Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

“UPIITA”

Ingeniería Mecatrónica

Asignatura: Programación Avanzada

“Tema 4: Polimorfismos”

Profesora: Maza Casas Lamberto

Grupo: 2MV3

Alumnos:

Alonso Bernal Alejandro

López Piedracruz Marcos Antonio

Luviano Murakawa Tsuioshi Alberto

**Introducción**

En programación orientada a objetos se denomina polimorfismo a la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación. Un objeto polimórfico es una entidad que puede contener valores de diferentes tipos durante la ejecución del programa.

En algunos lenguajes, el término polimorfismo es también conocido como ‘Sobrecarga de parámetros’ ya que las características de los objetos permiten aceptar distintos parámetros para un mismo método generalmente con comportamientos distintos e independientes para cada una de ellas.

En general, hay tres tipos de polimorfismos: el polimorfismo de sobre carga, el polimorfismo paramétrico y el polimorfismo de inclusión.

**Polimorfismo de sobrecarga:**

El polimorfismo de sobrecarga ocurre cuando las funciones del mismo nombre existen, con función similar, en clases que son completamente independientes unas de otras. El polimorfismo de sobrecarga nos permite definir operadores cuyos comportamientos varían de acuerdo a los parámetros que se les aplican.

**Polimorfismo paramétrico:**

El polimorfismo paramétrico es la capacidad para definir varias funciones utilizando el mismo nombre, pero usando parámetros diferentes. El polimorfismo paramétrico selecciona automáticamente el método correcto a aplicar en función del tipo de datos pasados en el parámetro.

Por lo tanto, podemos definir varios métodos homónimos de **addition()** efectuando uan suma de valores. El método **int adddition(int,int)** devolvería la suma de dos números enteros. Por su parte, **float addition(float,float)** devolvería la suma de dos flotantes. En cuanto a **char addition(char,char)** daría por resultado la suma de dos caracteres definidos por el autor.

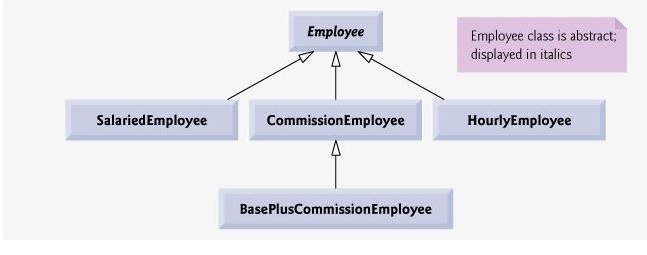
Una *signature* es el nombre y tipo (estático) que se da a los argumetos de una función. Por esto, una firma de método determina qué elemento se va a llamar.

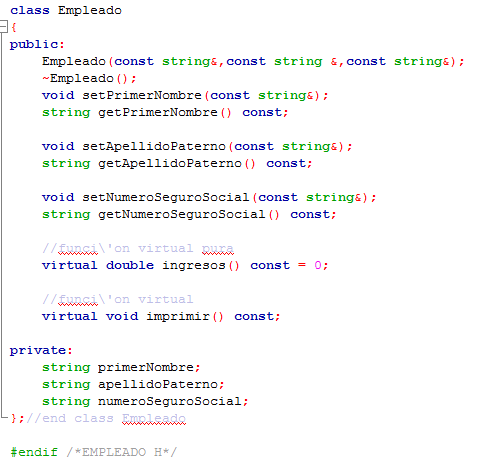
**Polimorfismo de subtipado:**

La habilidad para redefinir un método en clases que se hereda de una clase base se llama *especialización*. Por lo tanto, se puede llamar un método de objeto sin tener que conocer su tipo intrínseco: esto es *polimorfismo de subtipado*. Permite no tomar en cuenta detalles de las clases especializados de una familia de objetos, enmascarándolos con una interfaz común (siendo esta la clase básica).

**Desarrollo:**

Para el desarrollo del tema “polimorfismo” se diseñó un proyecto llamado Empleado, dentro de este proyecto, en el cual, a partir de una clase abstracta llamada “Empleado” con métodos para imprimir sus datos (nombre, apellido, NSS, etc.) y para definir su salario, se definen otro tipo de clases con los mismos nombres para sus métodos específicos (aplicando el concepto de polimorfismo) definiendo de esta manera distintas formas para calcular el salario de un empleado.   
El diagrama general de todo el proyecto es el siguiente:

El código principal del cual se derivan todos los tipos de empleado se muestra en la imagen siguiente: 



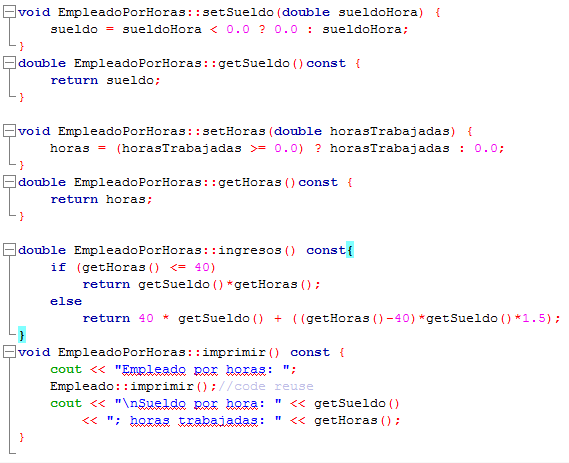
A continuación se procederá a mostrar todas las maneras en las que los métodos de la clase Empleado se ven sobrecargados de este modo aplicando el concepto de polimorfismo:



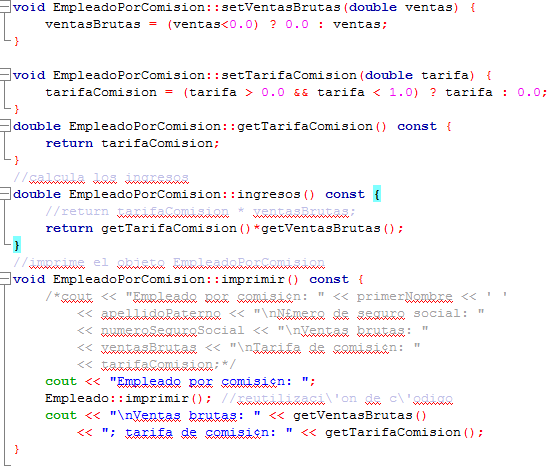
*Empleado asalariado*



*Empleado base mas comision*



*Empleado por horas*



*Empleado por horas*

Como se puede apreciar en las imágenes anteriores todos los tipos de empleado existentes comparten un método con un mismo nombre pero dependiendo de la clase que se esté trabajando, el método ejecutara una acción diferente.

**Conclusión**

Al ocupar los polimorfismos es más fácil utilizar una variable para diferentes propósitos, como se explicó anteriormente en el reporte, estos pueden ser variables de mismo nombre y funciones similares, para definir una variable con un nombre que puede ser utilizados en diferentes funciones y el que nos ayuda con la herencia, es decir, que no toma en cuenta los detalles de la clase heredada.