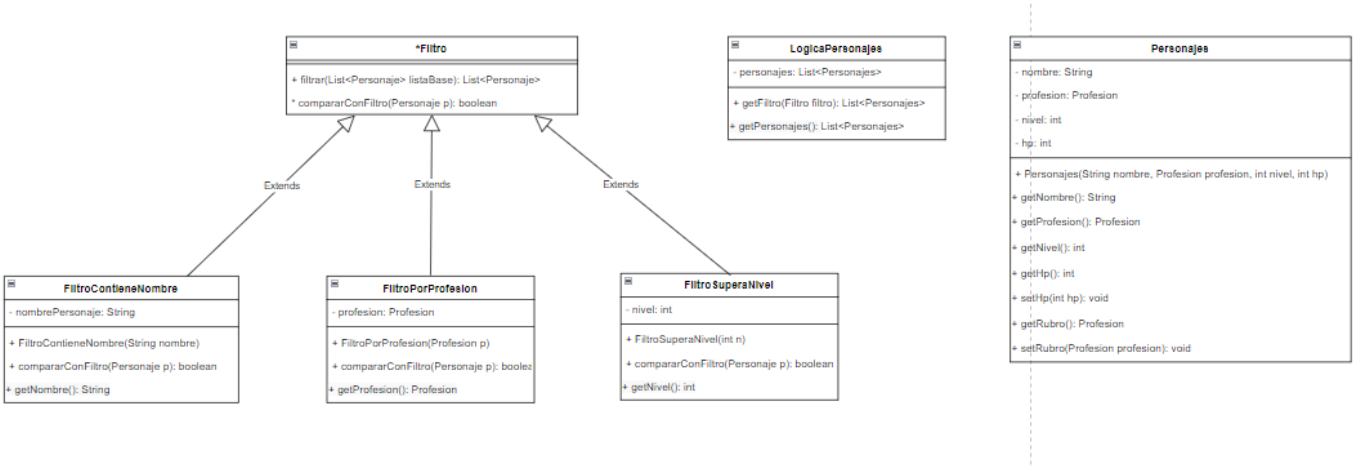
Actividad C

Los principios que no se cumplen son el Open Closed y Single Responsability, ya que si en un futuro queremos agregar un nuevo filtro deberíamos modificar la clase LógicaPersonaje agregando como atributo una nueva lista de tipo del nuevo filtro, además que en la clase FiltrosPersonaje hay un método distinto por cada filtro e igualmente que en el anterior si luego quisiéramos agregar un nuevo filtro, deberíamos modificar esa clase agregando otro método y no estaría cumpliendo los objetivos de los principios SOLID. Lo solucionaría de la siguiente manera: creando una nueva clase abstracta llamada “Filtro” en donde tendría un método llamado filtrar() y un método abstracto llamado compararConFiltro(). Como en todos los filtros tenían exactamente el mismo código exceptuando por la condición del if, lo que hice fue en el método filtrar() poner el código que tienen todos los filtros en común y en la parte que cambiaría llama al método abstracto el cual cada filtro lo implementa de acuerdo a lo que le corresponde.

Por otro lado, el segundo principio que no lo respeta ya que cada clase tiene más de una responsabilidad por ejemplo la clase LogicaPersonajes está manejando los filtroContieneNombre(), getPersonajesConNivelMayorA() y getPersonajesConProfesion(). Eso lo podríamos solucionar de la misma forma que se puede solucionar en Open Closed, es decir crear un único método getFiltro() en donde se encarga de retornar todos los filtros en vez de tener un método por cada filtro.

El diagrama de clases quedaría de la siguiente manera:



Consideraciones de implementación

- ♥ En las clases de cada filtro, como atributo tienen lo que se desea comparar ya que al luego implementar el método compararFiltro() y como parámetro está el personaje al cual se desea comparar se puede resolver más fácilmente