# PRÁCTICA 4

En esta última práctica, tenemos que diseñar un programa que trabaja con varios ficheros (“.txt”, “.csv” y “.bin”). Éstos, contienen información relativa a los alumnos y sus asignaturas del Grado Superior.

El programa Java diseñado va a ser capaz de leer los ficheros, almacenar su información y darle opciones al usuario para visualizar o manipular esta información y sobrescribir los ficheros.

## **FICHEROS y clases A TRATAR:**

* Fichero “.csv”, que contienen el “Nombre”, “Apellido”, y “nip” de los alumnos.

Use utiliza un método para leer el fichero, llamado “leerFicheroAlumnos\_csv”, que almacena cada alumno como un objeto de tipo Alumno que hemos creado. También se ha añadido un atributo de tipo ArrayList de enteros, que contiene los códigos de las asignaturas cursadas.

Clase Alumno: Implementa un constructor que recibe los parámetros del archivo y los Getter y Setter correspondientes.

Además dos métodos para mostrar el atributo ArrayList de asignaturas cursadas, que recorre el ArrayList de las asignaturas del curso y si coinciden los códigos lo muestra por pantalla. De estos métodos, existe uno que ordena por orden del número Nip, y otro que ordena por orden alfabético.

También se han creado dos métodos que añaden o eliminan una asignatura al atributo ArrayList que contiene los códigos de las asignaturas cursadas.

* Fichero “.txt”, que contiene el “Código Asignatura”, “Créditos”, “Cuatrimestre”, “Tipo” y “Nombre”. Se utiliza un método para leer el fichero, llamado “leerFicheroAsignaturas\_txt”, que almacena cada asignatura como un objeto de tipo Asignatura que se ha creado. También se ha añadido un atributo de tipo ArrayList, que contiene los alumnos que cursan la asignatura.

Clase Asignatura: Implementa un constructor que recibe los parámetros del archivo, y los Getter y Setter correspondientes, a su vez dos métodos para mostrar el atributo ArrayList de los alumnos del curso, que es recorrido y comparado con los datos de los alumnos, si coinciden los nip se mostrará por pantalla. De estos dos métodos mencionados, uno ordena por orden alfabético los nombres de las asignaturas, y el otro por el código de la asignatura.

* Fichero “.dat”, que contiene el “Id del alumno”, y su “Código de asignatura”. Utilizamos un método para leer el fichero, llamado “leerFicheroMatriculas\_dat”, que almacena cada matricula como un objeto de tipo Matricula que hemos creado.

Clase Matrícula: Implementa los Getter y Setter y un constructor, pero no tiene ningún método especial.

Para facilitar su explicación, se muestra el siguiente diagrama de flujo:



## **Desarrollo del programa:**

Empieza el programa y lee los ficheros que contienen la información gracias a los métodos creados para ello, y lo almacena en los correspondientes ArrayList de cada clase que se utilizarán a lo largo del programa.

Se han creado dos métodos que sirven para rellenar y actualizar las matrículas en cada asignatura y en cada alumno, por lo que recorre las matriculas que se han leído del fichero “.dat”.

A continuación se le va a pedir al usuario que introduzca órdenes y no finalizará hasta que se indique la orden: "fin";

A continuación se va a explicar detalladamente como se tratan en el programa los casos de uso requeridos:

**1º) Si introduce "Matriculas"**: Recorre las asignaturas y va mostrando su nombre y los alumnos matriculados que se corresponderán con el ArayList.size() de su atributo de los alumnos matriculados.

**2º) Si introduce “Asignaturas”:**

-Si no añade nada más, imprime que falta indicar un NIP. Se acaba el bucle actual.

-Comprueba si el siguiente parámetro corresponde con un NIP que exista, gracias a un método que devolverá su índice en el ArrayList de Alumnos o devolverá -1 en caso de que no se encuentre o haya sido erróneo.

-Seguidamente si no ha introducido más parámetros o ha añadido la letra **"C",** llama al método de la clase alumnos que recorre el atributo del ArrayList de las asignaturas matriculadas y muestra la información de las asignaturas ordenadas por el número del código.

-Si ha introducido **"A"** llamará al método similar pero que ordena por orden alfabético los nombres de las asignaturas que tenga.

- Si ha introducido un tercer parámetro diferente, escribirá que no se encuentra esa orden.

**3º) Si introduce "Alumnos":**

-Si no añade nada más, imprime por pantalla que falta indicar un Código de asignatura. Se acaba el bucle actual.

-Si introduce más de una palabra, comprueba que el siguiente parámetro introducido corresponde con un código de una asignatura que existe. Con el método que devuelve el índice en el ArrayList de asignaturas o devuelve -1 si ha introducido un código erróneo.

-Seguidamente si no ha introducido más parámetros o ha añadido la letra "N", llama al método de la clase asignaturas que recorre el atributo del ArrayList de las alumnos matriculados en ella y muestra el información por pantalla.

-Si ha introducido "A" llamará al método similar el cual lo muestra ordenado en orden alfabético.

-Si ha introducido otro parámetro diferente indicará que no se reconoce ese parámetro.

**4º) Si introduce “Matricular”:**

-Si solo introduce la palabra matricular escribe que falta indicar un NIP y no continua con la ejecución posterior del método.

-Comprueba si el NIP introducido existe y es válido gracias al método que comprueba el NIP comparándolo con los alumnos almacenados en el ArrayList principal. Devuelve el índice en ese ArrayList si lo encuentra y si no devuelve -1.

-Seguidamente si no ha introducido ningún código de matrícula, indicará que falta ese parámetro.

-Por cada elemento introducido posterior al NIP, si este es correcto, analiza si corresponde a un código que exista de asignatura en el método que comprueba las asignaturas almacenadas en el ArrayList de datos asignaturas, devuelve su índice y si no lo encuentra devuelve -1.

-A continuación si es un código válido comprueba si el alumno está ya matriculado de esa asignatura y así no duplicar matriculaciones. Por lo que se usa el método de comprobar matriculaciones, si devuelve -1 significa que no existe ese dato par de NIP + CÓDIGO que corresponde a una matriculación, así que se procede a añadir al ArrayList de las matriculaciones los datos introducidos. Una vez que se ha introducido en ese ArrayList llama al método “añadirAsignatura” de la clase Alumno que introduce ese dato al ArrayList de asignaturas del alumno, y al método “añadirAlumno” que hace los mismo para la clase Asignatura.

**5º) Si introduce “Eliminar”:**

-Si solo introduce la palabra eliminar te indica que falta indicar un nip, y se acaba la ejecución del método.

-Comprueba que el nip introducido se corresponde con un nip que exista, llama al método para ello que devolverá el índice en el ArrayList de ese alumno y si no lo encuentra un -1.

-Si no reconoce más parámetros, indica con una comprobación si está seguro el usuario de eliminar todas las asignaturas del alumno. Si el usuario indica "s" procede a recorrer con un Iterator el Arraylist de matrículas y eliminar cada objeto en el que aparezca su nip. Dejará vacío el atributo de las asignaturas cursadas de ese alumno, y actualizara las asignaturas recorriéndolas y actualizando esos datos.

-Por cada parámetro introducido después del nip, comprueba primero si ese código se corresponde con una asignatura que existe en el programa, gracias al método de comprobar asignatura, en caso de que si exista, pasa a comprobar con otro método que recorre las matriculaciones si esa matricula existe o no, en caso de que no exista, no se podrá eliminar nada, pero si sí existe, pasa a eliminar ese registro del ArrayList de las matriculaciones, llama a un método de la clase Alumno que recorre sus asignaturas matriculadas y cuando encuentra la que corresponde la elimina, y hace lo equivalente con otro método de la clase Asignatura que recorre su ArrayList de alumnos matriculados y elimina el que corresponda con ese NIP.

**6º) Si introduce "Actualizar":**

-Comando extra añadido al programa(no estaba contemplado en la tarea), se ejecuta un método que sobrescribe el fichero binario en el sistema con los datos actuales que se encuentran en el ArrayList de matrículas que contiene los cambios hechos por el usuario. Ya que hasta ahora no se actualizaba el fichero, y solo teníamos los datos correctos en el ArrayList.

**7º)** **ELSE**:

-Si lo que introduce no coincide con nada de lo anterior, indica que el comando es erróneo.

**8º) Si introduce un "fin":**

-Por último,saldrá del bucle while que permite al usuario introducir órdenes constantemente, y actualizará automáticamente el fichero binario con el método explicado en el punto **6º**. Además deja una opción a recibir por pantalla un resumen de los alumnos y como han quedado sus matriculaciones finalmente.