquierdo: 1 1 5 8 64 64 66 94 94 98
Subarreglo derecho: 10 12 62 72 73 73 82 83 88 99
Arreglo Ordenado:
1 1 5 8 10 12 62 64 64 66 72 73 73 82 83 88 94 94 98 99
```

7.

```
public static void main(String[] args){  
    int[] arreglo;  
    arreglo = new int[20];  
    Random rand = new Random();  
    for(int i = 0; i < 20; i++){  
        arreglo[i] = rand.nextInt(100) + 1;  
    }  
}
```

Ejercicio 1