ESTRUCTURAS DE DATOS
CURSO 2021-22.
Práctica 3: Listas.

1. OBJETIVOS:

- Definir un TAD lista ordinal y añadirle nuevas operaciones.
- Utilizar el TAD lista ordinal, haciendo especial hincapié en el uso de iteradores.
- Definir un TAD lista calificada y añadirle nuevas operaciones.
- Utilizar el TAD lista calificada.

2. Definición del TAD lista ordinal de evaluaciones.

2.1. El proyecto "ED 3 Practica. Listas".

Descargue el archivo "ED 3 Practica. Listas.zip" de *Moodle* y descomprímalo en una ubicación en la que no pierda el trabajo si se cierra el ordenador. Abra el proyecto con *IntelliJ* y, si aparece el mensaje de error, recuerde definir correctamente el JDK. Los archivos con los que va a trabajar en esta primera parte de la práctica son los siguientes:

- **NodoListaOrdinal**, **ListaOrdinal** son las clases que implementan el TAD Lista Ordinal de enteros a través de listas enlazadas, de forma similar a como se ha visto en clase.
- **IteradorListaOrdinal** para definir iteradores sobre la lista ordinal.
- **Evaluacion** es la clase en la que se definen los datos de una evaluación de un alumno en una asignatura: nombre de la asignatura, convocatoria y nota. Además de un constructor que recibe los tres datos anteriores, la interfaz de esta clase la componen los métodos: *getNombreAsignatura*, *getConvocatoria*, *getNota* y *mostrar* (visualiza los datos de la evaluación).
- **Pruebas** es una clase con un método *main* vacío en el que se irán realizando las pruebas que se vayan pidiendo en esta práctica.

2.2. Definición de un TAD lista ordinal de objetos *Evaluacion*.

Realizar las modificaciones oportunas en las clases *NodoListaOrdinal*, *ListaOrdinal* e *IteradorListaOrdinal* para que el TAD lista ordinal de enteros se transforme en un TAD lista ordinal de objetos *Evaluacion*.

A continuación, se probará su correcto funcionamiento en el *main* de la clase *Pruebas*:

- Crear cuatro objetos *Evaluacion* con los siguientes valores para nombreAsignatura, convocatoria y nota:
  - "ED", "Junio 19", 4.5.

- "ED", "Julio 19", -1.
- "ED", "Julio 20", 7.4.
- "Algebra", "Junio 18", 6.4.

Nota: un "no presentado" en la nota se representa a través de un valor negativo.

- Insertarlos ese orden en una lista ordinal de evaluaciones.
- Visualizar el contenido de la lista utilizando un *iterador*. Recuerde que para visualizar el contenido de una evaluación tiene el método *mostrar*.

El resultado de la ejecución será el siguiente:

```
------------ EVALUACIONES EN LA LISTA -----------
    ED (Junio 19): 4.5
    ED (Julio 19): NP
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4
```

2.3. Especialización del TAD Lista ordinal de evaluaciones.

Se pretende añadir ahora una nueva operación a la clase *ListaOrdinal*. Concretamente, el método:

```
public int numConvocatorias(String nombreAsignatura)
```

Este método contabilizará el número de evaluaciones realizadas en la asignatura `nombreAsignatura` para determinar el número de convocatorias utilizadas. Hay que tener en cuenta que un valor negativo en la nota representa "no presentado", por lo que no se contabilizará esa evaluación en las convocatorias.

Nota: para realizar este método no está permitido el uso de iteradores, es decir, deberá recorrerse la lista enlazada a través de sus nodos.

Nota: para comparar dos objetos *String* en Java, utilizar el método *equals* de dicha clase *String*.

A continuación, se incluirán las siguientes pruebas en el *main* de la clase *Pruebas:*

- En la lista de evaluaciones creada, contabilizar el número de convocatorias en las asignaturas de "ED", "Algebra" y "Fundamentos de Programación".
- Visualizar ambos resultados.

El resultado de las nuevas pruebas será el siguiente:

```
------------------------------
Convocatorias en ED: 2
Convocatorias en Algebra: 1
Convocatorias en Fundamentos de Programación: 0
```

3. Definición del TAD alumno.

Se pretende definir un TAD en el que se encapsulen los datos de un alumno: nombre, número de matrícula y expediente académico, formado por todas las evaluaciones realizadas en todas las asignaturas. Para implementar el expediente académico se utiliza el TAD Lista ordinal de evaluaciones definido en el apartado anterior de esta práctica.

Este TAD se implementa en la clase **Alumno** contenida en el proyecto "ED 3 Practica. Listas". Esta clase contiene los atributos *nombre*, *matricula* y *expediente* para almacenar los datos de un alumno. Además, define su interfaz a través de los siguientes métodos públicos:

- Constructor con el nombre y la matrícula del alumno.
- *getNombre*, *getMatricula*, *setNombre*, *setMatricula*.
- *void nuevaEvaluacion(Evaluacion evaluación)* que añade una nueva evaluación al alumno.
- *boolean estaAprobado(String nombreAsig)* que determina si el alumno ha aprobado o no la asignatura *nombreAsig*.

En los apartados que siguen, se pretende definir nuevos métodos que amplíen la interfaz del TAD alumno.

3.1. Método *asignaturasAprobadas.*

Añadir a la clase *Alumno* el método:

```
public ListaOrdinal asignaturasAprobadas()
```

Que devuelva una lista ordinal de evaluaciones que contenga únicamente las evaluaciones aprobadas por el alumno, es decir, con una nota superior o igual a 5. En definitiva, esta lista se corresponde con las asignaturas que el alumno tiene aprobadas, ya que no puede haber aprobado la misma asignatura dos veces. En caso de que el alumno no haya aprobado ninguna asignatura, este método devolverá una lista vacía.

Nota: para recorrer una lista ordinal de evaluaciones es obligatorio hacerlo con un iterador.

3.2. Método *mediaAprobadas.*

Añadir a la clase *Alumno* el método:

```
public double mediaAprobadas()
```

Que determine la calificación media de las asignaturas (evaluaciones) que el alumno tenga aprobadas. En caso de que el alumno no tenga aprobada ninguna asignatura, este método debe devolver 0.0.

Nota: aunque no es obligatorio, se recomienda utilizar el método *asignaturasAprobadas*.

Nota: para recorrer una lista ordinal de evaluaciones es obligatorio hacerlo con un iterador.

3.3. Método *getNumAprobadas.*

Añadir a la clase *Alumno* el método:

```
public int getNumAprobadas()
```

Que determine el número de asignaturas aprobadas que tiene el alumno.

Nota: aunque no es obligatorio, se recomienda utilizar el método *asignaturasAprobadas*.

Nota: si se hiciera algún recorrido por una lista ordinal de evaluaciones, se debería hacer con un iterador.

3.4. Método *mostrar.*

Añadir a la clase *Alumno* el método:

```
public void mostrar()
```

Que visualice los datos del alumno con el siguiente formato:

```
Felipe García Gómez. Matricula: 1253
    ED (Junio 19): 4.5
    ED (Julio 19): NP
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4
4 evaluaciones y 2 asignaturas aprobadas con calificación media 6.9
------------------------------
```

Si el alumno no tiene ninguna evaluación en su expediente, el formato será:

```
Alicia Blázquez Martín. Matricula: 5622
No ha realizado ninguna evaluación.
------------------------------
```

Aclaración: si el alumno se ha evaluado de alguna/s asignatura/s, pero no ha aprobado ninguna, el formato es el primero.

3.5. Pruebas de los métodos añadidos.

Realizar las siguientes pruebas en el *main* de la clase *Pruebas*:

- Crear un objeto *Alumno* con nombre "Felipe García Gómez" y matrícula 1253.
- Crear un objeto *Alumno* con nombre "Alicia Blázquez Martín" y matrícula 5622.
- Asociar a "Felipe García Gómez" las cuatro evaluaciones creadas en el apartado 2.2.
- Pedirle a "Felipe García Gómez" las asignaturas que tiene aprobadas (método *asignaturasAprobadas*).
- Recorrer con un iterador la lista devuelta por el método anterior y visualizar la información de las asignaturas aprobadas por "Felipe García Gómez".
- Pedirle a "Alicia Blázquez Martín" las asignaturas que tiene aprobadas (método *asignaturasAprobadas*).
- Recorrer con un iterador la lista devuelta por el método anterior y visualizar la información de las asignaturas aprobadas por "Alicia Blázquez Martín".
- Mostrar la información de ambos alumnos (método *mostrar*).

El resultado de las nuevas pruebas será el siguiente:

```
----------- Asignaturas aprobadas por Felipe García Gómez ----------
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4

----------- Asignaturas aprobadas por Alicia Blázquez Martín:

----------- MOSTRAR LOS ALUMNOS ------------
Felipe García Gómez. Matricula: 1253
    ED (Junio 19): 4.5
    ED (Julio 19): NP
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4
4 evaluaciones y 2 asignaturas aprobadas con calificación media 6.9
--------------------------------
Alicia Blázquez Martín. Matricula: 5622
No ha realizado ninguna evaluación
--------------------------------
```

3.6. Uso de la biblioteca estándar de Java.

En el apartado 2, se codificaron unas listas ordinales a medida para los expedientes de los alumnos. En este apartado se utilizará la biblioteca Java para definir dichos expedientes. Más concretamente, se volverá a definir la clase *Alumno* pero utilizando la clase ***LinkedList*** en lugar de la clase *ListaOrdinal*.

En primer lugar, se hará una copia de la clase *Alumno*, llamándola *AlumnoBib*. Para ello, lo más fácil es hacer esta copia dentro de *IntelliJ*, seleccionado en la ventana del proyecto la clase *Alumno* y copiándola. En cualquier caso, lo importante es que tanto el archivo como la clase de la copia se llamen *AlumnoBib*.

En *AlumnoBib*:

- Utilizar *LinkedList<Evaluacion>* en lugar de *ListaOrdinal*. La interfaz de *LinkedList* puede consultarse en el API de Java o en las transparencias de la asignatura.
- Utilizar *Iterator<Evaluacion>* en lugar de *IteradorListaOrdinal*. En este caso los métodos no cambian, siguen siendo *hasNext* y *next*.
- Tanto *LinkedList* como *Iterator* tendrán que ser importadas al comienzo de *AlumnoBib*.

En el método *main* de *Pruebas*:

- Crear un objeto *AlumnoBib* con nombre “Eduardo Parra Martín” y matrícula 8765.
- Crear un objeto *AlumnoBib* con nombre “Sonia Torres Pardo” y matrícula 2345.
- Asociar a “Eduardo Parra Martín” las cuatro evaluaciones creadas en el apartado 2.2.
- Mostrar la información de ambos (método *mostrar*).

El resultado de las nuevas pruebas será el siguiente:

```
---------- MOSTRAR LOS ALUMNOS BIBLIOTECA ------------
Eduardo Parra Martín. Matricula: 8765
    ED (Junio 19): 4.5
    ED (Julio 19): NP
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4
4 evaluaciones y 2 asignaturas aprobadas con calificación media 6.9
--------------------------------
Sonia Torres Pardo. Matricula: 2345
No ha realizado ninguna evaluación
--------------------------------
```

4. Definición del TAD Lista Calificada de alumnos.

Una vez definido el TAD Alumno, el objetivo ahora es definir un grupo de clase, formado por alumnos. Para ello, primero se requiere definir una lista de alumnos.

Si tenemos en cuenta que no es posible tener a un alumno repetido en un grupo y que la matrícula puede servir como campo clave, llegamos a la conclusión de que la lista de alumnos va a ser una lista calificada.

Las nuevas clases del proyecto con las que se va a trabajar ahora son:

- **NodoListaCalificada**, **ListaCalificada** son las clases que implementan un TAD lista calificada de alumnos con clave entera, mediante una lista enlazada. Se trata de la implementación de las transparencias de la asignatura.
- **IteradorListaCalificada** para definir iteradores sobre la lista calificada.

4.1. Definición de un TAD lista calificada de objetos *Alumno* y clave el número de matrícula.

Este primer apartado lo tenemos casi realizado, ya que la lista calificada que viene en el proyecto también es de objetos *Alumno* y clave el número de matrícula. Sin embargo, podemos apreciar que la clave, es decir, el número de matrícula viene incluida dentro del objeto *Alumno* (atributo *matricula*). Por ese motivo, nos proponemos como objetivo no repetir la clave en cada nodo de la lista enlazada. Las tareas a realizar son las siguientes:

- En *NodoListaCalificada* se eliminará el atributo *clave*, ya que la clave ya se encuentra implícita en el atributo *dato* (*Alumno*).

  Observación: el método *getClave* no debe eliminarse, lo que hay que hacer es codificarlo correctamente.
- En la clase *ListaCalificada* se modificará el método *insertar*, de manera que reciba un solo parámetro, el *dato* (*Alumno*). Nuevamente, la clave viene implícita en el dato.

A continuación, se probará el correcto funcionamiento de la lista calificada en el *main* de la clase *Pruebas*. Con este objetivo, se incluirá código para realizar las siguientes pruebas:

- Crear un nuevo objeto *Alumno* con nombre “Pedro Jiménez del Pozo” y matrícula 8510.
- Crear un nuevo objeto *Evaluación* en “Fundamentos de Programación”, en “Enero 19” y con una nota de 8.8.
- Asociarle a “Pedro Jiménez del Pozo” la nueva evaluación.
- Crear un objeto *ListaCalificada* e insertarle los siguientes alumnos por este orden: “Pedro Jiménez del Pozo”, “Felipe García Gómez” y “Alicia Blázquez Martín”.
- Visualizar el contenido de la lista calificada utilizando un *iterador*. Recuerde que para visualizar el contenido de un alumno tiene el método *mostrar*.

El resultado de la ejecución será el siguiente:

```
------------ ALUMNOS EN LA LISTA -------------
Felipe García Gómez. Matricula: 1253
    ED (Junio 19): 4.5
    ED (Julio 19): NP
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4
4 evaluaciones y 2 asignaturas aprobadas con calificación media 6.9
---------------------------------
Alicia Blázquez Martín. Matricula: 5622
No ha realizado ninguna evaluación
---------------------------------
Pedro Jiménez del Pozo. Matricula: 8510
    Fundamentos de Programación (Enero 19): 8.8
1 evaluaciones y 1 asignaturas aprobadas con calificación media 8.8
---------------------------------
```

4.2. Especialización del TAD lista calificada de alumnos.

4.2.1. Método *borrarMenores*.

Se pretende añadir ahora una nueva operación a la clase *ListaCalificada*. Concretamente, el método:

```
public void borrarMenores(int clave)
```

Este método recorrerá los nodos de la lista enlazada para eliminar todos aquellos que tienen una clave estrictamente menor que la recibida como parámetro.

Nota: para realizar este método no está permitido el uso de iteradores, ni del método *borrar*. Deberá realizarse directamente sobre los nodos de la lista enlazada en un solo recorrido.

4.2.2. Método *borrarMayores*.

Se pretende añadir ahora una nueva operación a la clase *ListaCalificada*. Concretamente, el método:

```
public void borrarMayores(int clave)
```

Este método recorrerá los nodos de la lista enlazada para eliminar todos aquellos que tienen una clave estrictamente mayor que la recibida como parámetro.

Nota: para realizar este método no está permitido el uso de iteradores, ni del método *borrar*. Deberá realizarse directamente sobre los nodos de la lista enlazada en un solo recorrido.

4.2.3. Pruebas de los métodos añadidos

A continuación, se incluirán las siguientes pruebas en el *main* de la clase *Pruebas:*

- En la lista de alumnos creada, eliminar con el método *borrarMenores* las claves inferiores a 6000.
- Visualizar nuevamente el contenido de la lista de alumnos utilizando un iterador.
- En la lista resultante, eliminar con el método *borrarMenores* las claves inferiores a 9000.
- Volver a visualizar la lista de alumnos con un iterador.
- Volver a meter en la lista de alumnos los tres alumnos que tenemos creados: “Felipe García Gómez”, “Alicia Blázquez Martín” y “Pedro Jiménez del Pozo”.
- Eliminar de la lista los alumnos con claves superiores a 7000 con *borrarMayores*.
- Visualizar nuevamente el contenido de la lista de alumnos utilizando un iterador.
- En la lista resultante, eliminar con el método *borrarMayores* las claves superiores a 1000.
- Volver a visualizar la lista de alumnos con un iterador.

El resultado de las nuevas pruebas será el siguiente:

```
---------- Borramos las claves menores a 6000------------
Pedro Jiménez del Pozo. Matricula: 8510
    Fundamentos de Programación (Enero 19): 8.8
1 evaluaciones y 1 asignaturas aprobadas con calificación media 8.8

------------ Borramos las claves menores a 9000 ------------

----- Metemos todos los alumnos y borramos las claves mayores a 7000 ------
Felipe García Gómez. Matricula: 1253
    ED (Junio 19): 4.5
    ED (Julio 19): NP
    ED (Junio 20): 7.4
    Algebra (Junio 18): 6.4
```

```
4 evaluaciones y 2 asignaturas aprobadas con calificación media 6.9
---------------------------------
Alicia Blázquez Martín. Matricula: 5622
No ha realizado ninguna evaluación
---------------------------------

------------ Borramos las claves mayores a 1000 -----------
```

5. Definición del TAD grupo de alumnos.

Se pretende definir un TAD en el que se encapsulen los datos de un grupo de clase: nombre del grupo y alumnos que lo componen. Obviamente, utilizaremos la lista calificada implementada en el apartado anterior para almacenar los alumnos del grupo.

Este TAD se implementa en la clase **GrupoAlumnos** contenida en el proyecto "ED 3 Practica. Listas". Esta clase contiene los atributos *nombre* y *listaAlumnos* para almacenar los datos de un grupo; además de los siguientes métodos públicos:

- Constructor con el nombre del grupo, que crea un grupo vacío.
- *getNombre*, *setNombre*.

En los apartados que siguen, se ampliará la clase *GrupoAlumnos* con nuevos métodos.

5.1. Método *nuevoAlumno*.

Añadir a la clase *GrupoAlumnos* el método:

```
public void nuevoAlumno(Alumno alumno)
```

Que añada un nuevo alumno al grupo.

5.2. Método *getNumAlumnos*.

Añadir a la clase *GrupoAlumnos* el método:

```
public int getNumAlumnos()
```

Que devuelva el número de alumnos que tiene el grupo.

5.3. Método *getAlumno*.

Añadir a la clase *GrupoAlumnos* el método:

```
public Alumno getAlumno(int matricula)
```

Que devuelva el alumno que tiene la matrícula recibida como parámetro. En caso de que dicho alumno no se encuentre en el grupo, este método devuelve *null*.

5.4. Método *porcentajeAprobados*.

Añadir a la clase *GrupoAlumnos* el método:

```
public double porcentajeAprobados(String nombreAsignatura)
```

Que determine el porcentaje de alumnos en el grupo que han aprobado la asignatura recibida como parámetro. En caso de que el grupo no tenga alumnos, este método devolverá 0.0.

Nota: para recorrer una lista calificada de alumnos es obligatorio hacerlo con un iterador.

Nota: recuerde que la clase *Alumno* dispone del método *estaAprobado(String nombreAsignatura)* que determina si un alumno está aprobado o no en una asignatura.

5.5. Pruebas de los métodos insertados.

Realizar las siguientes pruebas en el *main* de la clase *Pruebas*:

- Crear un objeto *GrupoAlumno* con nombre "GX11".
- Añadir en dicho grupo a los tres objetos Alumnos que tenemos creados: "Felipe García Gómez", "Alicia Blázquez Martín" y "Pedro Jiménez del Pozo".
- Visualizar el número de alumnos que tiene el grupo.
- Pedirle al grupo el alumno con número de matrícula 8510.
- Mostrar la información del alumno recibido.
- Calcular el porcentaje de alumnos que han aprobado "ED" en el grupo.

El resultado de las nuevas pruebas será el siguiente:

```
------------ CREADO EL GRUPO GX11------------
El grupo GX11 tiene 3 alumnos
------------ Grupo GX11. Alumno con matrícula 8510 ------------
Pedro Jiménez del Pozo. Matricula: 8510
    Fundamentos de Programación (Enero 19): 8.8
1 evaluaciones y 1 asignaturas aprobadas con calificación media 8.8
--------------------------------
porcentaje de aprobados en ED el grupo GX11: 33.333333333333336
```

6. Entrega de la práctica.

Se entregará el proyecto *IntelliJ* resultante: "ED 3 Practica. Listas", **que tendrá que tener exactamente ese nombre**.

Para entregarlo, se comprimirá el proyecto en un archivo, preferentemente ZIP, y se subirá a la plataforma. El nombre de dichos archivo ZIP será el mismo: "ED 3 Practica. Lista.zip".

No olvide que el proyecto debe incluir las pruebas solicitadas en este enunciado, mediante el código del método *main*.