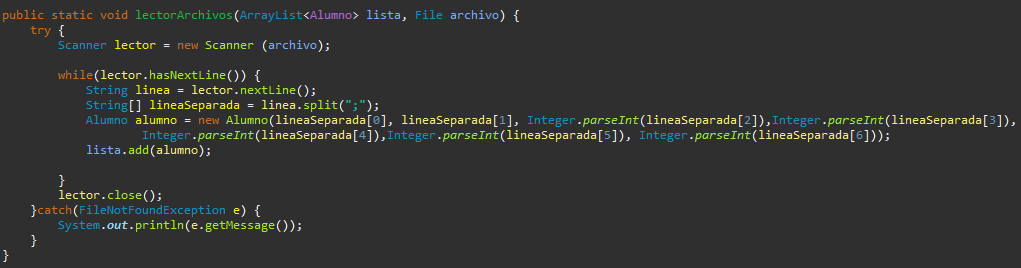
**Operaciones con ArrayList.**

**Joshua Sangareau 1ºDAM.**

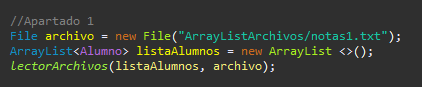
**1. Importar la información contenida en el documento notas1.txt, donde la información tiene el siguiente formato:**

grupo;nombre,matemáticas;lengua;física;química;inglés

He creado un método que recibe un ArrayList de tipo alumno en este caso y un objeto de tipo File, el cual se leerá y se irán creando objetos de tipo alumno en el ArrayList recibido.

****

Se crea el ArrayList, el objeto de tipo File y se llama al método lectorArchivos en el main.



**2. Mostrar el número de registros que contiene.**

Se muestra la cantidad de registros llamando al método size()

****

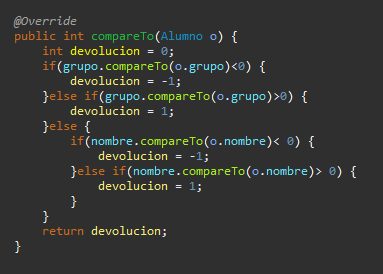
Genera esta salida por consola:



**3. Ordenar los registros por grupo y nombre.**

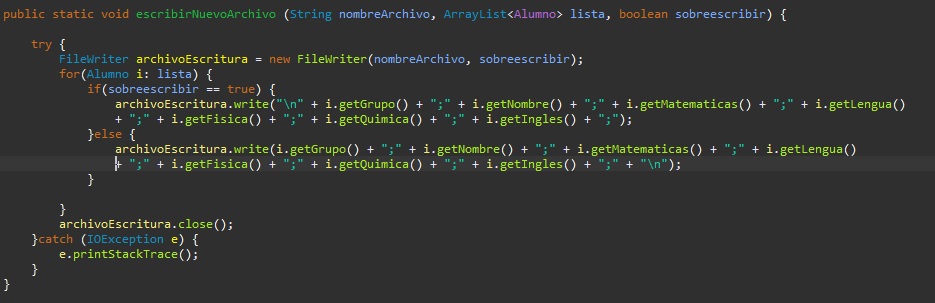
Se implementa el interfaz Comparable en la clase Alumno y se ordena por grupo y nombre:







Se crea método escribirNuevoArchivo para generar un nuevo archivo y poder mostrar la lista ordenada como se solicita. El archivo se llama AlumnosOrdenados y estará dentro de la carpeta del proyecto en una carpeta llamada ArrayListArchivos.



Este método recibe como parámetros la ruta relativa(con el nombre del archivo a crear), el ArrayList de tipo Alumno de donde cogerá los datos y un boolean para saber si se sobreescribe el archivo o se debe añadir la información a un archivo ya existente.

**4. Insertar siguiente registro:**

1º ESO D;López Muro, Rosario;3;3;4;4;3

Se genera un nuevo Alumno y se mete en una variable de tipo Alumno. Tras esto se añade a la lista de esta forma:

****

**5. Insertar el siguiente registro en mitad de la lista:**

1º ESO B;Duarte Durán, Alba;2;3;4;1;3

Se realiza el mismo procedimiento del apartado anterior, lo único que a la hora de introducirlo en la lista le indicamos la posición en la que lo debe insertar. (En mi caso divido entre 2 el tamaño de la lista y restándole 1 para que esté en el medio de esta. Se resta 1 porque las posiciones del ArrayList empiezan en 0)



**6. Mostrar la posición que ocupa el registro anterior insertado.**

Se muestra la información por consola



Genera esta salida:



**7. Insertar los registros del documento notas2.txt**

Se llama al método anteriormente mencionado escribirNuevoArchivo, esta vez con el nombre del archivo al que debe añadir la información, mandandole como parámetro true.

****

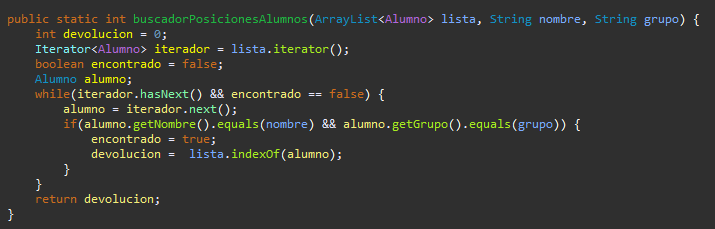
**8. Buscar el siguiente registro:**

1º ESO B;Cordero Benítez, Alejandro;2;3;2;1;4

He creado un método que recibe la lista a recorrer, el nombre y el grupo del alumno a encontrar.

Genero un iterador para recorrer la lista y que se detenga cuando encuentre al alumno con el nombre indicado y el grupo indicado.

Devuelve un int con la posición en la que se encuentra en la lista:



Se llama al método desde el main:



Genera esta salida:



**9. Buscar el siguiente registro.**

1º ESO C;Zoodsma Villalón, Natali;2;3;4;5;3

Se realiza el mismo procedimiento que el apartado anterior:



Genera esta salida:



**10. Borra el registro que está en la posición mitad más 1, indicando la información del registro que borras.**

Se llama al método remove en la lista de alumnos y se le indica la posición del alumno a borrar. En este caso la mitad +1. Yo le pongo la posición resultado de dividir entre 2 el tamaño de la lista(ya que al empezar el la posición 0 el ArrayList ya estaría sumando 1)

****

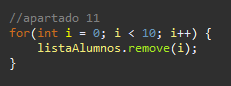
Se muestra la información del alumno borrado:



**11. Borra los primeros 10 registros.**

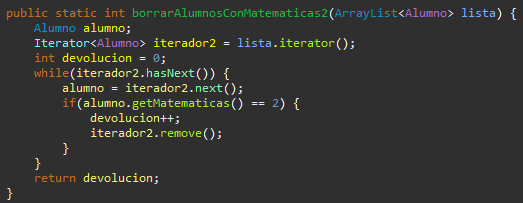
Genero un bucle para recorrer las 10 primeras posiciones y borrarlas.

(Existe un método removeRange() pero está definido como protected y no es visible. No se puede usar)

****

**12. Borra todos los registros que tengan en matemáticas un 2. Indica cuántos registros borras.**

He generado un método borrarAlumnosConMatematicas2 que recibe la lista como parámetro y se borrar los alumnos con un 2 en matemáticas. Se genera un iterador para poder borrar a la vez que se recorre el ArrayList.



Se llama al método en el main y se muestra por pantalla el resultado

****

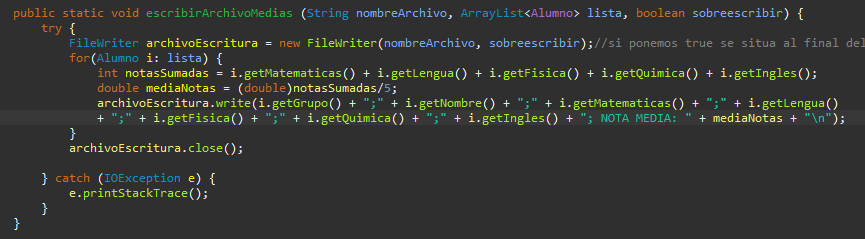
Genera esta salida:



**13. Calcula la nota media para cada registro.**

Se crea un método para calcular las medias y escribirlas en un nuevo archivo.

Es muy parecido al método escribirNuevoArchivo pero este método añadirá las medias de cada registro en el archivo

****

Llamo al método en el main pasándole los parámetros necesarios.

****

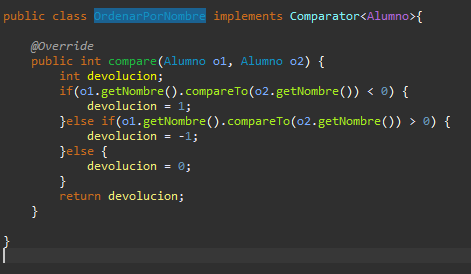
El archivo se llama mediasCadaRegistro y estará dentro de la carpeta del proyecto en una carpeta llamada ArrayListArchivos.

**14. Crea una sublista por cada grupo y ordénala por nombre descendente.**

Se crean 5 ArrayList para poder guardar la información de cada uno de ellos y se recorre la lista para ir añadiendo a cada lista sus alumnos correspondientes

****

Tras esto se ha creado una clase llamada OrdenarPorNombre para poder ordenar por el nombre y se llama al método sort() de la clase Collections indicando la lista que se quiere ordenar y el comparador a usar en este caso OrdenarPorNombre.

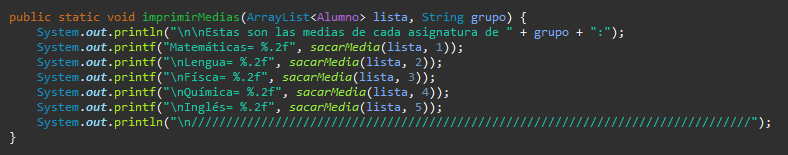


A continuación se llama al método crearNuevoArchivo para generar archivos con el contenido de las listas. Ordenados por nombre y cada grupo en su archivo y lista diferente.

Los archivos se llaman lista1ºA, lista1ºB, lista1ºC, lista1ºD, lista1ºE y estarán dentro de la carpeta del proyecto en una carpeta llamada ArrayListArchivos.

**15. Calcula para cada grupo la nota media de cada materia.**

Se crean 2 métodos ImprimirMedias y sacarMedias. En el primer método se imprime el texto deseado y se llama al segundo método para calcular las medias.

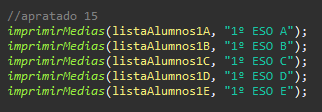


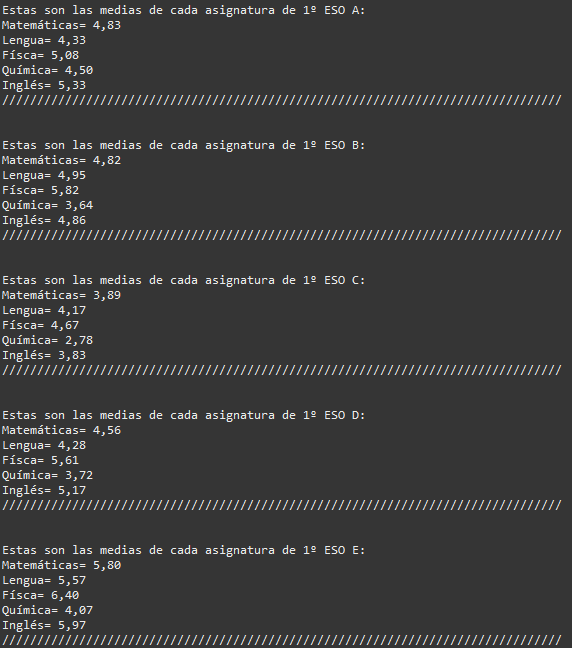
El segundo método recibe como parámetro la lista y las asignaturas. Este método recorrerá la lista para la asignatura pasada por parámetro y hará el cálculo de la media.

Se suman todas las notas y se divide por el tamaño de la lista.



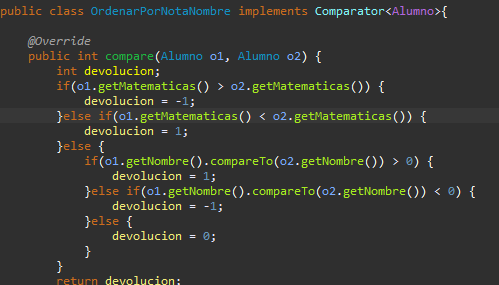
Tras esto se llama al metodo imprimirMedias en el main para cada una de las listas y se muestra por pantalla

****

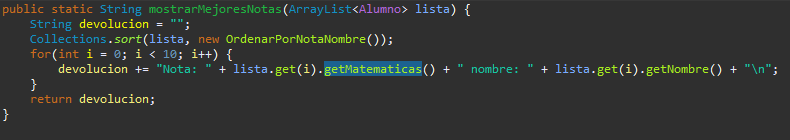
****

**16. Para cada grupo muestra las diez mejores notas en matemáticas ordenadas por nota y nombre.**

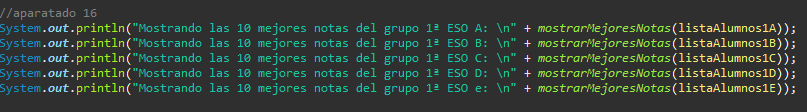
Se crea la clase OrdenarPorNotaNombre que implementa el interfaz comprator y se completa el método compare para ordenar primero por nota y luego por nombre.

****

Se crea un métodomostrarMejoresNotas que recibe como parámetro la lista, ordena esta lista con la clase comparador OrdenarPorNotaNombre y se añade al String las 10 primeras notas más altas con sus respectivos nombres



Se llama al método mostrarMejoresNotas y se muestra por pantalla

****

Genera esta salida por pantalla:

