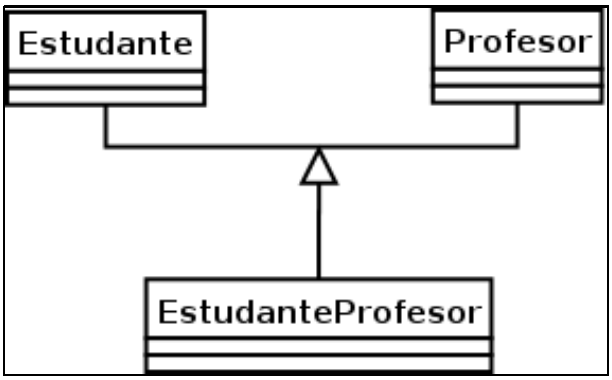
1 Herdanza múltiple

1.1 Herdanza múltiple

A herdanza múltiple é unha propiedade da programación orientada a obxectos que permite definir unha nova clase a partir de dúas ou máis superclases á vez.

Por exemplo, se temos unha clase **Estudante** e outra clase **Profesor**, unha terceira clase, chamada **EstudanteProfesor**, que podería herdar de Estudante e Profesor, xa que pode ser un becario contratado polo departamento que imparta clases.



Neste caso, se as dúas superclases teñen un atributo que se chame `nome` e un getter e un setter para el, o becario estudante ten agora dous nomes e dous métodos para lelo e escribilo. Prodúcese o que se coñece como **colisión**.

Este é un problema típico da herdanza múltiple que pode provocar problemas á hora de herdar de dous ou máis superclases atributos e métodos que se chamen igual nas superclases. As linguaxes de programación e os compiladores que permiten a herdanza múltiple directa deben dar solución a este problema.

Java non soporta herdanza múltiple directa pero pode simulala mediante outros mecanismos que son as **interfaces** e a **delegación**, evitando os problemas da colisión.

Unha interface é unha clase especial onde todos os seus métodos son abstractos, é dicir, non están implementados. Cando unha clase inclúe unha interface dise que o está implementando e significa que na clase teñen que aparecer implementados os métodos que hai declarados na interface. Noutras palabras, se tes unha clase que implementa unha (ou varias) interface, esta interface representa a API do teu código.

Dende o punto de vista do deseño as interfaces permiten ocultar detalles da implementación, de tal xeito que podemos reescribir o código do noso método sen forzar ningún cambio no código de terceiros que o usan. Unha clase pode implementar máis dun interface á vez. A sintaxe para facelo é a seguinte:

```
class UnhaClase extends SuperClase implements Interface1, Interface2{ }
```

Para implementar en Java o exemplo anterior temos que facer que a clase EstudanteProfesor herde directamente da clase Profesor e implemente unha interface que describa á clase Estudante. Facelo ao revés sería igualmente válido. Así pois, haberá que declarar unha interface chamada IEstudante onde se definan exactamente os mesmos métodos que se definen na clase Estudante. Acto seguido, faise que a clase EstudanteProfesor herde directamente da clase Profesor e implemente a interface IEstudante. Deste xeito obrigamos a implementar os mesmos métodos que forman a clase Estudante na clase EstudanteProfesor:

```
class Estudante {
  // Atributos e métodos
}

class Profesor {
  // Atributos e métodos
}

interface IEstudante {
  // Os mesmos método que teña a clase Estudante
}
class EstudanteProfesor extends Profesor implements IEstudante {
}
```

Con respecto á implemenación dos métodos da clase Estudante, utilízase o que se coñece como delegación ou contención, e que consiste en delegar noutra clase a implementación dos métodos, co fin de evitar ter o mesmo código duplicado en diferentes lugares. Isto conséguese declarando como atributo privado da clase EstudanteProfesor un obxecto de tipo Estudante, de maneira que cando se implementen os métodos da interface na clase EstudanteProfesor o que se fará é invocar aos métodos correspondentes do obxecto de tipo Estudante. Deste xeito conseguimos facer (de feito simúlase) que a clase EstudanteProfesor herde da clase Profesor e da clase Estudante.