INSTRUCCIONES:

1. Iniciar sesión y crear en Eclipse un nuevo workspace **examen**. Descargar del Campus Virtual el material del examen. Importar el proyecto Eclipse con parte del código de la aplicación, que se puede reutilizar o no, y modificar libremente. Se incluye un ejemplo de fichero de entrada y el fichero de salida correspondiente.
2. Desconectar el cable de red. Solo se volverá a conectar una vez finalizado el examen para subirlo a la tarea correspondiente en el Campus Virtual. **IMPORTANTE**: Comprimir el proyecto Eclipse en un archivo llamado ***NombreApellidos-L5.zip (L5 es el grupo)***
3. Se prohibe usar USBs, CDs, teléfonos móviles o cualquier otro dispositivo durante el examen.

Una importante multinacional oferta becas para estudiantes de ingeniería. Hay dos tipos de estudiantes que pueden optar a distintas becas: Estudiantes de Grado y de PostGrado (Máster, doctorado, ...).

Se trata de diseñar e implementar una aplicación para la gestión de estas solicitudes. Por simplificar únicamente se almacenará de todos los aspirantes a beca:

- *nombre (string)*
- *nota media del expediente académico (int en el rango 0-100)*
- Solamente para los aspirantes estudiantes de un Grado se almacenará también su *número de orden de solicitud*, mientras que solamente para los aspirantes que estudian un postGrado se almacenará una cadena de texto indicando los estudios de Grado que ya poseen ("*Informática*", "Telecomunicaciones", "Civil", "Geomática", "*Electrónica*").

Los datos de los aspirantes están en un fichero llamado `aspirantes.txt`, ordenados según el momento de llegada de la solicitud, con el siguiente formato:

`<nombre>;<nota>;<tipo de aspirante>[;<estudiosGrado>]`

```
Sofia Ramirez;74;PostGrado;Civil
Miguel Hernandez;21;Grado
Maria Rodriguez;86;PostGrado;Geomático
Armando Sanchez;65;Grado
Claudia Hernandez;36;Grado
...
```

Los aspirantes serán procesados mediante una clase que llamaremos **RegistroAspirantes**. La clase **RegistroAspirantes** tiene dos listas de espera, una para cada tipo de aspirante, que llamaremos **ListaEsperaGrado** y **ListaEsperaPostGrado**. Cada una requerirá una clase Java. La clase **RegistroAspirantes** debe tener los siguientes métodos:

1. **`void admitirAspirante(Aspirante u).`** Almacena el aspirante pasado como parámetro en la lista de espera que le corresponda según el tipo de aspirante. Además, si el aspirante es **AspiranteGrado,** se le asigna su número de orden de solicitud, y si es **AspirantePostGrado** se almacenan sus estudios de Grado.
2. **`ListaEspera ordenarAlfa()`**: Este método retornará una lista de espera ordenada alfabéticamente. Se usará en la clase Main para mostrar los aspirantes standard por orden alfabético.

Cada clase "lista de espera" contendrá una lista de aspirantes (`ArrayList o LinkedList`) y los siguientes métodos:

1. **`void add(Aspirante s):`** Añade a la lista de espera el aspirante pasado como parámetro.
2. **`void registrarAspirantes()`**. Implementar este método para que realice lo siguiente:
   a. En el caso de la lista de espera de estudiantes de ***PostGrado*** se insertan los aspirantes de manera que queden ordenados por sus estudios de Grado, y en caso de estudios iguales por nota. Los datos de cada aspirante ***PostGrado*** registrado (nombre, nota, tipo) se guardan en un fichero de texto llamado `registroAspirantesPostGrado.out.txt`
   b. En la lista de espera de estudiantes de ***Grado*** los aspirantes deberán están ordenados por su número de orden de solicitud, y se mostrará en la consola estándar la información de cada aspirante (nombre, nota, número de orden de solicitud).
3. **`void sort()`**. Ordena los aspirantes de una lista de espera usando un criterio de comparación entre los implementados. (con un `Comparator`).

TAREAS

Una vez leído y entendido el enunciado, realizar las siguientes tareas:

1. Diseñar la jerarquía de clases y sus relaciones. Dibujar un diagrama UML **antes de empezar a codificar.**
2. **Implementar** el diseño realizado en un proyecto Eclipse llamado **examen**.
3. **Realizar una prueba unitaria /test) para comprobar la robustez del método `admitirAspirante`.**
4. Incluir todas las excepciones que se consideren necesarias en el código**.**
5. Opcional para subir nota: Implementar un método para almacenar en un archivo comprimido los datos de cada aspirante PostGrado registrado, además de hacerlo en el fichero de texto.