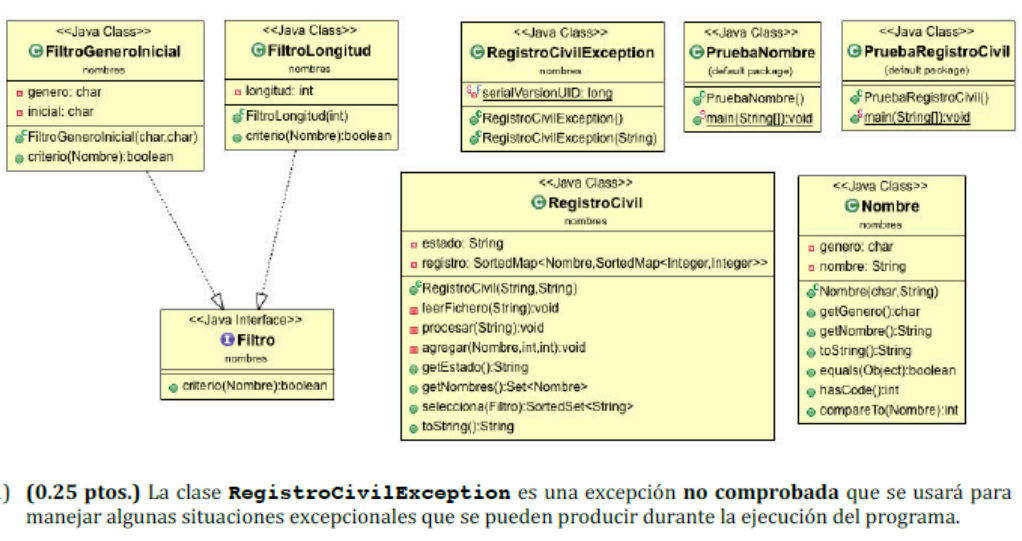
NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- El alumnado debe haber comprobado si cumple los requisitos para acogerse a la modalidad de evaluación continua. De no ser así, además de la prueba práctica, deberá realizar un examen teórico adicional y, por lo tanto, ponerse en contacto con el profesorado.
- Al **inicio del contenido de cada fichero** realizado deberá aparecer un comentario con los **apellidos y nombre, titulación y grupo.**
- **Está permitido:**
  - Consultar los apuntes (CV), la API (Internet), la guía rápida de la API (CV).
  - Añadir métodos privados a las clases.
- **No está permitido**:
  - Intercambiar documentación con otros compañeros.
  - Recibir ayuda de otras personas. Se debe realizar personal e individualmente la solución del ejercicio propuesto.
  - Añadir métodos no privados a las clases.
  - Añadir variables o constantes a las clases.
  - Modificar la visibilidad de las variables, constantes y métodos que aparecen en el diagrama UML.
  - Modificar el código suministrado.
- Una vez terminado el ejercicio, se debe crear un archivo comprimido de la carpeta SRC del proyecto y subirlo a la tarea correspondiente del campus virtual.
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización.
- Para la corrección del ejercicio se utilizarán **programas de detección de copias/plagios**.
- Con posterioridad a la realización del ejercicio, el profesor podrá convocar a determinado/as alumno/as para realizar **entrevistas personales** con objeto de comprobar la autoría de las soluciones entregadas.

Se pide desarrollar un programa para la gestión de los datos sobre los nombres de las personas nacidas en un país. Se debe crear un proyecto `prRegistro` con las clases `RegistroCivilException`, `Nombre`, `RegistroCivil`, la interfaz `Filtro` y dos clases que la implementan: `FiltroGeneroInicial` y `FiltroPorcentaje`, todas ellas en el paquete `nombres`. Además se creará en el paquete por defecto del proyecto, la clase `PruebaNombre` para probar algunas de las clases anteriores.

En el campus virtual se puede consultar: el diagrama de clases, la clase `PruebaRegistroCivil`, el fichero `registrocivil.txt` con datos de prueba, y el fichero `salidaRegistroCivil.txt` que contiene la salida completa de la ejecución de la clase `PruebaRegistroCivil` usando como entrada los datos de prueba proporcionados.



1) **(0.25 ptos.)** La clase **`RegistroCivilException`** es una excepción **no comprobada** que se usará para manejar algunas situaciones excepcionales que se pueden producir durante la ejecución del programa.

2) **(2 ptos.)** La clase **`Nombre`** mantendrá información relativa a los nombres con los que las diferentes personas se han inscrito en el registro civil en distintos estados o regiones de un país a lo largo de los años. Los datos que se almacenan son: el género del nombre (`char`) que tomará los valores '`F`' o '`M`', para nombres femeninos y masculinos respectivamente, y el nombre (`String`) propiamente dicho.

   La clase `Nombre` dispondrá de los siguientes constructores y métodos de instancia:

   a. Un constructor para crear un objeto de la clase dados sendos valores para los atributos descritos (en ese mismo orden). Debe lanzar la excepción `RegistroCivilException` si el carácter del género es distinto de '`F`' o '`M`' o si la cadena nombre está vacía o no existe.
   b. Métodos para obtener los valores de los diferentes atributos (*getter*).
   c. Dos objetos de la clase `Nombre` se consideran iguales cuando tienen el mismo nombre (sin diferenciar mayúsculas de minúsculas) y el mismo género.
   d. La clase `Nombre` proporciona el orden natural. Un `Nombre` se considera menor que otro si su nombre es previo lexicográficamente (sin diferenciar mayúsculas de minúsculas). En caso de igualdad de nombre, será menor el `Nombre` con un género previo lexicográficamente.
   e. La representación textual de un `Nombre` vendrá dada por una cadena con el formato `(nombre, género)`, por ejemplo `(Stephanie, F)`

3) **(1 pto.)** La clase **`PruebaNombre`** es una clase distinguida que creará y mostrará por la consola los siguientes nombres, cuyos datos recibirá el `main` a través de su argumento String `args[]`:

   ```
   Charlotte F Lorena F Gael M Alexis M
   ```

   Se debe ubicar en el paquete por defecto del proyecto.
   En ella deben tratarse las distintas situaciones erróneas que puedan darse emitiendo un mensaje diferenciado de error (`System.err`). Estas situaciones incluirán:
   - No se proporcionan valores suficientes como argumentos del `main`.
   - Alguno de los valores introducidos como género no es una cadena de un único carácter.
   - Alguno de los valores introducidos como género es una cadena de un único carácter, pero no es ni `F` ni `M`.

4) **(4.25 ptos.)** La clase **`RegistroCivil`** almacena información acerca de los nombres que se han puesto a las personas nacidas en un estado o región de un país a lo largo de los años. En concreto se guardará el código del estado o región (String) y una correspondencia ordenada que asocia un objeto `Nombre` con otra correspondencia ordenada, que asocia cada año con el número de veces que ese nombre se ha usado en dicho año. Un ejemplo de los datos guardados en el registro podría ser: `{(Aaron,M)={2015=58, 2016=63, 2017=76, 2018=64, 2019=65, 2020=76}, (Abdiel,M)={2015=43, 2016=31, 2017=31, 2020=22}, […]}` , que nos dice que 58 niños en 2015, 63 en 2016, 76 en 2017, 64 en 2018, 65 en 2019 y 76 en 2020 fueron registrados con el nombre de Aaron, que Abdiel se le puso a 43 niños en 2015, a 31 en 2016, etc.

   La información sobre los nombres se proporciona en un fichero de datos organizado por líneas con el siguiente formato:

   código-del-estado-o-región;género;año;nombre;número-de-repeticiones

   El siguiente fragmento de fichero muestra como ejemplo algunas líneas con datos de nombres masculinos (M) y femeninos (F), de los estados de California (CA), Puerto Rico (PR) y Nueva York (NY):

   ```
   CA;F;1985;Angeline;23
   CA;M;2020;Eziah;9
   CA;M;2020;Fabio;9
   PR;F;2015;Amanda;191
   PR;F;2015;Mikaela;177
   PR;M;2015;Ryan;86
   PR;F;2015;Alaia;133
   NY;F;1910;Alice;410
   NY;F;1910;Marion;387
   ```

   La clase `RegistroCivil` contiene:

   a. Un **constructor con dos parámetros**, el código de un estado o región y el nombre de un fichero, que inicializa el atributo `estado` y carga en el registro los datos incluidos en el fichero, haciendo uso del método `leerFichero`. Si el código es la cadena vacía o nula se lanzará la excepción `RegistroCivilException` y se informará del error.
   b. El método **`leerFichero`**, que recibe como parámetro el nombre de un fichero, lo abre, lo procesa y añade a la correspondencia la información necesaria. Si el fichero no puede abrirse, se debe lanzar una excepción del tipo `RegistroCivilException`. Para realizar esta tarea, hará uso del método privado **`procesar`** que se describe en el siguiente apartado.
   c. El método **`procesar`**, que recibe como parámetro una línea del fichero (`String`). Si el código del estado o región leído no coincide con el código del registro civil que se está creando, esa línea no se tiene en cuenta. Cada línea contiene los siguientes datos (código del estado o región, género, año, nombre y número de veces), separados por los siguientes delimitadores: "[;]". El método extrae los campos de la línea, crea un objeto de la clase `Nombre` y agrega a la asociación la información del año y el número de veces que ese nombre se ha usado. Para ello hará uso del método **`agregar`** que se describe en el siguiente apartado. Si se produce algún error de formato en una línea (falta algún dato o el dato no es del tipo correcto), esa línea se ignora.
   d. El método **`agregar`**, que recibe como parámetros un objeto `Nombre`, un año y un número. Su tarea es añadir al registro el número de veces que el nombre se ha puesto el año indicado. Si el nombre ya está en el registro, se actualiza la entrada correspondiente con los nuevos datos (año y número). En caso contrario, se añade una entrada nueva y se inicializa con el año y el número.
   e. El método **`getEstado`** devuelve el valor del atributo `estado`.
   f. El método **`getNombres`** devuelve un conjunto con los nombres almacenados en la asociación.
   g. El método **`toString`** devuelve la representación del objeto como una cadena de texto, usando `StringBuilder` o `StringJoiner`, con el siguiente formato (donde PR es el código del estado o región):

   ```
   PR
   (Aaron,M):        {2015=58, 2016=63, 2017=76, 2018=64, 2019=65, 2020=76}
   (Abdiel,M):       {2015=43, 2016=31, 2017=31, 2020=22}
   (Abigail,F):      {2015=31, 2016=27, 2017=21, 2018=27}
   (Abraham,M):      {2015=24, 2017=21, 2018=23}
   ....
   ```

5) **(2.5 ptos.)** Con el objetivo de extraer del registro conjuntos de nombres que cumplan ciertas condiciones que nos interesan, se deben implementar las interfaces, clases y métodos que se describen a continuación:
   a. **(0.5 ptos.)** La interfaz `Filtro` define un método `boolean criterio(Nombre n)` que devolverá un valor `true` o `false`, si el `Nombre` que recibe como parámetro cumple una condición que se determinará en las clases que implementen la interfaz.
   b. **(0.75 ptos.)** La clase `FiltroGeneroInicial` implementa la interfaz `Filtro`. Tiene un constructor con dos parámetros que inicializa los atributos `genero` e `inicial`, de tipo `char`, que posee la clase; y el método **`criterio`**, que devuelve `true` si el género y la inicial del nombre que recibe como parámetro coinciden con los valores de los atributos, y `false` en caso contrario.
   c. **(0.75 ptos.)** La clase `FiltroLongitud` implementa la interfaz `Filtro`. Tiene un constructor con un parámetro que inicializa el atributo `longitud (int)` de la clase. El método **`criterio`** devuelve **`true`** si la longitud del nombre que recibe como parámetro coinciden con el valor del atributo, y `false` en caso contrario.
   d. **(0.5 ptos.)** El método `selecciona`, que se añadirá a la clase `RegistroCivil`. Este método recibe como parámentro un objeto de la clase `Filtro` y devuelve un conjunto ordenado con los nombres del registro que cumplen la condición implementada en el `Filtro`.

6) La clase `PruebaRegistroCivil` es una clase distinguida que se proporciona en el cv que se debe ubicar en el paquete por defecto del proyecto. A continuación, se presenta un fragmento de la salida de la ejecución del programa. La salida completa se puede consultar en el fichero salidaRegistroCivil.txt.

```
PR


(Aaron,M):      {2015=58, 2016=63, 2017=76, 2018=64, 2019=65, 2020=76}
(Abdiel,M):     {2015=43, 2016=31, 2017=31, 2020=22}
(Abigail,F):    {2015=31, 2016=27, 2017=21, 2018=27}
(Abraham,M):    {2015=24, 2017=21, 2018=23}
(Adrian,M):     {2015=218, 2016=198, 2017=180, 2018=167, 2019=160, 2020=106} […]
[…]
(Zahir,M):      {2015=35, 2016=32, 2020=20}
(Zaid,M):       {2018=35, 2019=35, 2020=28}
(Zoe,F):        {2015=26, 2017=20}

Nombres femeninos que empiezan por 'L': [Laia, Lara, Layla, Leah, Lia, Liah, Lorena,
Lucia, Luna, Lyah]
Nombres de 3 letras: [Amy, Ana, Ann, Eva, Gia, Ian, Jan, Lia, Mia, Noa, Zoe]
```