Paquete: Operaciones LinkedList.

1. Importa los registros contenidos en el documento notas 1.txt, donde la información tiene el siguiente formato:

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

Para este apartado se ha cogido el método de Operaciones con ArrayList y se ha modificado, devolviendo una lista y añadiendo esta a la lista generada en el main.

```
public static LinkedList<Alumno> lectorArchivos(File archivo) {
    LinkedList<Alumno> devolucion = new LinkedList<>();
    try {
        Scanner lector = new Scanner (archivo);

        while(lector.hasNextLine()) {
            String linea = lector.nextLine();
            String[] lineaSeparada = linea.split(";");
            Alumno alumno = new Alumno(lineaSeparada[0], lineaSeparada[1], Integer.parseInt(lineaSeparada[2]),Integer.parseInt(lineaSeparada[3]),
                    Integer.parseInt(lineaSeparada[4]),Integer.parseInt(lineaSeparada[5]), Integer.parseInt(lineaSeparada[6]));
            devolucion.add(alumno);

        }
        lector.close();
    }catch(FileNotFoundException e) {
        System.out.println(e.getMessage());
    }
    return devolucion;
}
```

```
LinkedList<Alumno> listaAlumnos = new LinkedList<>();
LinkedList<Alumno> listaAlumnos2 = new LinkedList<>();
File archivo = new File("notas1.txt");
File archivo2 = new File("notas2.txt");

//1. Importa los registros contenidos en el documento notas1.txt, donde la información tiene el
//siguiente formato:
    //grupo;nombre,matemáticas;lengua;física;química;inglés
listaAlumnos.addAll(lectorArchivos(archivo));
```

2. Muestra el número de registros que contiene.

```
//2. Muestra el número de registros que contiene.
System.out.println("Mostrando la cantidad de registros que tiene la lista: " + listaAlumnos.size());
```

```
Mostrando la cantidad de registros que tiene la lista: 129
```

3. Ordena los registros por grupo y nombre.

```
//3. Ordena los registros por grupo y nombre.
Collections.sort(listaAlumnos);
escribirNuevoArchivo("LinkedListArchivos/AlumnosOrdenados", listaAlumnos, false);
```

Se muestran los resultados de la lista ordenada en un documento txt dentro de la carpeta LnkedListArchivos, llamado AlumnosOrdenados.

4. Añade el registro abajo indicado en la mitad de la lista.

1º ESO A;Plaza Gallego, Juan;1;1;3;1;1

```
    //1º ESO A;Plaza Gallego, Juan;1;1;3;1;1
Alumno alumno1 = new Alumno("1º ESO A", "Plaza Gallego, Juan", 1, 1, 3, 1, 1);
listaAlumnos.add((listaAlumnos.size()-1)/2, alumno1);
```

Se genera una variable de tipo Alumno y se instancia con los datos del registro. Se añade a la lista indicando la posición (mitad de la lista). Resultado de apartados 4, 5 ,6 y 7 se

muestran en documento de texto dentro de la carpeta LnkedListArchivos, llamado Notas1MasNotas2.

5. Añade el mismo registro al principio de la lista.

```
//5. Añade el mismo registro al principio de la lista.
listaAlumnos.addFirst(alumno1);
```

Se añade el mismo registro al principio de la lista con el método addFirst(). Resultado de apartados 4, 5 ,6 y 7 se muestran en documento de texto dentro de la carpeta LnkedListArchivos, llamado Notas1MasNotas2.

6. Importa los registros contenidos en el documento notas2.txt en una lista auxiliar.

```
LinkedList<Alumno> listaAlumnos2 = new LinkedList<>();
```

```
//6. Importa los registros contenidos en el documento notas2.txt en una lista auxiliar.
listaAlumnos2.addAll(lectorArchivos(archivo2));
```

Añado el contenido de la lista de notas2 a una lista auxiliar creada utilizando el método addAll(), solo se añadirán los registros que no sean iguales que alguno de los que ya existe en la lista. Se usa el método creado de lectorArchivos para leer el archivo y devolverlo en una lista.
Resultado de apartados 4, 5 ,6 y 7 se muestran en documento de texto dentro de la carpeta LnkedListArchivos, llamado Notas1MasNotas2.

7. Añade la lista auxiliar al principio de la lista original.

```
//7. Añade la lista auxiliar al principio de la lista original.
listaAlumnos.addAll(0, listaAlumnos2);
escribirNuevoArchivo("LinkedListArchivos/Notas1MasNotas2", listaAlumnos, false);
```

Se añade la lista auxiliar a la lista original con el método addAll() indicando la posición donde debe introducir los registros de la lista auxiliar.
Resultado de apartados 4, 5 ,6 y 7 se muestran en documento de texto dentro de la carpeta LnkedListArchivos, llamado Notas1MasNotas2.

8. Muestra el registro que está en la primera posición de la lista.

```
//8. Muestra el registro que está en la primera posición de la lista.
System.out.println("Mostrando el registro de la primera posicion de la lista original: " + listaAlumnos.get(0).toString());
```

```
Mostrando el registro de la primera posicion de la lista original: Alumno [grupo=1º ESO D, nombre=Molina González, Samuel Oliver, matematicas=4, lengua=3, fisica=3, quimica=3, ingles=4]
```

9. Muestra el registro que está en la última posición de la lista.

```
//9. Muestra el registro que está en la última posición posición de la lista.
System.out.println("Mostrando el registro de la ultima posicion de la lista: " + listaAlumnos.getLast());
```

```
Mostrando el registro de la ultima posicion de la lista: Alumno [grupo=1º ESO E, nombre=Vega Jiménez, Ana María, matematicas=6, lengua=7, fisica=6, quimica=4, ingles=3]
```

10. Muestra el registro que está en la mitad de la lista.

```
//10. Muestra el registro que está en la mitad de la lista.
System.out.println("Mostrando el registro que está en la mitad de la lista: " + listaAlumnos.get((listaAlumnos.size()-1)/2));
```

```
Mostrando el registro que está en la mitad de la lista: Alumno [grupo=1º ESO C, nombre=Fernández Rico, Paula, matematicas=2, lengua=3, fisica=3, quimica=1, ingles=3]
```

11. ¿Qué posición ocupa la primera ocurrencia del registro insertado?

```
//11. ¿Qué posición ocupa la primera ocurrencia del registro insertado?
System.out.println("Mostrando la primera coincidencia del registro insertado: " + listaAlumnos.indexOf(alumno1));
```

```
Mostrando la primera coincidencia del registro insertado: 16
```

12. ¿Qué posición ocupa la última ocurrencia del registro insertado?

```
//12. Qué posición ocupa la última ocurrencia del registro insertado?
System.out.println("Mostrando la ultima coincidencia del registro insertado: " + listaAlumnos.lastIndexOf(alumno1));
```

```
Mostrando la ultima coincidencia del registro insertado: 81
```

13. ¿El siguiente registro está en la lista?

1º ESO E;Postigo Vázquez, Salvador;3;6;5;3;4

Se crea una variable de tipo Alumno y se usa el método contains() para saber si existe o no un registro como el solicitado:

```
//13. ¿El siguiente registro está en la lista? 1º ESO E;Postigo Vázquez, Salvador;3;6;5;3;4
Alumno alumno2 = new Alumno("1º ESO E", "Postigo Vázquez, Salvador", 3, 6, 5, 3,4);
System.out.println("Mostrando si existe el registro Postigo Vázquez, Salvador en la lista: " + listaAlumnos.contains(alumno2));
```

```
Mostrando si existe el registro Postigo Vázquez, Salvador en la lista: false
```

14. ¿Y éste?

1º ESO A;García Fernández, María;6;5;6;5;5

Se crea una variable de tipo Alumno y se usa el método contains() para saber si existe o no un registro como el solicitado:

```
//14. ¿Y éste? 1º ESO A;García Fernández, María;6;5;6;5;5
Alumno alumno3 = new Alumno("1º ESO A", "García Fernández, María", 6, 5, 6, 5, 5);
System.out.println("Mostrando si existe el registro García Fernández, María en la lista: " + listaAlumnos.contains(alumno3));
```

```
Mostrando si existe el registro García Fernández, María en la lista: false
```

15. Elimina la primera ocurrencia del registro insertado y muestra su posición y contenido.

Se busca la posicion del alumno insertado con el método indexOf() y se usa para obtener la información con el método get() que devuelve el objeto de la posición introducida. Luego se usa el método removeFirstOccurence() para eliminar el primer registro con esa coincidencia.

```
//15. Elimina la primera ocurrencia del registro insertado y muestra su posición y contenido.
System.out.println("Mostrando la posicion e informacion del registro que se va a eliminar \n(se borrará la primera coincidencia del registro que se encuentre):\nPosicion:"
+ listaAlumnos.indexOf(alumno1) + " Informacion: " + listaAlumnos.get(listaAlumnos.indexOf(alumno1)).toString());
listaAlumnos.removeFirstOccurrence(alumno1);
```

```
Mostrando la posicion e informacion del registro que se va a eliminar
(se borrará la primera coincidencia del registro que se encuentre):
Posicion:16 Informacion: Alumno [grupo=1º ESO A, nombre=Plaza Gallego, Juan, matematicas=1, lengua=1, fisica=3, quimica=1, ingles=1]
```

16. Elimina el registro que ocupa la posición media de la lista mostrando su contenido.

Se usa el método remove() que se le indica la posición del objeto a borrar y devuelve el objeto. Se muestra usando el método toString

```
//16. Elimina el registro que ocupa la posición media de la lista mostrando su contenido.
System.out.println("Mostrando la informacion del registro de la posicion media de la lista que se va a borrar:\n" + listaAlumnos.remove((listaAlumnos.size()-1)/2));
```

```
Mostrando la informacion del registro de la posicion media de la lista que se va a borrar:
Alumno [grupo=1º ESO C, nombre=Fernández Rico, Paula, matematicas=2, lengua=3, fisica=3, quimica=1, ingles=3]
```

17. Elimina el registro cuyo nombre es: Carrillo Segura, Félix, indicando su posición y contenido.

Se genera un método para pasarle un nombre y una lista y que busque alguna coincidencia con ese nombre y lo borre. Mostrará la posición y la información del registro que se va a borrar.

```
public static void borrarAlumno(String nombre, LinkedList<Alumno> lista) {
    Iterator<Alumno> iterador = lista.iterator();
    boolean encontrado = false;
    Alumno alumno;
    while(iterador.hasNext() && encontrado == false) {
        alumno = iterador.next();
        if(alumno.getNombre().equalsIgnoreCase(nombre)) {
            System.out.println("Mostrando la posicion y la informacion del registro con nombre " + nombre);
            System.out.println("Posicion: " + lista.indexOf(alumno) + " Informacion: " + alumno.toString());
            lista.remove(alumno);
            encontrado = true;
        }

    }

}
```

```
//17. Elimina el registro cuyo nombre es: Carrillo Segura, Félix, indicando su posición y contenido.
borrarAlumno("Carrillo Segura, Félix", listaAlumnos);
```

```
Borrando el registro de nombre Carrillo Segura, Félix
Mostrando la posicion y la informacion del registro con nombre Carrillo Segura, Félix
Posicion: 18 Informacion: Alumno [grupo=1º ESO A, nombre=Carrillo Segura, Félix, matematicas=6, lengua=6, fisica=8, quimica=2, ingles=8]
```

18. Elimina el registro cuyo nombre es: Robles Ortiz, Rafael, indicando su posición y contenido.

Con el método anteriormente mencionado se borra el registro con nombre solicitado.

```
//18. Elimina el registro cuyo nombre es: Robles Ortiz, Rafael, indicando su posición y contenido.
borrarAlumno("Robles Ortiz, Rafael", listaAlumnos);
```

```
Borrando el registro de nombre Robles Ortiz, Rafael
No existe ningun alumno con este nombre para poder borrarlo
```

19. Intenta simular una pila (FIFO First In, First out) con la lista original. Realiza dos inserciones y extracciones detallando su funcionamiento.

Con los métodos de pila push() y pop() se procede a simular el funcionamiento de una pila.
El método push() añade un objeto al final de la lista o pila en este caso.
El método pop() elimina el último registro de la lista o pila en este caso.
Simulan una pila donde el último elemento puesto al final de la pila es el primero en salir.
Por último se muestra la información de los objetos a borrar.

```
//19. Intenta simular una pila (FIFO First In, First out) con la lista original. Realiza dos inserciones y extracciones detallando su funcionamiento.
listaAlumnos.push(alumno1);
listaAlumnos.push(alumno2);
escribirNuevoArchivo("LinkedListArchivos/Pila", listaAlumnos, false);
System.out.println("Eliminando: " + listaAlumnos.pop().toString());
System.out.println("Eliminando: " + listaAlumnos.pop().toString());
```

```
Eliminando: Alumno [grupo=1º ESO E, nombre=Postigo Vázquez, Salvador, matematicas=3, lengua=6, fisica=5, quimica=3, ingles=4]
Eliminando: Alumno [grupo=1º ESO A, nombre=Plaza Gallego, Juan, matematicas=1, lengua=1, fisica=3, quimica=1, ingles=1]
```

20. Intenta simular una cola (LIFO, Last In, Last Out) con la lista original. Realiza dos inserciones y extracciones detallando su funcionamiento.

Con los métodos de cola offer() y poll() se procede a simular el funcionamiento de una cola.
El método offer() añade al final de la lista o cola en este caso el objeto indicado.
El método poll() elimina el primer elemento de la lista o cola en este caso.
Simula una cola donde el último objeto se añade al final de la cola y se elimina el objeto de la cabeza o primer objeto de la lista o cola.
Por último se muestra la información de los objetos a borrar.

```
//20. Intenta simular una cola (LIFO, Last In, Last Out) con la lista original. Realiza dos inserciones y extracciones detallando su funcionamiento.
listaAlumnos.offer(alumno1);
listaAlumnos.offer(alumno2);
escribirNuevoArchivo("LinkedListArchivos/Cola", listaAlumnos, false);
System.out.println("Elimiando la cabeza de la lista (el primero): " + listaAlumnos.poll().toString());
System.out.println("Elimiando la cabeza de la lista (el primero): " + listaAlumnos.poll().toString());
```

```
Elimiando la cabeza de la lista (el primero): Alumno [grupo=1º ESO D, nombre=Molina González, Samuel Oliver, matematicas=4, lengua=3, fisica=3, quimica=3, ingles=4]
Elimiando la cabeza de la lista (el primero): Alumno [grupo=1º ESO A, nombre=Subiri Rueda, Fco José, matematicas=3, lengua=3, fisica=4, quimica=2, ingles=2]
```