

Taller 3

Fecha: Septiembre de 2024

Indicador de logro a medir: Usar los conceptos de la lógica de la programación, del paradigma orientado a objetos y los lenguajes de programación para solucionar problemas utilizando una metodología y herramienta apropiada.

NOTAS:

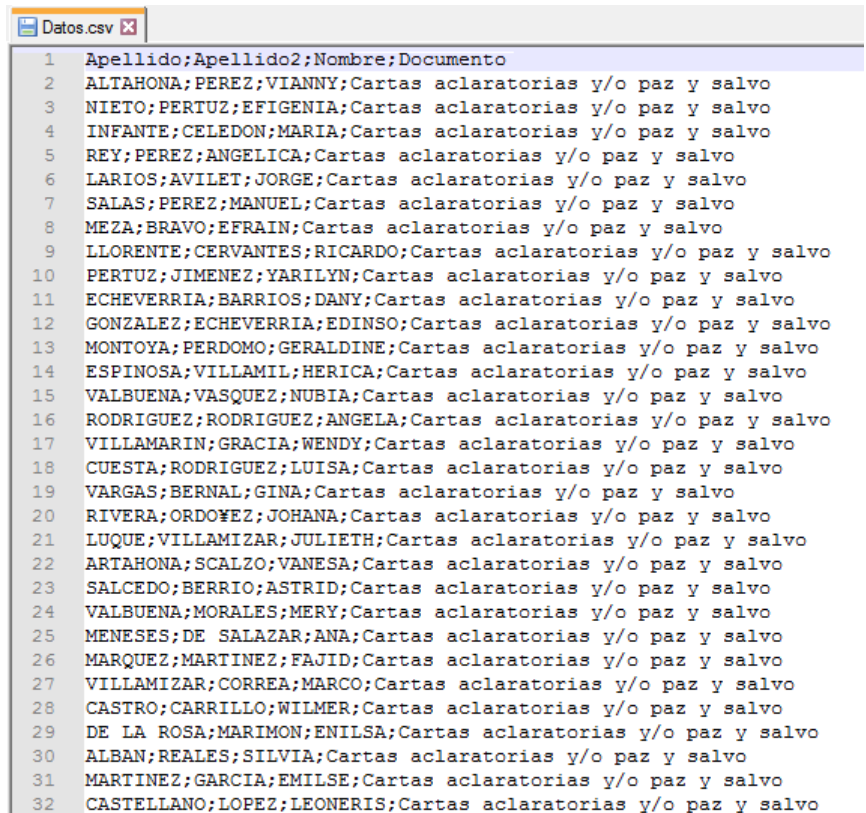
- Este taller se debe hacer como preparación para el quiz. En ningún caso representará una calificación.
- Se entregan ejercicios resueltos como ejemplo para el desarrollo de los demás.

Elaborar una aplicación en un lenguaje orientado a objetos para los siguientes enunciados:

1. Una entidad bancaria requiere un aplicativo que le permita ordenar un listado muy grande de datos

La información está almacenada en 1 archivo plano y contiene:

- Apellidos y Nombre del cliente
- El tipo de documento suministrado



```
1 Apellido;Apellido2;Nombre;Documento
2 ALTAHONA;PEREZ;VIANNY;Cartas aclaratorias y/o paz y salvo
3 NIETO;PERTUZ;EFIGENIA;Cartas aclaratorias y/o paz y salvo
4 INFANTE;CELEDON;MARIA;Cartas aclaratorias y/o paz y salvo
5 REY;PEREZ;ANGELICA;Cartas aclaratorias y/o paz y salvo
6 LARIOS;AVILET;JORGE;Cartas aclaratorias y/o paz y salvo
7 SALAS;PEREZ;MANUEL;Cartas aclaratorias y/o paz y salvo
8 MEZA;BRAVO;EFRAIN;Cartas aclaratorias y/o paz y salvo
9 LLORENTE;CERVANTES;RICARDO;Cartas aclaratorias y/o paz y salvo
10 PERTUZ;JIMENEZ;YARILYN;Cartas aclaratorias y/o paz y salvo
11 ECHEVERRIA;BARRIOS;DANY;Cartas aclaratorias y/o paz y salvo
12 GONZALEZ;ECHEVERRIA;EDINSO;Cartas aclaratorias y/o paz y salvo
13 MONTOYA;PERDOMO;GERALDINE;Cartas aclaratorias y/o paz y salvo
14 ESPINOSA;VILLAMIL;HERICA;Cartas aclaratorias y/o paz y salvo
15 VALBUENA;VASQUEZ;NUBIA;Cartas aclaratorias y/o paz y salvo
16 RODRIGUEZ;RODRIGUEZ;ANGELA;Cartas aclaratorias y/o paz y salvo
17 VILLAMARIN;GRACIA;WENDY;Cartas aclaratorias y/o paz y salvo
18 CUESTA;RODRIGUEZ;LUISA;Cartas aclaratorias y/o paz y salvo
19 VARGAS;BERNAL;GINA;Cartas aclaratorias y/o paz y salvo
20 RIVERA;ORDOÑEZ;JOHANA;Cartas aclaratorias y/o paz y salvo
21 LUQUE;VILLAMIZAR;JULIETH;Cartas aclaratorias y/o paz y salvo
22 ARTAHONA;SCALZO;VANESA;Cartas aclaratorias y/o paz y salvo
23 SALCEDO;BERRIO;ASTRID;Cartas aclaratorias y/o paz y salvo
24 VALBUENA;MORALES;MERY;Cartas aclaratorias y/o paz y salvo
25 MENESES;DE SALAZAR;ANA;Cartas aclaratorias y/o paz y salvo
26 MARQUEZ;MARTINEZ;FAJID;Cartas aclaratorias y/o paz y salvo
27 VILLAMIZAR;CORREA;MARCO;Cartas aclaratorias y/o paz y salvo
28 CASTRO;CARRILLO;WILMER;Cartas aclaratorias y/o paz y salvo
29 DE LA ROSA;MARIMON;ENILSA;Cartas aclaratorias y/o paz y salvo
30 ALBAN;REALES;SILVIA;Cartas aclaratorias y/o paz y salvo
31 MARTINEZ;GARCIA;EMILSE;Cartas aclaratorias y/o paz y salvo
32 CASTELLANO;LOPEZ;LEONERIS;Cartas aclaratorias y/o paz y salvo
```

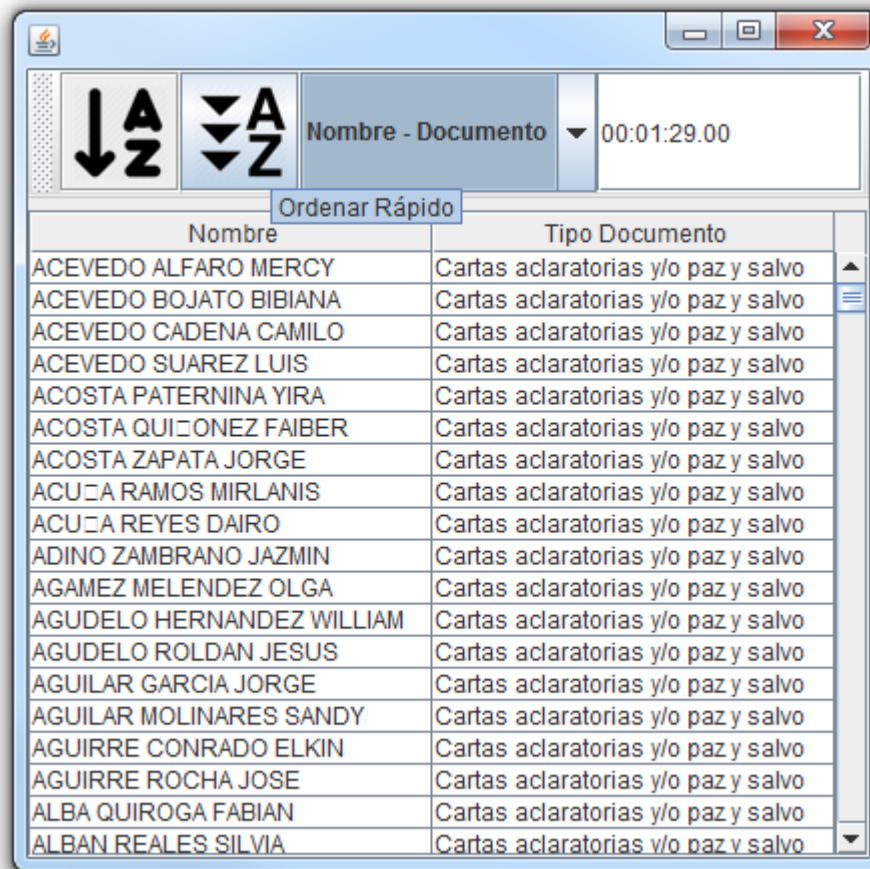
El aplicativo debe permitir ordenar la lista utilizando dos algoritmos de ordenamiento:

- Un algoritmo no recursivo como lo es el **Método de la Burbuja**
- Un algoritmo recursivo como lo es el **Método de ordenamiento Rápido (Quicksort)**

Se deben permitir dos secuencias de ordenamiento para que el usuario elija:

- Primero el nombre completo y luego el tipo de documento
- Primero el tipo de documento y luego el nombre completo

Se debe mostrar el tiempo que demora la ejecución del ordenamiento para comparar cuál algoritmo es más eficiente



Nombre	Tipo Documento
ACEVEDO ALFARO MERCY	Cartas aclaratorias y/o paz y salvo
ACEVEDO BOJATO BIBIANA	Cartas aclaratorias y/o paz y salvo
ACEVEDO CADENA CAMILO	Cartas aclaratorias y/o paz y salvo
ACEVEDO SUAREZ LUIS	Cartas aclaratorias y/o paz y salvo
ACOSTA PATERNINA YIRA	Cartas aclaratorias y/o paz y salvo
ACOSTA QUIÑONEZ FAIBER	Cartas aclaratorias y/o paz y salvo
ACOSTA ZAPATA JORGE	Cartas aclaratorias y/o paz y salvo
ACUÑA RAMOS MIRLANIS	Cartas aclaratorias y/o paz y salvo
ACUÑA REYES DAIRO	Cartas aclaratorias y/o paz y salvo
ADINO ZAMBRANO JAZMIN	Cartas aclaratorias y/o paz y salvo
AGAMEZ MELENDEZ OLGA	Cartas aclaratorias y/o paz y salvo
AGUDELO HERNANDEZ WILLIAM	Cartas aclaratorias y/o paz y salvo
AGUDELO ROLDAN JESUS	Cartas aclaratorias y/o paz y salvo
AGUILAR GARCIA JORGE	Cartas aclaratorias y/o paz y salvo
AGUILAR MOLINARES SANDY	Cartas aclaratorias y/o paz y salvo
AGUIRRE CONRADO ELKIN	Cartas aclaratorias y/o paz y salvo
AGUIRRE ROCHA JOSE	Cartas aclaratorias y/o paz y salvo
ALBA QUIROGA FABIAN	Cartas aclaratorias y/o paz y salvo
ALBAN REALES SILVIA	Cartas aclaratorias y/o paz y salvo

R/

Para comprender este diagrama y el ejercicio, es importante tener en cuenta los siguientes fundamentos:

Recursividad

Recursión es, en ciencias de la computación, una forma de atajar y solventar problemas. De hecho, recursión es una de las ideas centrales de las ciencias computacionales. Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema

Un ejemplo de recursividad es la forma como está hecha una matrioshka (Muñeca Rusa).

Inicialmente se ve una sola muñeca en madera que se puede destapar. Al hacerlo, en su interior hay una copia exacta de la muñeca de menor tamaño. Esta también se puede seguir destapando, hasta llegar a la de menor tamaño, que ya no es destapable.

Se comienza con un matrioshka grande, y vamos destapando matrioshkas más y más pequeñas, hasta que vemos una que es tan pequeña que no ya no puede contener a otra.



Así como una matrioshka contiene a una más pequeña dentro de ella, que tiene a su vez otra aún más pequeña dentro de ella, hasta llegar a una matrioshka tan pequeña que ya no puede contener otra, se van a diseñar algoritmos para resolver problemas de modo que se resuelva una instancia más pequeña del mismo problema, a menos que el problema sea tan pequeño que no se pueda resolver directamente. Esta técnica se denomina **recursividad**.

Los siguientes son ejemplos de algoritmos recursivos, con su respectiva versión iterativa:

a. Algoritmo que calcule el Máximo Común Divisor (MCD) de dos números

Para hallar el MCD de 2 números enteros existe el algoritmo de Euclides el cual se basa en la operación residuo de la división entera. Este consiste en obtener el residuo entre los 2 números enteros. Si este es 0, indica que el divisor es el MCD, de lo contrario, el divisor se hace dividendo y el residuo se hace divisor y se calcula de nuevo el residuo. Estas instrucciones se repiten hasta que el residuo se haga 0 y el último divisor será el MCD. Ejemplo:

Dividendo	Divisor	Residuo
90	35	20
35	20	15
20	15	5
15	5	0

Los algoritmos serían:

Recursivo

```
int calcularMCD(int n1, int n2) {
    if (n2 == 0) {
        return n1;
    } else {
        return calcularMCD(n2, n1 %
n2);
    }
}
```

Iterativo

```
int calcularMCD(int n1, int n2)
{
    int r = n1 % n2;
    while (r > 0) {
        n1 = n2;
        n2 = r;
        r = n1 % n2;
    }
    return n2;
}
```

En el algoritmo recursivo puede observarse:

- Un llamado a la misma función, pero con cambios en los valores de los parámetros. En este caso cuando se llama de nuevo la función, el nuevo divisor es el residuo de los dos números
- Existe una condición donde no se llama de nuevo la función. Este es el punto de detenimiento del algoritmo recursivo. En este caso es cuando el divisor es cero.

b. Algoritmo que permita sumar los elementos de un vector:

Recursivo

```
double sumatoria(double v[], int n) {
    if (n == 0) {
        return v[n];
    } else {
        return sumatoria(v, n - 1) +
v[n];
    }
}
```

Iterativo

```
double sumatoria(double v[]) {
    double suma = 0;
    for (double d : v) {
        suma += d;
    }
    return suma;
}
```

En el algoritmo recursivo se acumula el valor del elemento en la posición que se pasa como parámetro de entrada hasta que se llega al primer elemento.

c. Algoritmo que permita sumar los dígitos de un número:

Recursivo

```
int sumarDigitos(int n) {
    if (n == 0) { //caso base
        return n;
    } else {
        return sumarDigitos(n / 10) +
(n % 10);
    }
}
```

Iterativo

```
int sumarDigitos(int n) {
    int suma = 0;
    while (n > 0) {
        suma += n % 10;
        n /= 10;
    }
    return suma;
}
```

En el algoritmo recursivo se acumula el residuo de dividir por 10 el número que se pasa como parámetro de entrada, el cual se obtiene dividiendo sucesivamente por 10.

Ordenamiento

En ciencias de la computación, un algoritmo de ordenamiento es aquel que coloca los elementos de una lista o vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser una permutación - o reordenamiento - de la entrada que satisfaga la relación de orden dada.

Existen varias clasificaciones de ordenamientos de acuerdo a su estrategia:

- **Algoritmos de inserción.** En este tipo de algoritmo los elementos que van a ser ordenados son considerados uno a la vez. Cada elemento es “insertado” en la posición apropiada con respecto al resto de los elementos ya ordenados. Entre estos algoritmos se encuentran el de *Inserción Directa*, *ShellSort*, *Inserción Binaria* y *Hashing*.
- **Algoritmos de intercambio.** En este tipo de algoritmos se toman los elementos de dos en dos, se comparan y se “intercambian” si no están en el orden adecuado. Este proceso se repite hasta que se ha analizado todo el conjunto de elementos y ya no hay intercambios. Entre estos algoritmos se encuentran el *Burbuja* (*BubbleSort*) y *QuickSort* (*Rápido*).
- **Algoritmos de selección.** En este tipo de algoritmos se “selecciona” o se busca el elemento más pequeño (o más grande) de todo el conjunto de elementos y se coloca en su posición adecuada. Este proceso se repite para el resto de los elementos hasta que todos son analizados. Entre estos algoritmos se encuentra el de *Selección Directa*.
- **Algoritmos de enumeración.** En este tipo de algoritmos cada elemento es comparado contra los demás. En la comparación se cuenta cuántos elementos son más pequeños que el elemento que se está analizando, generando así una “enumeración”. El número generado para cada elemento indicará su posición.

Otra clasificación de los métodos de ordenamiento, los divide en simples y complejos:

- **Los métodos simples:** *Inserción* (o por inserción directa), *Selección*, *Burbuja* y *ShellSort*, en dónde el último es una extensión al método de inserción, siendo más rápido
- **Los métodos complejos:** el *QuickSort* (ordenación rápida) y el *HeapSort*

Consideraremos aquí dos ejemplos clásicos que nos permiten comprender tanto su rapidez como complejidad.

El **algoritmo de la burbuja** es uno de los métodos de ordenación más conocidos y uno de los primeros que aprenden los programadores.

Consiste en comparar cada posición de un vector con las restantes, y si están desordenados, intercambiarlos hasta que estén todos ordenados.

Esto plantea dos recorridos:

- Un recorrido principal entre la primera y penúltima posición (para que pueda tener con quien compararse)
- Unos recorridos secundarios por cada iteración del anterior, iniciando en la posición siguiente hasta la última.

Veamos un ejemplo:

Vector original

50	26	7	9	15	27
----	----	---	---	----	----

Primera pasada:

26	50	7	9	15	27
----	----	---	---	----	----

Se intercambia el 50 y el 26

7	50	26	9	15	27
---	----	----	---	----	----

Se intercambia el 26 y el 7

7	50	26	9	15	27
---	----	----	---	----	----

7	50	26	9	15	27
---	----	----	---	----	----

7	50	26	9	15	27
---	----	----	---	----	----

Segunda pasada:

7	26	50	9	15	27
---	----	----	---	----	----

Se intercambia el 50 y el 26

7	9	50	26	15	27
---	---	----	----	----	----

Se intercambia el 26 y el 9

7	9	50	26	15	27
---	---	----	----	----	----

7	9	50	26	15	27
---	---	----	----	----	----

Tercera pasada:

7	9	26	50	15	27
---	---	----	----	----	----

Se intercambia el 50 y el 26

7	9	15	50	26	27
---	---	----	----	----	----

Se intercambia el 26 y el 15

7	9	15	50	26	27
---	---	----	----	----	----

Cuarta pasada:

7	9	15	26	50	27
---	---	----	----	----	----

Se intercambia el 50 y el 26

7	9	15	26	50	27
---	---	----	----	----	----

Quinta pasada:

7	9	15	26	27	50
---	---	----	----	----	----

Se intercambia el 50 y el 27

El **algoritmo de ordenamiento rápido** (*quicksort* en inglés) es un algoritmo de ordenación creado por el científico británico en computación C. A. R. Hoare.

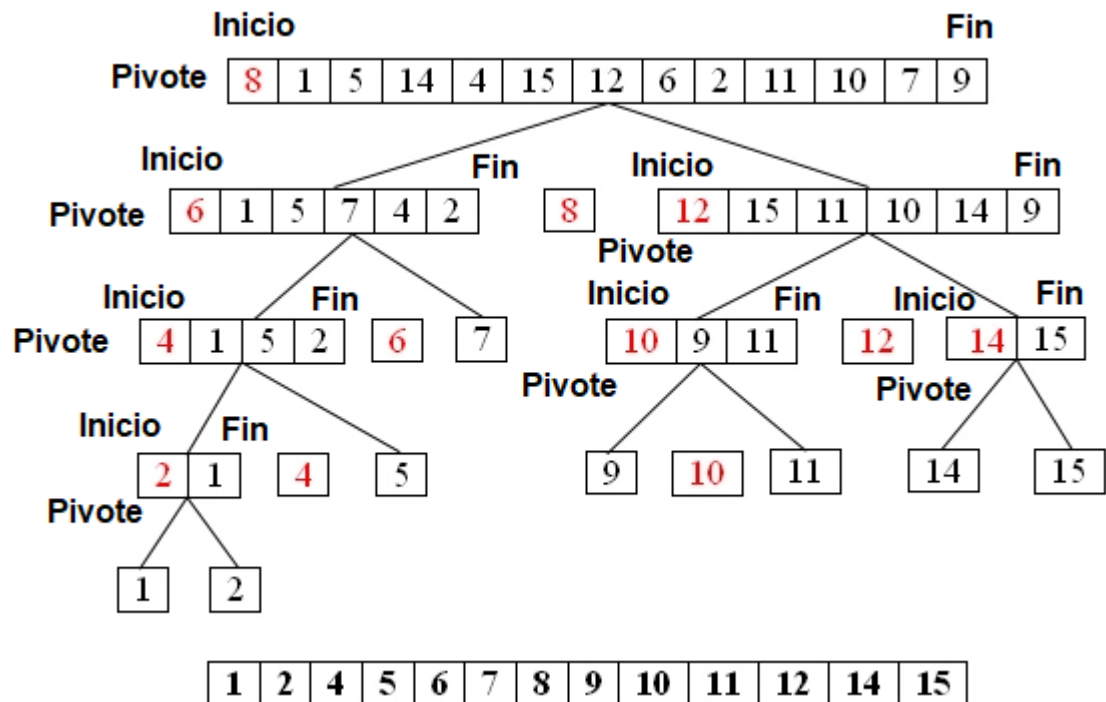
El algoritmo trabaja de la siguiente forma:

- Elegir un elemento del conjunto de elementos a ordenar, al que se llamará pivote
- Reubicar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los

elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada

- La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha
- Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados

La siguiente gráfica ilustra su funcionamiento, asumiendo como pivote la primera posición:



Comprendidos los temas de recursividad y ordenamiento, se procederá a programar la siguiente clase que contendrá tanto la definición de la estructura de almacenamiento de la información (mediante variables y métodos de clase), como la funcionalidad para operar con ella (mediante variables y métodos estáticos):

Inicialmente la clase tendrá las siguientes propiedades:

Documento
nombre: String apellido1: String apellido2: String tipoDocumento: String
getNombreCompleto(): String getTipoDocumento(): String

- *nombre*: El nombre del contacto
- *apellido1*: El número de teléfono fijo del contacto
- *apellido2*: El número de teléfono móvil del contacto
- *tipoDocumento*: El dato del correo del contacto

Y los siguientes métodos (además del método constructor):

- *getNombreCompleto()*: El cual permite obtener una cadena de texto concatenando todos los datos del nombre almacenados
- *getTipoDocumento()*: El cual devuelve el tipo del documento almacenado

El código sería el siguiente:

```
public class Documento {  
  
    private String nombre;  
    private String apellido1;  
    private String apellido2;  
    private String tipoDocumento;  
  
    public Documento() {  
        nombre = "";  
        apellido1 = "";  
        apellido2 = "";  
        tipoDocumento = "";  
    }  
  
    public Documento(String apellido1,  
        String apellido2,  
        String nombre,  
        String tipoDocumento) {  
        this.nombre = nombre;  
        this.apellido1 = apellido1;  
        this.apellido2 = apellido2;  
        this.tipoDocumento = tipoDocumento;  
    }  
  
    public String getNombreCompleto() {  
        return apellido1 + " " + apellido2 + " " + nombre;  
    }  
  
    public String getTipoDocumento() {  
        return tipoDocumento;  
    }  
}
```

Ahora se agregará a esta clase los siguientes atributos y métodos estáticos:

El atributo estático será:

Documento
#documentos: Documento[]
#agregar(): void
#obtenerLista(): void
#mostrarLista(): void
- #intercambiar(): void
- #esMayor(): boolean
#ordenarBurbuja(): String
#ordenarRapido(): String
- #_ordenarRapido(): void
- #ubicarPivote: int

- *documentos*: Es un vector que almacenará la lista de objetos tipo *Documento* que van a ser procesados

Y los métodos estáticos:

- *agregar()*: Permite adicionar a la lista un nuevo objeto *Documento*
- *obtenerLista()*: Se encarga de alimentar la lista de documentos a partir de un archivo plano
- *mostrarLista()*: Se encarga de mostrar en la interfaz gráfica la lista de documentos cargados
- *intercambiar()*: Este método permite, dadas 2 posiciones, intercambiar los respectivos objetos *Documento*
- *esMayor()*: Dadas dos posiciones, permite saber si una es mayor que otra de acuerdo al criterio de ordenación
- *ordenarBurbuja()*: Se encarga de ordenar la lista según el criterio de ordenación usando el algoritmo de la burbuja
- *ordenarRapido()*: Se encarga de ordenar la lista según el criterio de ordenación usando el algoritmo rápido
- *_ordenarRapido()*: Es el algoritmo recursivo de ordenamiento rápido
- *ubicarPivote()*: se encarga de ubicar el primer elemento de la lista de tal forma que queden antes los menores valores y después los mayores, según el criterio de ordenamiento

El código sería el siguiente:

```
//***** Atributos y Metodos estaticos *****
//Almacena la lista de documentos
public static Documento[] documentos;

//Método que agrega un documento a la lista
public static void agregar(Documento d) {
    if (documentos == null) {
        documentos = new Documento[1];
    } else {
        documentos = (Documento[]) Util.redimensionar(documentos,
documentos.length + 1);
    }
    documentos[documentos.length - 1] = d;
}

//Método que llena la lista con datos provenientes de un archivo
public static void obtenerLista(String nombreArchivo) {
    try {
        BufferedReader br = Archivo.abrirArchivo(nombreArchivo);
        String linea = br.readLine();
        linea = br.readLine();
        while (linea != null) {
            String[] textos = linea.split(";");
            if (textos.length >= 4) {
                Documento d = new Documento(textos[0],
                    textos[1],
                    textos[2],
                    textos[3]
                );
                agregar(d);
            }
            linea = br.readLine();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
        }
        linea = br.readLine();
    }
    } catch (Exception ex) {
    }
} //ObtenerLista

//Método que despliega en un JTABLE la lista de documentos cargados
public static void mostrarLista(JTable tbl) {
    String[] encabezados = new String[]{"Nombre", "Tipo Documento"};
    String[][] datos = new
String[documentos.length][encabezados.length];
    for (int i = 0; i < documentos.length; i++) {
        datos[i][0] = documentos[i].getNombreCompleto();
        datos[i][1] = documentos[i].getTipoDocumento();
    }
    tbl.setModel(new DefaultTableModel(datos, encabezados));
} //mostrarLista

//Metodo para intercambiar elementos
private static void intercambiar(int origen, int destino) {
    Documento t = documentos[origen];
    documentos[origen] = documentos[destino];
    documentos[destino] = t;
}

//Método para verificar si un documento es mayor que otro segun el
criterio de ordenación
private static boolean esMayor(Documento dI, Documento dJ, int
criterio) {
    if (criterio == 0) {
        //ordenar primero por Nombre Completo y luego por Tipo de
Documento
        return
((dI.getNombreCompleto().compareTo(dJ.getNombreCompleto()) > 0)
||
(dI.getNombreCompleto().compareTo(dJ.getNombreCompleto()) == 0
&&
dI.getTipoDocumento().compareTo(dJ.getTipoDocumento()) > 0));
    } else {
        //ordenar primero por Tipo de Documento y luego por Nombre
Completo
        return
((dI.getTipoDocumento().compareTo(dJ.getTipoDocumento()) > 0)
||
(dI.getTipoDocumento().compareTo(dJ.getTipoDocumento()) == 0
&&
dI.getNombreCompleto().compareTo(dJ.getNombreCompleto()) > 0));
    }
}

//Método que ordena los datos según el algoritmo de la BURBUJA
public static String ordenarBurbuja(int criterio) {
    Util.iniciarCronometro();
    for (int i = 0; i < documentos.length - 1; i++) {
        for (int j = i + 1; j < documentos.length; j++) {
            if (esMayor(documentos[i], documentos[j], criterio)) {
                intercambiar(i, j);
            }
        }
    }
}
```

```
    }
    }
    return Util.obtenerTextoTiempoCronometro();
}

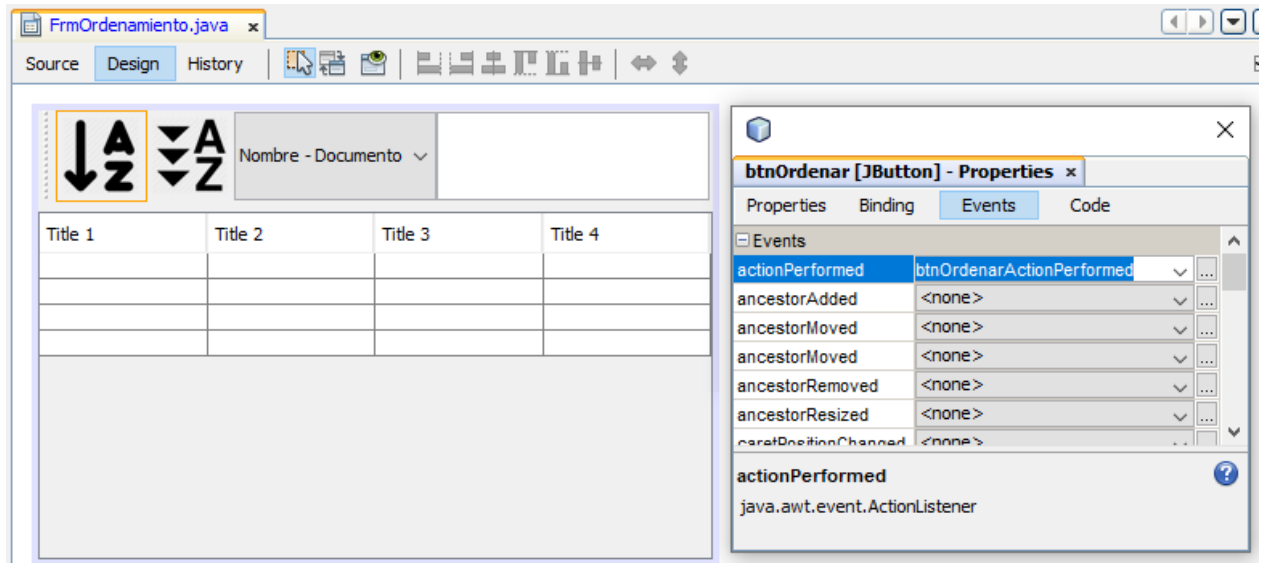
//***** Ordenamiento Rápido
//Método que ordena los datos según el algoritmo RÁPIDO
public static String ordenarRapido(int criterio) {
    Util.iniciarCronometro();
    _ordenarRapido(0, documentos.length - 1, criterio);
    return Util.obtenerTextoTiempoCronometro();
}

//Metodo recursivo de ordenamiento
//los índices inicio y fin indican sobre qué parte de la lista va
operar
private static void _ordenarRapido(int inicio, int fin, int
criterio) {
    //Punto de finalización
    if (inicio >= fin) {
        return;
    }
    //Caso recursivo
    int posicionPivote = ubicarPivote(inicio, fin, criterio);
    _ordenarRapido(inicio, posicionPivote - 1, criterio); //ordenar
los menores
    _ordenarRapido(posicionPivote + 1, fin, criterio); //ordenar los
mayores
}

//Metodo que busca la posición pivote
//donde todos los elementos por encima son mayores
//y todos los elementos por debajo son menores
private static int ubicarPivote(int inicio, int fin, int criterio) {
    Documento dPivote = documentos[inicio];
    int posicionPivote = inicio;
    // Cambia de lugar los elementos
    for (int i = inicio + 1; i < fin + 1; i++) {
        if (esMayor(dPivote, documentos[i], criterio)) {
            posicionPivote++;
            if (i != posicionPivote) {
                intercambiar(i, posicionPivote);
            }
        }
    }
    //Pone el pivote al final de los menores
    if (inicio != posicionPivote) {
        intercambiar(inicio, posicionPivote);
    }
    // Devuelve la posición del pivote
    return posicionPivote;
}
}
```

Ahora bien, para continuar con la codificación, se deben programar los métodos *ordenar()* y *ordenarRapido()* de la interfaz gráfica y que corresponderán a los eventos de los botones de comando agregados al formulario.

Para ordenar usando el algoritmo de la burbuja se hará mediante un método que responde al evento **actionPerformed** del botón de comando **btnOrdenar**:



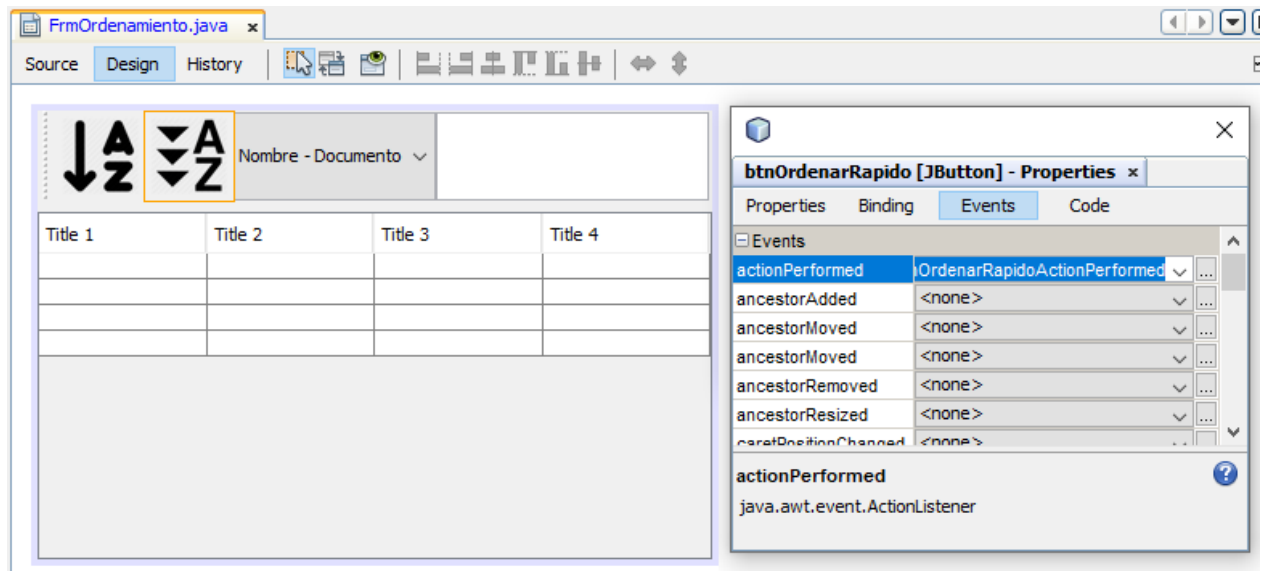
El código respectivo sería el siguiente:

```
private void btnOrdenarActionPerformed(java.awt.event.ActionEvent evt) {
    if (cmbCriterio.getSelectedIndex() >= 0) {

        txtTiempo.setText(Documento.ordenarBurbuja(cmbCriterio.getSelectedIndex(
        )));
        Documento.mostrarLista(tblDatos);
    }
}
```

En este código, se verifica que haya sido elegido el criterio de ordenamiento, luego se llama al método de ordenamiento de la burbuja (mostrándose en una caja de texto el tiempo transcurrido) y por último se muestran en la rejilla, los datos ordenados.

Para ordenar usando el algoritmo rápido, se hará mediante un método que responde al evento **actionPerformed** del botón de comando **btnOrdenarRapido**:



El código respectivo sería el siguiente:

```
private void btnOrdenarRapidoActionPerformed(java.awt.event.ActionEvent evt) {
    if (cmbCriterio.getSelectedIndex() >= 0) {

txtTiempo.setText(Documento.ordenarRapido(cmbCriterio.getSelectedIndex()
));
        Documento.mostrarLista(tblDatos);
    }
}
```

En este código, de manera similar al anterior, se verifica que haya sido elegido el criterio de ordenamiento, luego se llama al método de ordenamiento rápido (mostrándose en una caja de texto el tiempo transcurrido) y por último se muestran en la rejilla, los datos ordenados.

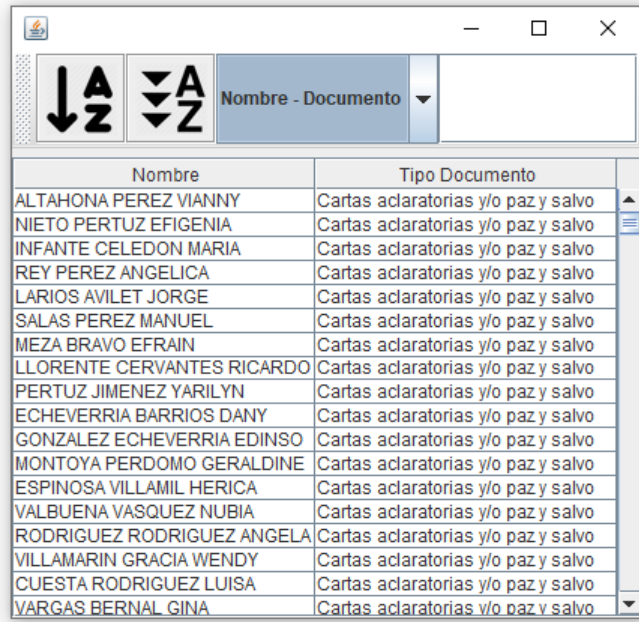
Por último, el código que corresponde el método constructor de la clase y que permite desplegar el estado inicial de la lista a partir de la información almacenada en el archivo plano, es el siguiente:

```
public class FrmOrdenamiento extends javax.swing.JFrame {

    /**
     * Creates new form FrmOrdenamiento
     */
    public FrmOrdenamiento() {
        initComponents();
        String nombreArchivo = System.getProperty("user.dir")
            + "/src/Datos/Datos.csv";
        Documento.obtenerLista(nombreArchivo);
        Documento.mostrarLista(tblDatos);
    }
}
```

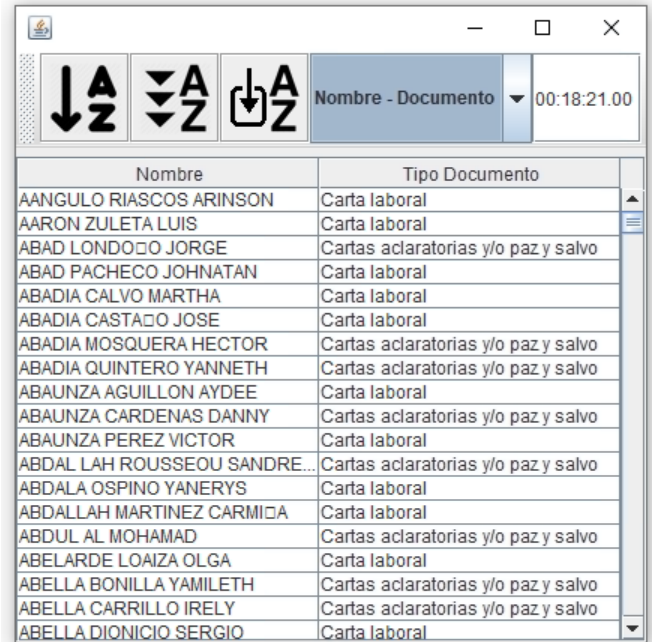
La ejecución de programa luciría así:

- Carga inicial de los datos desordenados



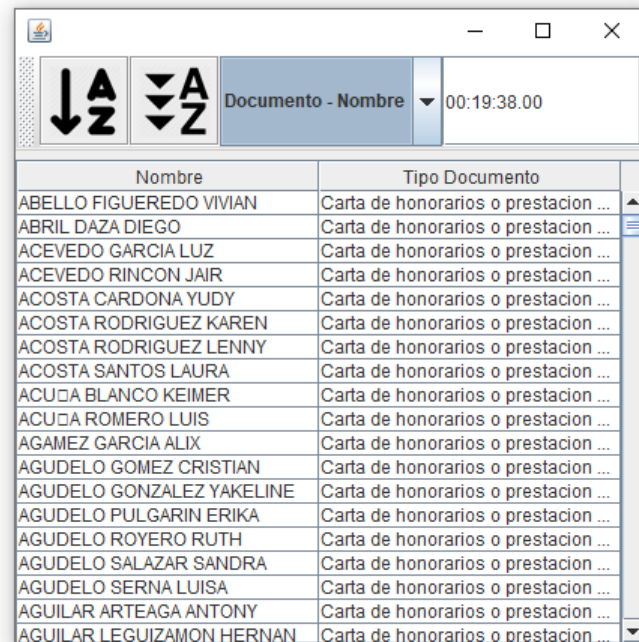
Nombre	Tipo Documento
ALTAHONA PEREZ VIANNY	Cartas aclaratorias y/o paz y salvo
NIETO PERTUZ EFIGENIA	Cartas aclaratorias y/o paz y salvo
INFANTE CELEDON MARIA	Cartas aclaratorias y/o paz y salvo
REY PEREZ ANGELICA	Cartas aclaratorias y/o paz y salvo
LARIOS AVILET JORGE	Cartas aclaratorias y/o paz y salvo
SALAS PEREZ MANUEL	Cartas aclaratorias y/o paz y salvo
MEZA BRAVO EFRAIN	Cartas aclaratorias y/o paz y salvo
LLORENTE CERVANTES RICARDO	Cartas aclaratorias y/o paz y salvo
PERTUZ JIMENEZ YARILYN	Cartas aclaratorias y/o paz y salvo
ECHEVERRIA BARRIOS DANY	Cartas aclaratorias y/o paz y salvo
GONZALEZ ECHEVERRIA EDINSO	Cartas aclaratorias y/o paz y salvo
MONTOYA PERDOMO GERALDINE	Cartas aclaratorias y/o paz y salvo
ESPINOSA VILLAMIL HERICA	Cartas aclaratorias y/o paz y salvo
VALBUENA VASQUEZ NUBIA	Cartas aclaratorias y/o paz y salvo
RODRIGUEZ RODRIGUEZ ANGELA	Cartas aclaratorias y/o paz y salvo
VILLAMARIN GRACIA WENDY	Cartas aclaratorias y/o paz y salvo
CUESTA RODRIGUEZ LUISA	Cartas aclaratorias y/o paz y salvo
VARGAS BERNAL GINA	Cartas aclaratorias y/o paz y salvo

- Ordenando mediante el método de la burbuja usando el primer criterio de ordenamiento



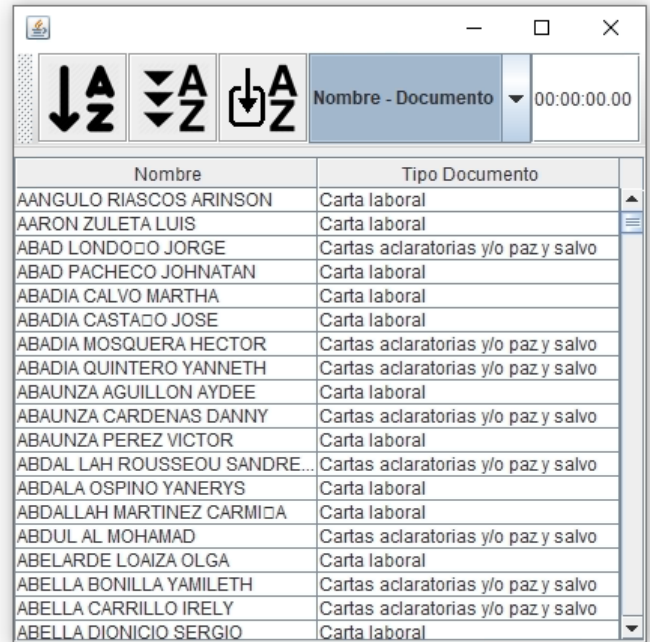
Nombre	Tipo Documento
AANGULO RIASCOS ARINSON	Carta laboral
AARON ZULETA LUIS	Carta laboral
ABAD LONDOÑO JORGE	Cartas aclaratorias y/o paz y salvo
ABAD PACHECO JOHNATAN	Carta laboral
ABADIA CALVO MARTHA	Carta laboral
ABADIA CASTAÑO JOSE	Carta laboral
ABADIA MOSQUERA HECTOR	Cartas aclaratorias y/o paz y salvo
ABADIA QUINTERO YANNETH	Cartas aclaratorias y/o paz y salvo
ABAUNZA AGUILLON AYDEE	Carta laboral
ABAUNZA CARDENAS DANNY	Cartas aclaratorias y/o paz y salvo
ABAUNZA PEREZ VICTOR	Carta laboral
ABDAL LAH ROUSSEOU SANDRE...	Cartas aclaratorias y/o paz y salvo
ABDALA OSPINO YANERYS	Carta laboral
ABDALLAH MARTINEZ CARMIDA	Carta laboral
ABDUL AL MOHAMAD	Cartas aclaratorias y/o paz y salvo
ABELARDE LOAIZA OLGA	Carta laboral
ABELLA BONILLA YAMILETH	Cartas aclaratorias y/o paz y salvo
ABELLA CARRILLO IRELY	Cartas aclaratorias y/o paz y salvo
ABELLA DIONICIO SERGIO	Carta laboral

- Ordenando mediante el método de la burbuja usando el segundo criterio de ordenamiento



Nombre	Tipo Documento
ABELLO FIGUEREDO VIVIAN	Carta de honorarios o prestacion ...
ABRIL DAZA DIEGO	Carta de honorarios o prestacion ...
ACEVEDO GARCIA LUZ	Carta de honorarios o prestacion ...
ACEVEDO RINCON JAIR	Carta de honorarios o prestacion ...
ACOSTA CARDONA YUDY	Carta de honorarios o prestacion ...
ACOSTA RODRIGUEZ KAREN	Carta de honorarios o prestacion ...
ACOSTA RODRIGUEZ LENNY	Carta de honorarios o prestacion ...
ACOSTA SANTOS LAURA	Carta de honorarios o prestacion ...
ACUÑA BLANCO KEIMER	Carta de honorarios o prestacion ...
ACUÑA ROMERO LUIS	Carta de honorarios o prestacion ...
AGAMEZ GARCIA ALIX	Carta de honorarios o prestacion ...
AGUDELO GOMEZ CRISTIAN	Carta de honorarios o prestacion ...
AGUDELO GONZALEZ YAKELINE	Carta de honorarios o prestacion ...
AGUDELO PULGARIN ERIKA	Carta de honorarios o prestacion ...
AGUDELO ROYERO RUTH	Carta de honorarios o prestacion ...
AGUDELO SALAZAR SANDRA	Carta de honorarios o prestacion ...
AGUDELO SERNA LUISA	Carta de honorarios o prestacion ...
AGUILAR ARTEAGA ANTONY	Carta de honorarios o prestacion ...
AGUILAR LEGUIZAMON HERNAN	Carta de honorarios o prestacion ...

- Ordenando mediante el método rápido usando el primer criterio de ordenamiento



Nombre	Tipo Documento
AANGULO RIASCOS ARINSON	Carta laboral
AARON ZULETA LUIS	Carta laboral
ABAD LONDOÑO JORGE	Cartas aclaratorias y/o paz y salvo
ABAD PACHECO JOHNATAN	Carta laboral
ABADIA CALVO MARTHA	Carta laboral
ABADIA CASTAÑO JOSE	Carta laboral
ABADIA MOSQUERA HECTOR	Cartas aclaratorias y/o paz y salvo
ABADIA QUINTERO YANNETH	Cartas aclaratorias y/o paz y salvo
ABAUNZA AGUILLON AYDEE	Carta laboral
ABAUNZA CARDENAS DANNY	Cartas aclaratorias y/o paz y salvo
ABAUNZA PEREZ VICTOR	Carta laboral
ABDAL LAH ROUSSEOU SANDRE...	Cartas aclaratorias y/o paz y salvo
ABDALA OSPINO YANERYS	Carta laboral
ABDALLAH MARTINEZ CARMIDA	Carta laboral
ABDUL AL MOHAMAD	Cartas aclaratorias y/o paz y salvo
ABELARDE LOAIZA OLGA	Carta laboral
ABELLA BONILLA YAMILETH	Cartas aclaratorias y/o paz y salvo
ABELLA CARRILLO IRELY	Cartas aclaratorias y/o paz y salvo
ABELLA DIONICIO SERGIO	Carta laboral

- Ordenando mediante el método rápido usando el segundo criterio de ordenamiento

<div> <div>↓ A Z</div> <div>↕ A Z</div> <div>↑ A Z</div> </div> <div>Documento - Nombre</div> <div>00:00:00.00</div>	
Nombre	Tipo Documento
ABELLO FIGUEREDO VIVIAN	Carta de honorarios o prestacion ...
ABRIL DAZA DIEGO	Carta de honorarios o prestacion ...
ACEVEDO GARCIA LUZ	Carta de honorarios o prestacion ...
ACEVEDO RINCON JAIR	Carta de honorarios o prestacion ...
ACOSTA CARDONA YUDY	Carta de honorarios o prestacion ...
ACOSTA RODRIGUEZ KAREN	Carta de honorarios o prestacion ...
ACOSTA RODRIGUEZ LENNY	Carta de honorarios o prestacion ...
ACOSTA SANTOS LAURA	Carta de honorarios o prestacion ...
ACUÑA BLANCO KEIMER	Carta de honorarios o prestacion ...
ACUÑA ROMERO LUIS	Carta de honorarios o prestacion ...
AGAMEZ GARCIA ALIX	Carta de honorarios o prestacion ...
AGUDELO GOMEZ CRISTIAN	Carta de honorarios o prestacion ...
AGUDELO GONZALEZ YAKELINE	Carta de honorarios o prestacion ...
AGUDELO PULGARIN ERIKA	Carta de honorarios o prestacion ...
AGUDELO ROYERO RUTH	Carta de honorarios o prestacion ...
AGUDELO SALAZAR SANDRA	Carta de honorarios o prestacion ...
AGUDELO SERNA LUISA	Carta de honorarios o prestacion ...
AGUILAR ARTEAGA ANTONY	Carta de honorarios o prestacion ...
AGUILAR LEGUIZAMON HERNAN	Carta de honorarios o prestacion ...

2. Implementar los algoritmos de Inserción, Selección y Mezcla en el anterior ejercicio