



APELLIDOS, Nombre

TITULACIÓN Y GRUPO

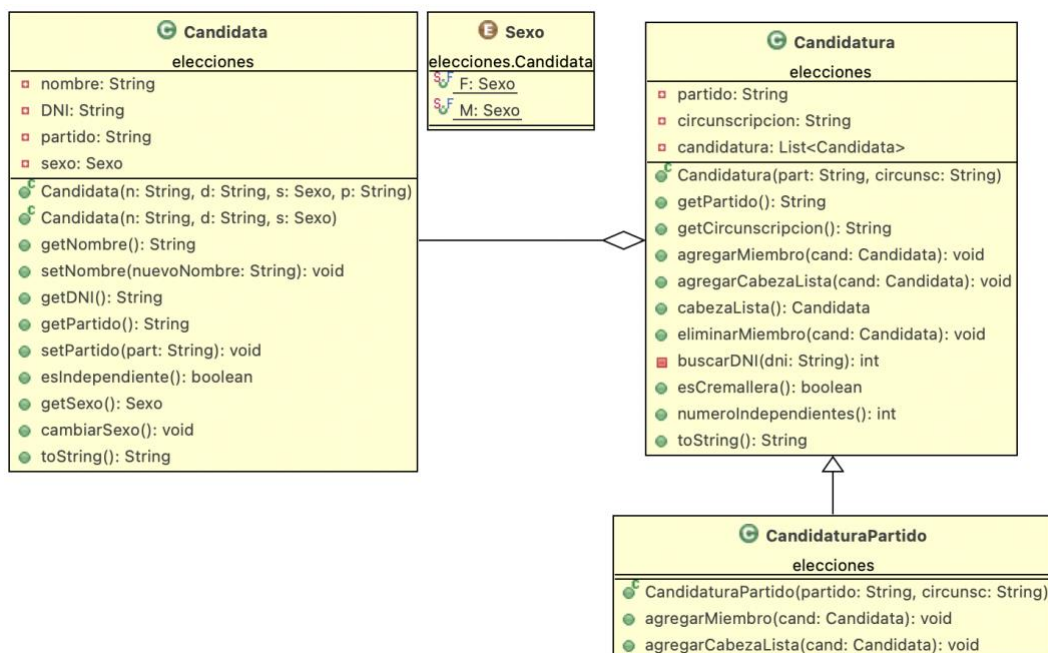
MÁQUINA

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- Al **inicio del contenido de cada fichero** realizado deberá aparecer un comentario con tus **apellidos y nombre, titulación y grupo**.
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no lo sabes hacer, *no debes pararte en él indefinidamente*. Puedes abordar otros.
- **Está permitido:**
 - Consultar los apuntes (CV), y 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 se indican.
 - Modificar el código suministrado.
- Una vez terminado el ejercicio, debéis subir (a la tarea creada en el campus virtual para ello) un fichero comprimido de la carpeta **src** que hayáis realizado y usáis vuestros apellidos y nombre para su denominación (**Apellido1Apellido2Nombre.rar** o **.zip**).
- 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.

Proyecto prElecciones

Se va a crear una aplicación para mantener listas electorales. Para ello se crearán las clases **Candidata**, **Candidatura** y **CandidaturaPartido** en un paquete **elecciones**.



- 1) **(2,5 puntos)** La clase `Candidata` mantiene información sobre personas que se puedan presentar como candidatas a unas elecciones, incluyendo su nombre (de tipo `String`), su afiliación si pertenece a algún partido político (de tipo `String`), su documento de identidad (de tipo `String`) y su sexo, que se definirá del tipo enumerado;

```
public static enum Sexo {F,M};
```

La clase incluirá:

- a) Un constructor con cuatro argumentos que proporcionan el nombre de la persona candidata, su DNI, su sexo y su partido político (este último podrá ser `null`, indicando que es un candidato o candidata independiente, sin partido). Añadir también un constructor con solo tres argumentos, en cuyo caso la persona se supondrá no afiliada; es decir, el atributo que representa el partido será `null`.
- b) Métodos para obtener el nombre de la persona, su DNI, su sexo, su partido y para saber si es independiente:

```
String getNombre()  
String getDNI()  
Sexo getSexo()  
String getPartido()  
boolean esIndependiente() // devuelve true si lo es
```

Y métodos para poder cambiar el nombre, el sexo y el grupo:

```
void setNombre(String nombreNuevo)  
void setPartido(String partido)  
void cambiarSexo()
```

En los dos primeros casos, el atributo correspondiente cambiará al que se pasa como argumento, y en el último, el sexo cambiará por el contrario al que estuviese definido.

- c) La representación textual de un candidato o candidata vendrá dada por su nombre, todo en mayúsculas. En caso de que la persona se presente como independiente (no esté asociado a ningún partido), se añadirá entre paréntesis la palabra "independiente".

- 2) **(6 puntos)** La clase `Candidatura` deberá incluir información privada sobre el nombre o siglas del partido que constituye una candidatura electoral (de tipo `String`), la circunscripción para la que se presenta esa candidatura (de tipo `String`) y una lista con los miembros de la candidatura (de la clase `Candidata`). En este caso, la clase incluirá:

- a) Un constructor con el nombre del partido y la circunscripción (por ejemplo, el municipio) de la candidatura como argumentos. Inicialmente, la lista de candidatos estará vacía.
- b) Métodos para obtener el partido y la circunscripción:

```
String getPartido()  
String getCircunscripcion()
```

- c) Un método para añadir un miembro a la candidatura:

```
void agregarMiembro(Candidata cand)
```

En caso de que exista ya algún candidato o candidata en la candidatura con el mismo DNI (independientemente de mayúsculas y minúsculas) que `cand`, la existente se sustituirá por esa nueva persona (`cand`). En otro caso, se añadirá al final de la lista. Para la implementación de este método se puede utilizar el método del apartado d).

- d) Un método para eliminar una persona de la candidatura:

```
void eliminarMiembro(Candidata cand)
```

Se eliminará el miembro de la lista cuyo DNI coincida (independientemente de mayúsculas y minúsculas) con el DNI de la persona que se pasa como argumento. En caso de que no exista ningún candidato o candidata con el mismo DNI en la candidatura,

se lanzará una excepción `RuntimeException` con un mensaje apropiado. Para la implementación de este método se puede utilizar el método del apartado d).

- e) Un método privado para obtener la posición en la lista de un candidato o una candidata con un DNI determinado:

```
int buscarDNI(String dni)
```

Devolverá la posición de la persona que tenga el DNI pasado como parámetro (independientemente de mayúsculas o minúsculas) o -1 en caso de no encontrar ninguna persona candidata con ese DNI.

- f) Métodos para añadir una cabeza de lista y para consultarla, respectivamente:

```
void agregarCabezaLista(Candidata cand)
```

```
Candidata cabezaLista()
```

La persona que encabece la lista se insertará en la primera posición. Si la lista está vacía, el segundo método deberá lanzar una excepción `RuntimeException` con un mensaje apropiado.

- g) Un método que devuelva el número de candidatos y candidatas independientes:

```
int numeroIndependientes()
```

- h) Un método para determinar si la lista cumple con los requisitos de una candidatura cremallera (es decir, que el sexo de las personas candidatas se va alternando):

```
boolean esCremallera()
```

- i) La representación textual de una candidatura vendrá dada por el nombre del partido, seguido de un guion ("-"), seguido de la circunscripción (en mayúsculas) y a continuación dos puntos (":"), seguido de la lista (entre corchetes, todos los miembros separados por comas). Por ejemplo:

Los 4 Fantásticos-NUEVA YORK:[MR. FANTÁSTICO, ANTORCHA HUMANA, MUJER INVISIBLE, LA COSA]

- 3) **(1,5 puntos)** La clase `CandidaturaPartido` representará candidaturas en las que todos los afiliados a un partido que no coincida con el partido de la candidatura serán considerados como independientes. La clase deberá reutilizar la clase anterior de forma que:

- a) Incluya un constructor con el nombre del partido y la circunscripción como argumentos, considerando una lista de miembros inicialmente vacía.

- b) Y garantice que cuando se añada un nuevo miembro o un miembro cabeza de lista, si el partido al que está afiliado no coincide (independientemente de mayúsculas y minúsculas) con el partido de la candidatura, se elimine esa afiliación para considerarlo un candidato o candidata independiente. Ello se consigue redefiniendo de forma adecuada los métodos:

```
void agregarMiembro(Candidata cand)
```

```
void agregarCabezaLista(Candidata cand)
```