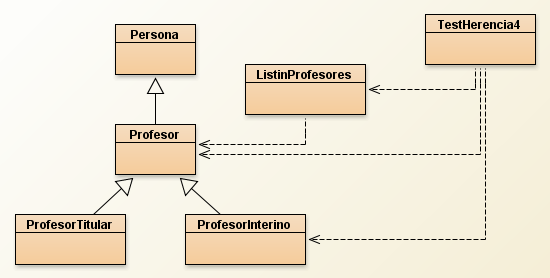
**HERENCIA Y POLIMORFISMO**

Analiza el siguiente diagrama de clases. En él se pueden observar relaciones de herencia y relaciones de uso:



Trata de definir el código de las clases, estableciendo las relaciones de herencia y uso entre ellas. Trata de crear una clase con el método main (TestHerencia4) donde de alguna manera crees objetos de los distintos tipos y hagas uso de ellos, por ejemplo crea profesores interinos y titulares y luego recórrelos con un for extendido donde el tipo sea Profesor (uso del polimorfismo). Luego compáralo con las explicaciones y soluciones que damos a continuación.

En la solución que hemos planteado nosotros, en el tipo Profesor hemos incluido un método denominado mostrarDatos() que muestra los datos propios de un objeto Profesor. Luego, en las subclases ProfesorInterino y ProfesorTitular hemos sobreescrito el método mostrarDatos() de modo que en este caso únicamente muestra los datos específicos de los subtipos.

Por último, en la clase ListinProfesores simulamos un listín que admite todo tipo de profesores mediante un ArrayList que usa objetos de tipo Profesor, y que permite listar los profesores mediante un método listar() que lo que hace es invocar el método mostrarDatos() de los objetos contenidos en la lista. Si el método utilizado se basara en el tipo declarado en el código, listar() siempre nos devolvería los datos de los objetos Profesor. Sin embargo, como veremos, esto no es así: cuando la variable apunta a un subtipo, el método invocado en tiempo de ejecución es el propio del subtipo, mientras que cuando la variable apunta a un tipo sí se invoca el método propio del tipo. Por eso decimos que Java hace una búsqueda dinámica del método: el método que se usa depende del tipo dinámico del objeto.