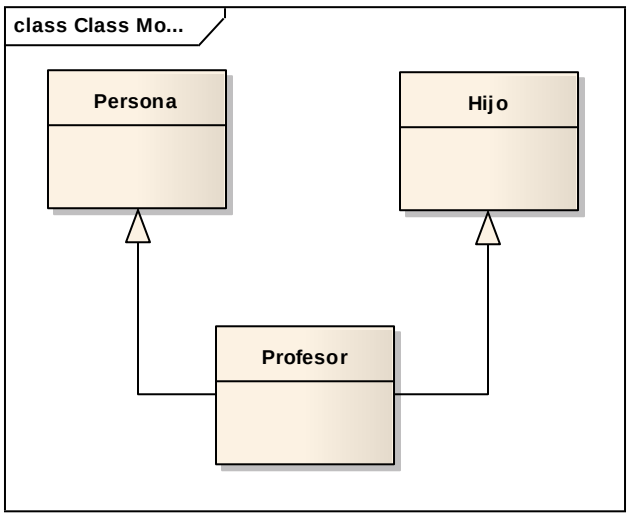
UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS



Herencia múltiple.

2013
Transversal Programación Orientada a Objetos
Proyecto Curricular de Ingeniería de Sistemas

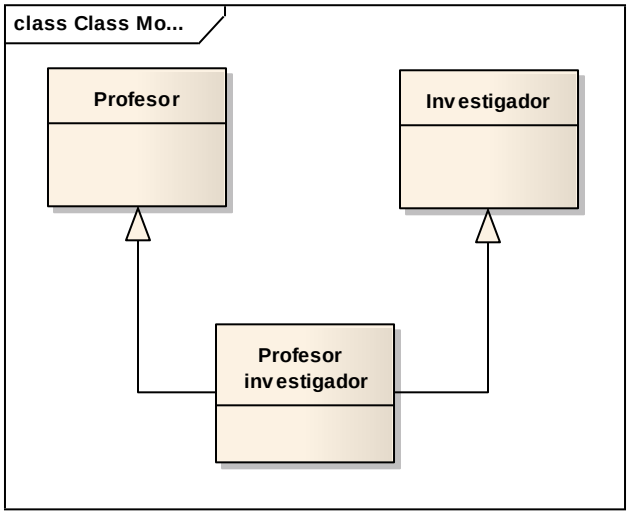
Introducción

Existen casos donde las relaciones entre clases pueden generar una herencia de parte de dos clase base, a eso le denominamos herencia múltiple, ¿Qué problemas acarrea?

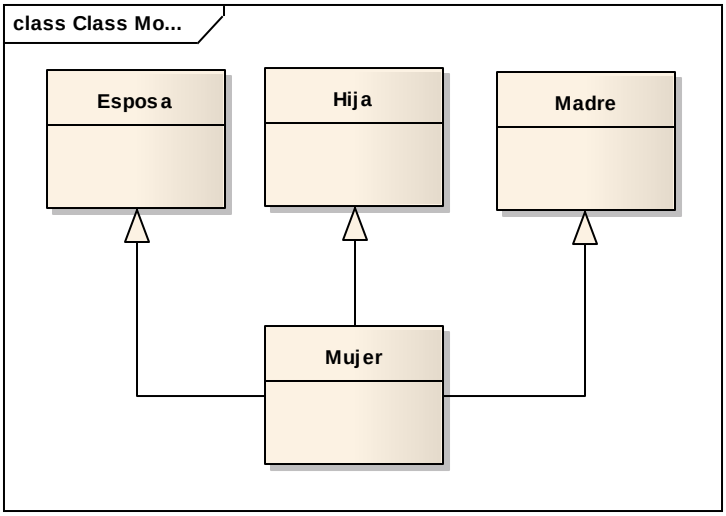
Puede darse a la hora de programar algún caso donde se requiera que una clase tenga características provenientes de dos clases como es el caso de un profesor, el estudiante ante todo es un ciudadano y tendrá ciertas características que afecten su comportamiento con respecto a esta condición, pero también es hijo de alguien y esto puede incurrir en otras características adicionales. A esto se le denomina herencia múltiple, lo cual acarrea diversos problemas y para lo cual existen unas cuantas soluciones (dentro de ellas las interfaces). Más adelante se observará diversos ejemplos de herencia múltiple, los inconvenientes, soluciones y qué alternativas presentan los lenguajes cuando se presenta.

1. Herencia múltiple

La definición es concisa, se trata de si se puede heredar de más de una clase, pero ¿cómo surgió esa necesidad? Smalltalk fue un lenguaje popular en objetos, antecesor de C++, donde solo se consideró la herencia simple, empezando por object. Allí no se podía crear una clase sin una existente, así que C++ propone el uso de la herencia múltiple (aunque no se dieran tantos escenarios donde fuese necesario). Básicamente se involucra con los contenedores, donde para el caso de C++, un fabricante de objetos construía una jerarquía con un contenedor Holder, y otro fabricante da otra jerarquía de clases donde se encuentra BitImage, suponiendo que se requieran los dos, requeriría que existiese la herencia múltiple.
Veamos algunos ejemplos de herencia múltiple:



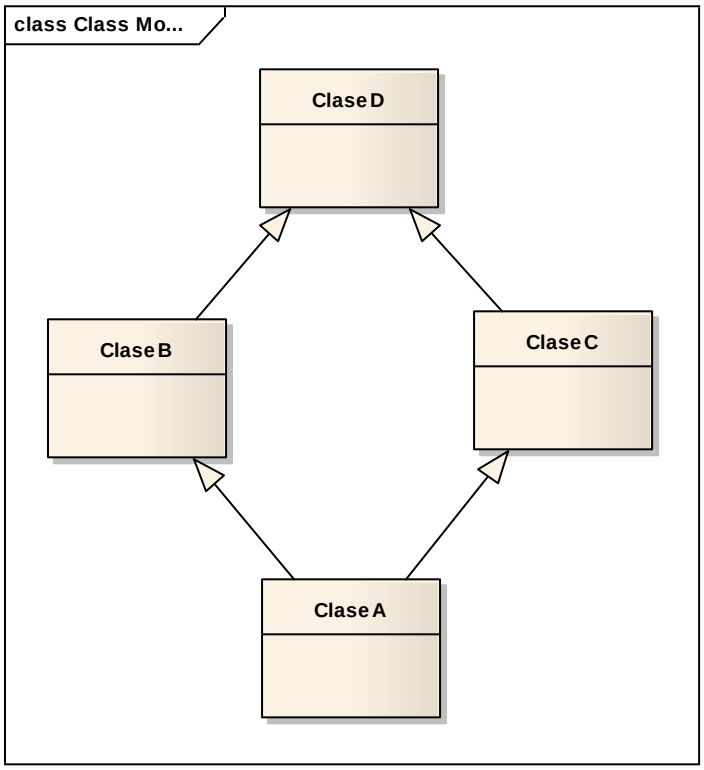
Fuente: Autor.



Fuente:Autor

2. Problemas que genera

- **Problema del diamante:** Si dos clases heredan de una tercera y una cuarta clase tiene como padre a las dos iniciales, como se observa en la figura (formando un diamante), cuando una instancia de la clase A, llame un método de la clase D , desde donde lo hereda, ¿desde la clase B o C?. Cada lenguaje presenta mecanismos para responder a este caso.



- **Colisiones:** En la herencia múltiple pueden coincidir el nombre de los atributos y de los métodos entre las clases base. En el caso de Java establece unas reglas al respecto:
  - ♣ **Si se repite un atributo:** Obliga al programador a especificar de qué interfaz base lo va a utilizar.
  - ♣ **Si se repiten nombres en los métodos:** Se dan ciertas condiciones: Si tienen diferentes parámetros se produce sobrecarga de los métodos, así existen diversas alternativas para llamar al método, si solo cambia el valor que devuelve se da un error de compilación indicando que no se pueden implementar simultáneamente y si coinciden en sus declaraciones se elimina uno de los dos.

3. Qué pasa con los lenguajes.

En vista de todos los inconvenientes que puede generar los lenguajes han desarrollado diversas estrategias para admitirlo (en el caso de los que no los implementan) y otros han sido diseñados para soportarla:

Lenguaje	Mecanismo
Java	No soporta herencia múltiple. Se emplean las interfaces como mecanismo de solución.
Python	Soporta la herencia múltiple simil a C++. Simplemente se emplea la sintaxis class ClaseDerivada(ClaseBaseUno,ClaseBAseDos): Responde al problema del diamante, tomando toda clase descendiente de object. Crea una lista de clases (de derecha a izquierda y de abajo hacia arriba), luego va eliminando toda repetición menos la de la última clase, generando un orden.
Ruby	No soporta la herencia múltiple, emplea los mixins como mecanismo de solución.
C++	Soporta herencia múltiple. La sintaxis empleada es class ClaseHija: public ClasePadreA [,ClasePadreB]. Utiliza la herencia virtual en caso de tener elementos repetidos entre las clases base, esto permite que cada objeto de la clase derivada no contenga todos los objetos de la clase base si están duplicados.

Bibliografía

- Curso de Java: Capsula formativa. Tomado en línea de : http://www2.uah.es/jcaceres/capsulas/java_interfaces.pdf
- Multiple Inheritance. Tomado en línea de: http://docs.python.org/release/1.5.1p1/tut/multiple.html