El proyecto creado para esta tarea le he llamado “Tarea6” dentro del mismo he creado 3 clases MenuGestion, DatosCliente, MetodosFicheros, y los he metido en un paquete llamado FicherosEmpresa.

La clase principal que contiene el método main “MenuGestion”, nos muestra por pantalla el menú con las distintas opciones de la gestión de clientes: añadir, listar, buscar, borrar, borrar fichero y salir del programa. Una vez recogida y comprobada que la selección por teclado sea la correcta, evalúa la opción por medio de un switch-case y llama al método pertinente creado en la clase “MetodosFicheros”:

```
switch (op) {
     case 1:
         MetodosFicheros.añadirCliente();
         break;
     case 2:
         MetodosFicheros.listarClientes();
         break;
     case 3:
         MetodosFicheros.buscarCliente();
         break;
     case 4:
         MetodosFicheros.borrarCliente();
         break;
     case 5:
         MetodosFicheros.borrarFichero();
         break;
     case 6:
         System.out.println("Hasta la próxima!!");
         m = false;
         break;
     default:
         System.out.println("Introduzca una opción válida del 1 al 6: ");}
```

Allí nos encontramos con todos los métodos con los que va a funcionar nuestra aplicación:

- `public static void añadirCliente()`

  Recogemos los datos de cliente por teclado, comprobamos que; dni sea correcto (mediante método `comprobarDni()`), teléfono tenga formato numérico y de 9 dígitos, deuda sea formato numerico y de tipo float y que nombre y dirección sean de tipo String, recogiendo todas las excepciones mediante try catch. Si alguno de estos datos no fuera correcto nos lo volvería a pedir controlado por do-while.

  Una vez tenemos esos datos los guardamos en un ArrayList del tipo objeto “ClienteDatos”, comprobamos si existe el fichero (con llamada a método `comprobarFichero()`) si no existiera lo creamos y a continuación guardamos en él con llamada al método `guardarFichero()`.

- `static boolean comprobarDni(String dni)`

  Recibe como parámetro el número dni introducido por teclado, comprueba que el formato sea de 9 caracteres, a continuación compara mediante el algoritmo de “comprobación de letras del dni”, la letra introducida con una constante que hemos declarado como tipo String (LETRADNI) y nos devolverá un valor “true” si el dni es correcto o un valor “false” si es incorrecto el formato o la letra.

- `private static boolean comprobarFichero()`

  Comprueba que "clientes.dat" existe en nuestro directorio actual (dirección relativa) y devuelve "true" o "false" dependiendo si se cumple o no. Contempla excepciones de entrada-salida mediante try-catch.

- `private static ArrayList<DatosCliente> leerFichero()`

  Primero creamos un array tipo `DatosCliente`, comprobamos que el fichero existe (`comprobarFichero()`), a continuación creamos un flujo de entrada de fichero (clientes.dat), una entrada de flujo del tipo objeto y leemos ese array tipo objeto del flujo introduciéndolo en el array creado al principio. Retornamos dicho array al método.

  Cerramos los flujos y comprobamos excepciones mediante try-catch.

- `private static void guardarFichero(ArrayList<DatosCliente> clienteArray)`

  Creamos un flujo de salida al fichero "clientes.dat", un flujo de salida del tipo objeto y escribimos el arrayList del tipo DatosCliente para que se guarde en fichero.

  Cerramos los flujos y comprobamos excepciones mediante try-catch.

- `public static void listarClientes()`

  Comprueba si el fichero existe y si el arrayList del tipo DatosCliente no está vacío y si es así lo recorre mostrando por pantalla todos los datos de la clase DatosCliente mediante el método imprimir().

- `public static void buscarCliente()`

  Pide al usuario que introduzca dni por teclado, lo comprueba mediante método, sino lo vuelve a pedir. Lee el fichero y lo vuelca en arrayList del tipo objeto DatosCliente, recorre el array y comprobando si el String dni del fichero coincide con el introducido, si fuera así muestra por pantalla todos los datos de ese cliente. Sino mostraría un mensaje de que no existe ese cliente en fichero.

- `public static void borrarCliente()`

  Hace lo mismo que el método `buscarCliente()` con la diferencia que cuando el dni coincide en vez mostrar por pantalla borra los datos del cliente de la posición del arrayList y guarda el nuevo arrayList en el fichero.

- `public static void borrarFichero()`

  Comprueba si el fichero existe mediante método `comprobarFichero()` y si es así lo borra por completo utilizando el método *delete* de la clase File.

Por último tenemos la clase DatosCliente definida de la siguiente manera:

```
public class DatosCliente implements Serializable {

    //Bloque de declaración de atributos de clase
    private float deuda;
    private String dni, nombre, direccion, telefono;
    private static final long serialVersionUID =-5588012102836621537L;

    DatosCliente(String dni, String nombre, String telefono, String direccion,
Float deuda) {
        this.dni = dni;
        this.nombre = nombre;
        this.telefono = telefono;
        this.direccion = direccion;
        this.deuda = deuda;
    }//Fin del método constructor de la clase
```

Hemos implementado la interfaz java.io.Serializable para poder convertir las instancias de este objeto a secuencias de bytes y así poder escribir o leer en el fichero. La clase tiene definidos unos atributos encapsulados del tipo private uno de ellos es la constante "`serialVersionUID`" para solucionar una de las excepciones a la hora de acceder al fichero.

También contiene un constructor con los parámetros requeridos por la aplicación y varios métodos para recogida y muestra de los atributos. Uno de ellos, `public String imprimir()`, nos devuelve la concatenación de todos los atributos de clase en un solo String, lo utilizaremos en varias partes del programa para mostrar todos los datos del cliente requerido por pantalla.