Introducción a la programación con C#

Herencia, interfaces, polimorfismo

Tipos de relaciones

Relación de herencia:

- Es cuando un objeto o clase se basa en otro objeto o clase, usando la misma implementación o comportamiento

Relación de composición:

- Composición quiere decir que tenemos una instancia de una clase que contiene instancias de otras clases que implementan las funciones deseadas

¿Qué es la herencia?

La herencia, en todos los ámbitos, tiene connotaciones de transmisión.

En Programación Orientada de Objetos, es la transmisión de los métodos y atributos de una clase **(clase base)** a otra clase **(clase derivada)**.

Clase base es la clase que transmite la herencia.

Clase derivada es la clase que recibe la transmisión.

La clase **derivada hereda** de la clase **base**.

Tipos de relación de herencia

Herencia simple:

- Cuando una clase derivada hereda de una única clase base

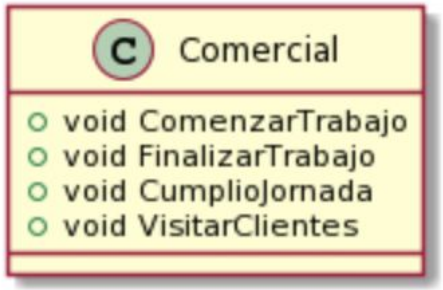
Herencia múltiple:

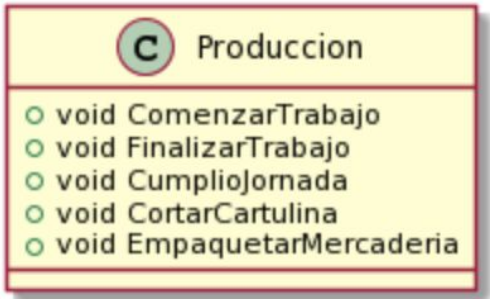
- Cuando una clase derivada hereda de varias clases base

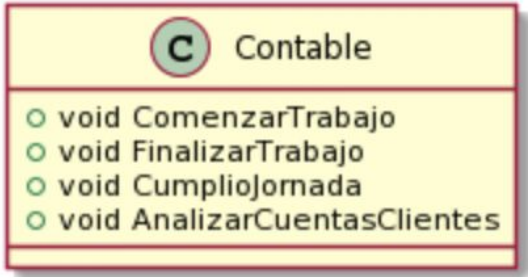
¿Porque aplicar herencia?

- Nos permite reutilizar código de una manera rápida, sencilla y evidente
- Nos permite crear objetos polimórficos
- Nos permite generalizar y clasificar

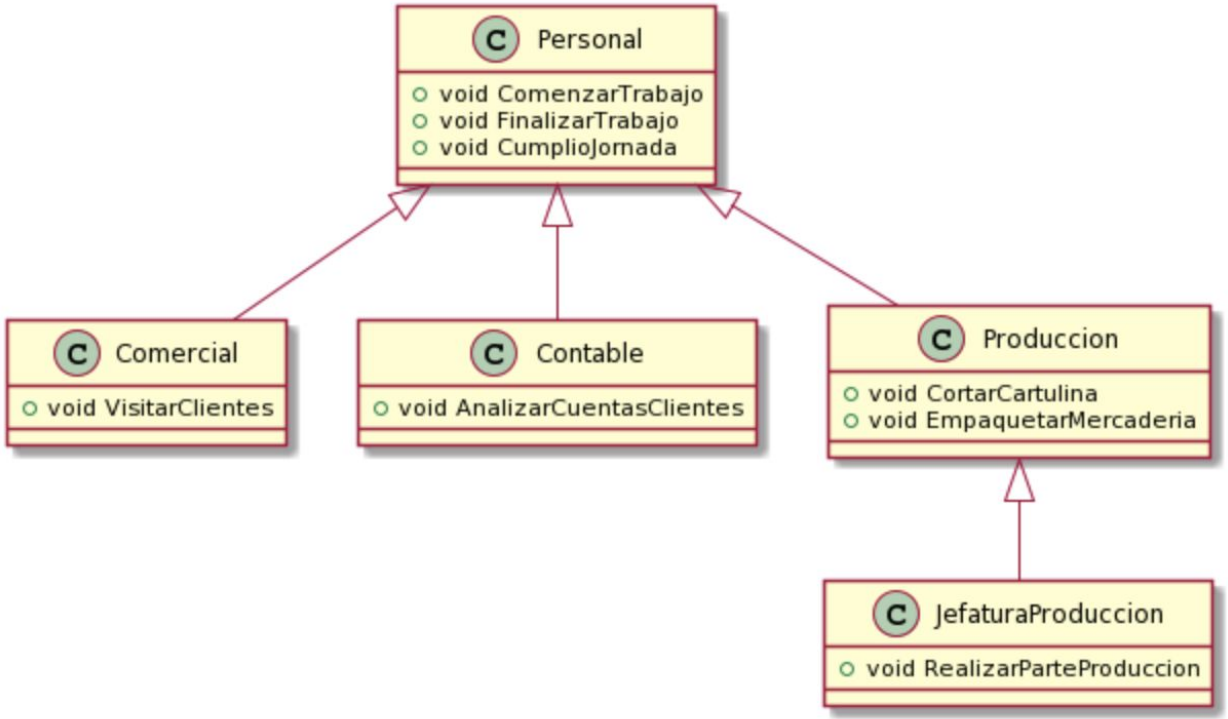
Pero entonces... ¿Qué es una herencia?







Pero entonces... ¿Qué es una herencia?



Clasificación y generalización

La mente humana clasifica los conceptos de acuerdo a dos dimensiones:

- Pertenencia **(TIENE-UN)**
- Variedad **(ES-UN)**

La herencia consigue clasificar los tipos de datos **(abstracciones)** por variedad, acercándose al modo de razonar humano.

Esto da lugar a jerarquías de **generalización** y **especialización**.

La implementación de estas jerarquías en un lenguaje de programación da lugar a jerarquías de herencia.

Implicancia sobre los objetos

Los objetos de la clase padre NO sufren ninguna alteración por la presencia de clases derivadas.

Los objetos de la clase hija:

- Tienen todos los atributos transmitidos desde la clase padre junto con los atributos añadidos en la clase hija
- Responden a mensajes que corresponden con los métodos públicos transmitidos desde la clase padre junto con los métodos públicos añadidos en la clase derivada

Clases abstractas

Son clases NO instanciables que surgen del factor común del código de otras clases con atributos comunes, métodos comunes y/o cabeceras de métodos comunes sin definición **(misma firma de método)**.

Una clase abstracta puede ser hija de una clase concreta si en su especialización añade algún método abstracto.

Una clase abstracta puede ser hija de otra clase abstracta porque se especializa (añadiendo atributos y/o métodos y/o redefiniendo métodos) pero NO redefine todos los métodos.

Interface

Son clases abstractas puras que no contienen ningún atributo ni la definición de ningún método, sólo contienen métodos abstractos.

Un **interface** NO puede heredar de una **clase**, pero SI puede heredar de otro **interface** por extensión.

Una **clase** SI puede heredar de una **clase** por extensión o de un **interface** por implementación.

La herencia por **extensión** NO disfruta de herencia múltiple, pero la herencia por **implementación** SI disfruta de herencia múltiple.

Polimorfismo

El **polimorfismo** se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos.

El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.

¡¡A trabajar!!