```
public class App {
             public static void main (String [] args) throws Exception{
                System.out.println(x: "Hello , Word!, metodo de ordenamiento por insercion" );
                 int [] arr ={ 12, 11, 13, 5, 6};
                 System.out.println(x:"Arreglo original : ");
                   r (int num : arr){
                    System.out.println(num+ " ");
                 System.out.println();
                 System.out.println(x:"========");
                 insertionSort(arr); // Llamada al metodo de ordenamiento por insersion
                 System.out.println(x:"Arreglo Ordenado");
                 for (int num : arr){
  PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
  6
  PS C:\Users\s j\OneDrive\Escritorio\Programa_Insert>
📎 0 🛕 0 🔌 0 😓 🖒 Java: Ready
          int n =arr.length;
          for (int i = 1 ; i<n; i++){
             int key = arr [i]; //Selecionamos el elmento a insertaar en su posicion correcta
             while (j >= 0 && arr[j] > key){
    arr [j+1] = arr[j];
                 j--;
               arr [j + 1] = key; // Insertamos key en su posicion correcta
```

El algoritmo de ordenamiento insertsort es especialmente eficiente cuando se trata de listas pequeñas o casi ordenadas. Su enfoque simple y su capacidad para ordenar elementos a medida que avanza a través de la lista lo hacen ideal para aplicaciones en las que se manejan conjuntos de datos más pequeños. Sin embargo, a medida que el tamaño de la lista aumenta, el tiempo de ejecución de insertsort también tiende a aumentar, lo que lo hace menos eficiente que otros algoritmos como mergesort en esos casos.

En resumen, insertsort es una excelente opción para listas pequeñas o casi ordenadas debido a su simplicidad y eficiencia en esos contextos específicos.

Abel Melquisedec Bermúdez Díaz

Número de carnet:2024-1669U