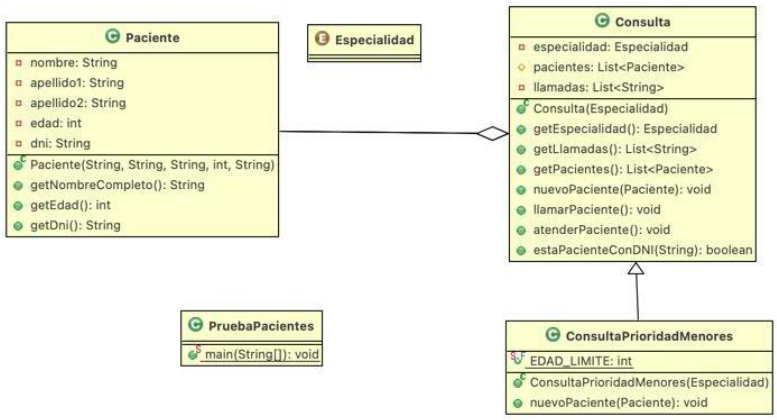
DEPARTAMENTO LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN	**PROGRAMACIÓN AVANZADA I** (Prueba del 7 de abril de 2025)	
APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA	

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 prPacientes

Se va a crear una aplicación para modelar consultas a pacientes en un servicio de salud. Para ello se crearán las clases `Paciente`, `Consulta`, `ConsultaPrioridadMenores` y `PruebaPacientes` y el enumerado `Especialidad` (con valores PRIMARIA, PEDIATRIA, OFTALMOLOGIA, TRAUMATOLOGIA y GINECOLOGIA). Todo ello en un paquete denominado `salud`.



1) **(2 puntos)** La clase `Paciente` mantiene información privada sobre pacientes de un centro de salud, incluyendo el nombre (de tipo `String`), el primer apellido (de tipo `String`), el segundo apellido (de tipo `String`), la edad (de tipo `int`) y el DNI (de tipo `String`). La clase incluirá:

a) Un constructor con cinco argumentos que proporcionan el nombre, primer apellido, segundo apellido, edad y DNI. Si la edad que se pasa es negativa se debe lanzar una excepción `RuntimeException` con un mensaje adecuado.
b) Métodos para obtener el nombre completo del paciente, la edad y el DNI:
```
String getNombreCompleto()    int getEdad()    String getDNI()
```
El nombre completo se compondrá uniendo el nombre y los apellidos, separados por un espacio en blanco.
c) La representación textual de un paciente vendrá dada por la inicial del nombre, seguida de la inicial del primer apellido, la inicial del segundo apellido, y los tres últimos dígitos del DNI[1] (antes de la letra). Por ejemplo, para un paciente con nombre completo "Pablo Ruiz Picasso", con DNI "12345678A", su representación textual es: "PRP678".

2) **(5 puntos)** La clase `Consulta` pretende representar consultas en un centro médico. Deberá incluir información privada sobre la especialidad de la consulta (de tipo `Especialidad`), una lista protegida de pacientes (`List<Paciente>`) esperando para ser atendidos y una lista privada de llamadas (`List<String>`), cada una de las cuales se corresponderá con la representación textual de los pacientes que van siendo llamados. En este caso, la clase incluirá:
a) Un constructor con un argumento que indique la especialidad de la consulta. Inicialmente, las listas de pacientes y de llamadas serán vacías.
b) Métodos para obtener la especialidad de la consulta, la lista de pacientes y la lista de llamadas:
```
Especialidad getEspecialidad()
List<Paciente> getPacientes()
List<String> getLlamadas()
```
c) Un método para añadir nuevos pacientes a la consulta, de forma que el último en ser añadido será el último en ser atendido:
```
void nuevoPaciente(Paciente paciente)
```
d) Un método para realizar la llamada de un paciente:
```
void llamarPaciente()
```
que añadirá al comienzo de la lista de llamadas la representación textual del primer paciente en la lista de pacientes, siempre que esta no esté vacía. Si la lista de pacientes está vacía, no se realizará llamada alguna.
e) Un método para atender al primer paciente de la lista de pacientes, siempre que haya sido llamado previamente (es decir, que esté el primero en la lista de llamadas):
```
void atenderPaciente()
```
Tras atenderlo, el paciente se elimina de la lista de pacientes (no así de la lista de llamadas). Si el paciente a atender no ha sido llamado, se lanzará una excepción de tipo `RuntimeException` con un mensaje apropiado. Si no hay pacientes que atender, no se hace nada.
f) Un método que determine si un paciente, cuyo DNI se proporciona como argumento, está entre los pacientes a atender.
```
boolean estaPacienteConDNI(String dni)
```
Téngase en cuenta que es indiferente que la letra del DNI esté en mayúsculas o minúsculas.
g) La representación textual de una consulta vendrá dada por el nombre de la especialidad, seguido de información (entre corchetes) sobre el número de pacientes en espera y el número de pacientes que se han llamado a la consulta, separados por un guion. Por ejemplo, si la especialidad es PRIMARIA, la lista de espera tiene tres pacientes y la lista de llamadas dos, la representación textual debe ser:
```
PRIMARIA [3 en espera – 2 atendidos]
```

3) **(2 puntos)** La clase `ConsultaPrioridadMenores` debe representar consultas en las que los menores de cierta edad (14 años, representada por una constante estática) sean atendidos con prioridad. Es decir, se situarán al principio de la lista de pacientes en espera, antes que los mayores de esa edad, pero después de los menores que ya estén en la lista de pacientes.

4) (1 punto) Implementar una clase de pruebas, siguiendo las instrucciones comentadas en la clase proporcionada `PruebaPacientes.java`.

---

[1] Utilícese el método `substring(int,int)` de la clase `String` para obtener la subcadena de una cadena de caracteres entre las posiciones indicadas en los argumentos.