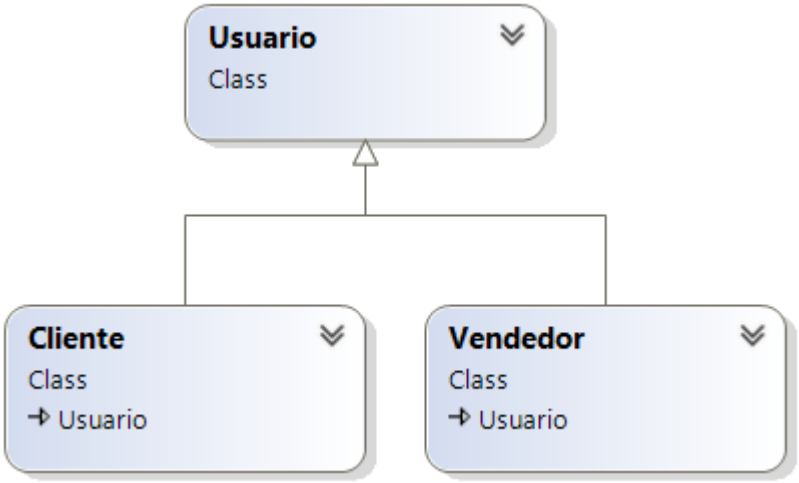
Universidad Católica de El Salvador
Facultad de Ingeniería y Arquitectura
Materia: Programación II
Docente: Master Giovanni Acosta

Tema 10:
Herencia en programación orientada a objetos



Objetivos:

- Conocer cómo la herencia promueve la reutilización de código
- Diseñar la jerarquía de clases
- Construir la clases base y las clases derivadas
- Utilizar los constructores en las jerarquías de herencia

¿QUÉ ES LA HERENCIA?

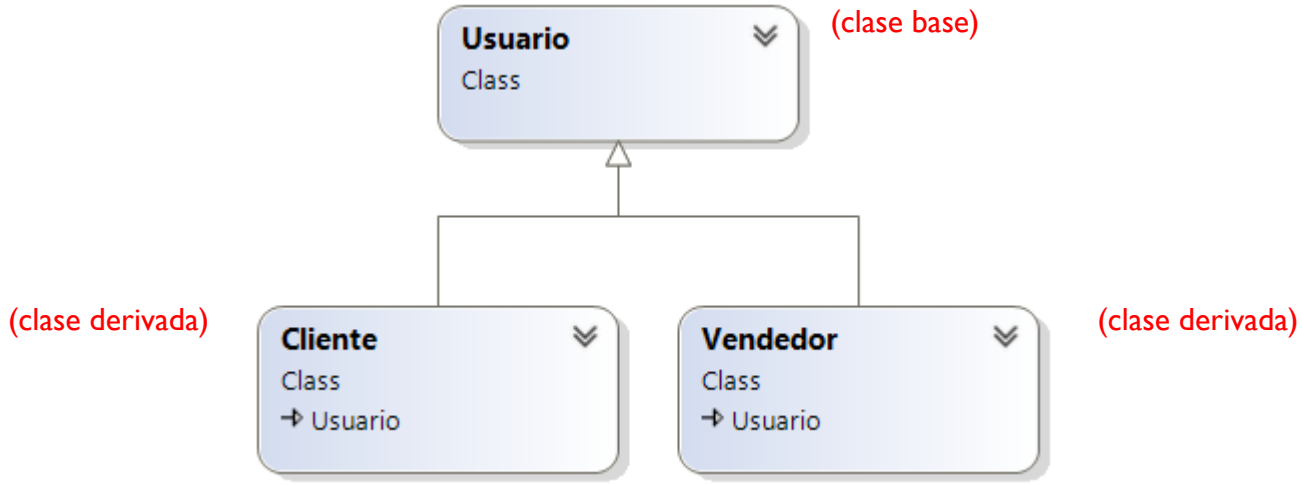
- En **Derecho**, la herencia es el acto jurídico mediante el cual una persona que fallece transmite sus bienes, derechos y obligaciones a otra u otras personas, que se denominan herederos.

- En **Genética**, la herencia es el conjunto de cromosomas y de recetas contenidas en ellos, que heredamos de nuestros padres y que a su vez heredaremos a nuestros hijos.

¿QUÉ ES LA HERENCIA EN PROGRAMACIÓN?

- En **Programación Orientada a Objetos**, la herencia es un mecanismo de **reutilización de código** en el que se crea una nueva clase al absorber los elementos (campos, propiedades y métodos) de una existente, y se pueden mejorar con nuevas capacidades, o con modificaciones en las capacidades ya existentes.

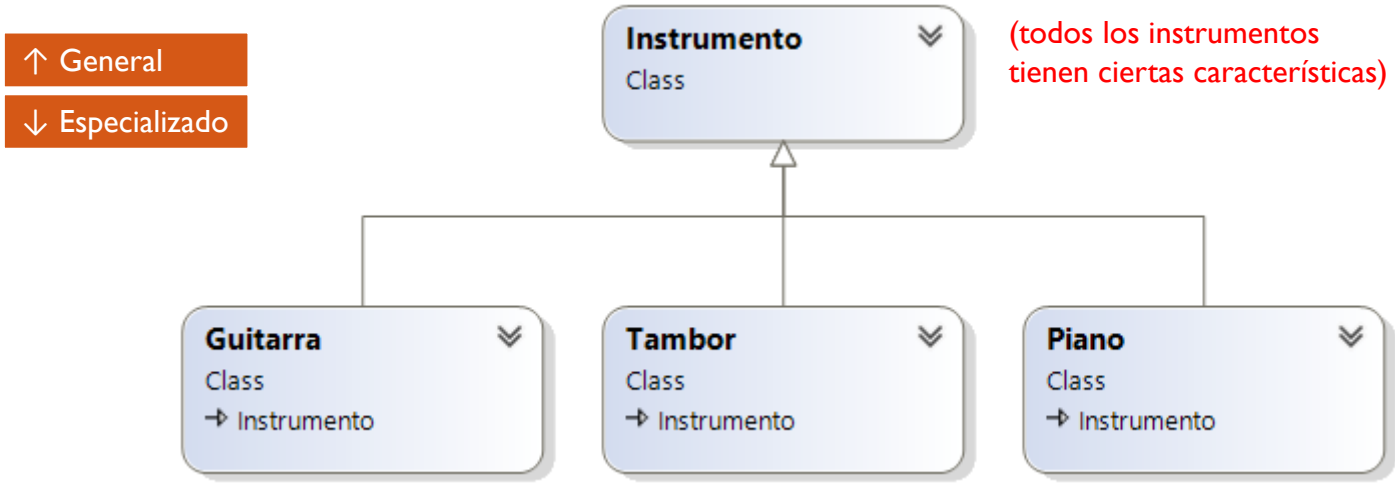
- La herencia es una relación entre:
  - Una **clase base**: clase más general, algunas veces llamada *superclase*
  - Y una o varias **clases derivadas**: clase más especializada, algunas veces llamada *subclase*



GENERALIZACIÓN Y ESPECIALIZACIÓN

- En el mundo real, puede encontrar muchos objetos que son **versiones especializadas de otros objetos más generales**. Por ejemplo, debido a que guitarra, tambor y piano son instrumentos musicales, tienen todas las características generales de un instrumento musical. Además, tienen características propias según su tipo.

- En el mundo de la programación orientada a objetos, cada uno de estos instrumentos es una clase derivada de la clase Instrumento, al ser Instrumento la clase base de estas clases.



(además de las características comunes de los instrumentos, cada instrumento tiene sus propias características únicas)

HERENCIA Y LA RELACIÓN “ES UN(A)” (“is a” en inglés)

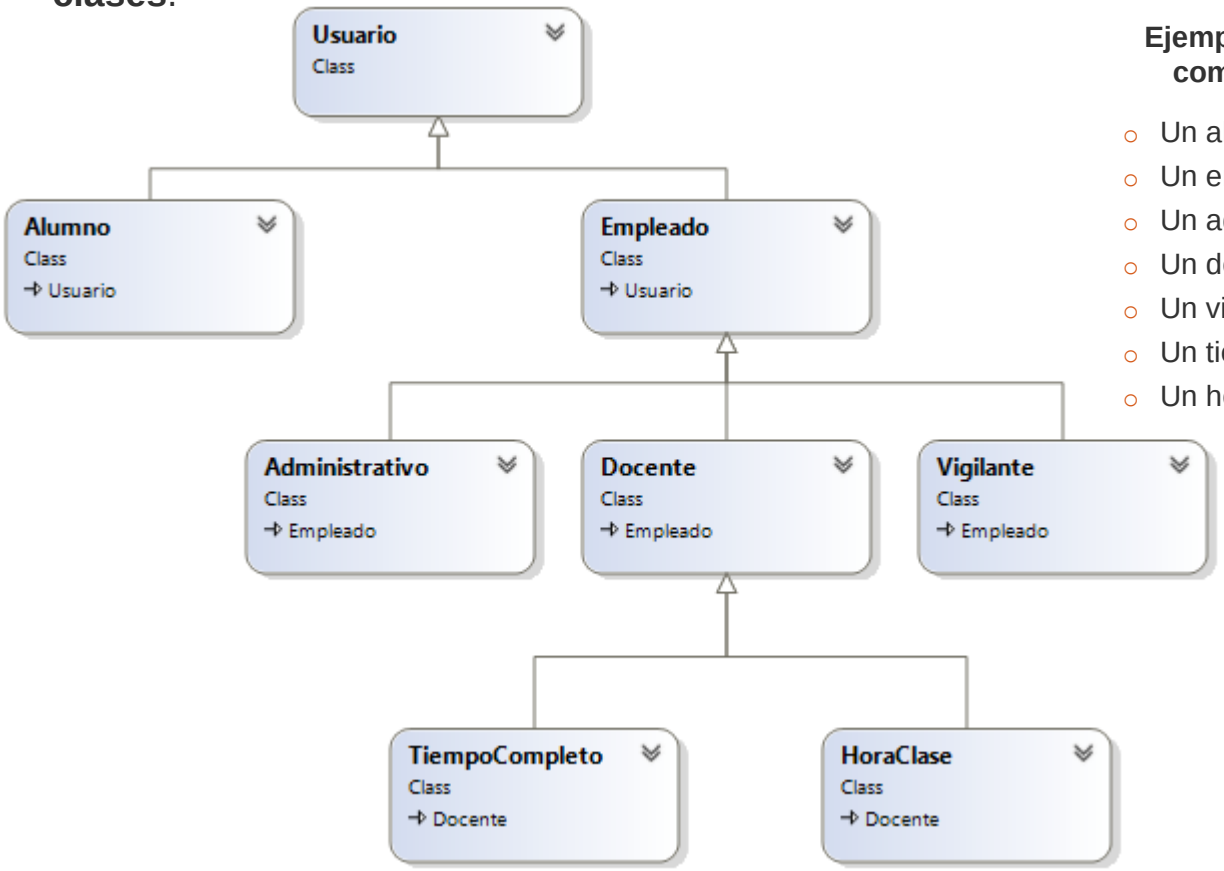
- Cuando un objeto es una versión especializada de otro objeto, existe una relación “Is a" entre ellos.

- Ejemplos:
  - Un rottweiler **es un** perro
  - Un auto **es un** transporte
  - Una flor **es una** planta
  - Un rectángulo **es una** forma

- La herencia implica una clase base y una clase derivada. **La clase base es la clase general y la clase derivada es la clase especializada**. Se puede pensar en la clase derivada como una versión extendida de la clase base. La clase derivada hereda campos, propiedades y métodos de la clase base sin que ninguno de ellos deba reescribirse. Además, se pueden agregar nuevos campos, propiedades y métodos a la clase derivada, y eso es lo que la convierte en una versión especializada de la clase base.

JERARQUÍA DE CLASES

- Una clase derivada puede así mismo ser una clase base de otra clase, y así sucesivamente. El conjunto de clases así definido da lugar a una **jerarquía de clases**. Cuando cada clase derivada lo es de una sola clase base, la estructura jerárquica recibe el nombre de **árbol de clases**.



Ejemplo: Jerarquía de clases comunidad universitaria.

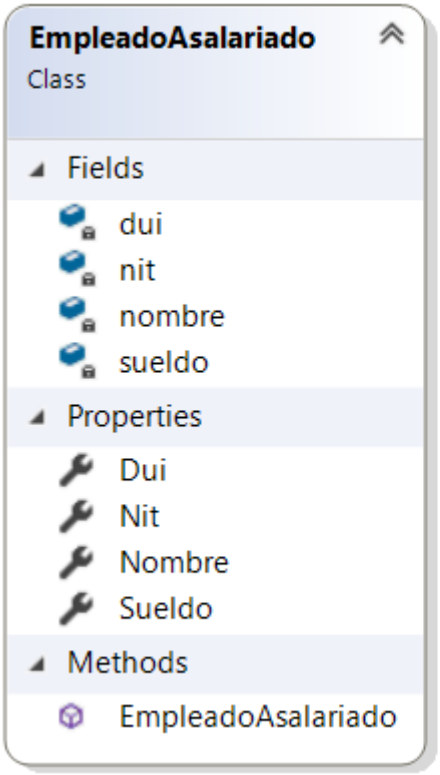
- Un alumno es un usuario
- Un empleado es un usuario
- Un administrativo es un empleado
- Un docente es un empleado
- Un vigilante es un empleado
- Un tiempoCompleto es un docente
- Un horaClase es un docente

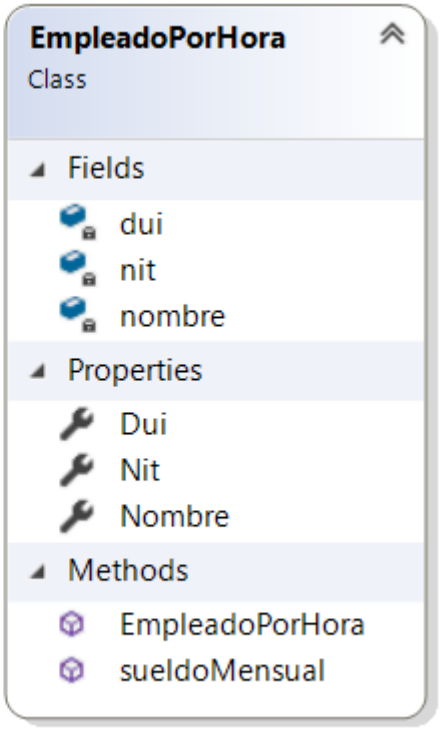
EJERCICIO:

- Diseñar la **jerarquía de clases** para la publicación de: Libros, revistas, periódicos y artículos de una editorial.

¿POR QUÉ USAR HERENCIA EN POO?

- Analizar las siguientes clases:





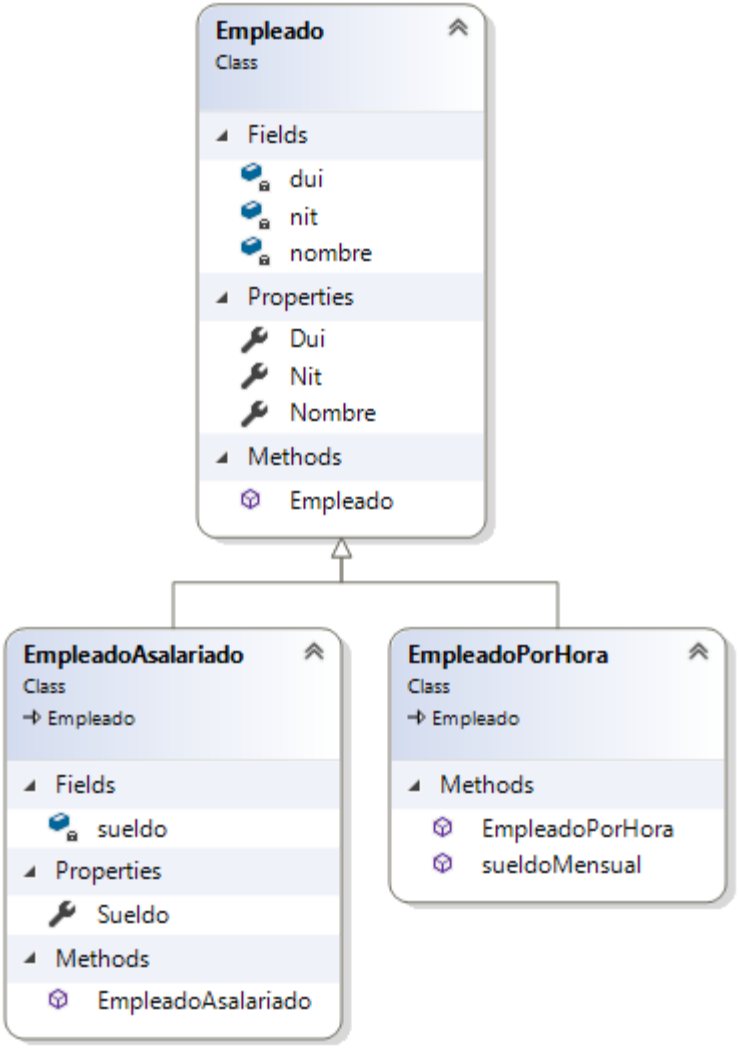
- El empleado asalariado: es el que posee un sueldo fijo mensual.
- El empleado por hora: el sueldo mensual depende de la cantidad de horas trabajadas y del valor de la hora.

¿Poseen código repetitivo?

¿Qué elementos tienen en común?

¿Cómo evitar el código repetitivo?

¿CUÁL ES LA SOLUCIÓN?



- Solución **herencia**: relación generalización / especialización
- Las características generales se establecen en la clase base Empleado.
- Las características particulares se establecen en cada clase derivada.
- La clase derivada EmpleadoAsalariado posee un sueldo.
- La clase derivada EmpleadoPorHora posee un método para calcular el sueldo mensual, según las horas trabajadas y el valor de la hora.

La Herencia: evita el código repetitivo y permite extender las características de las clases derivadas

¿CÓMO CREAR UNA CLASE DERIVADA EN C#?

- Sintaxis para crear una clase derivada.

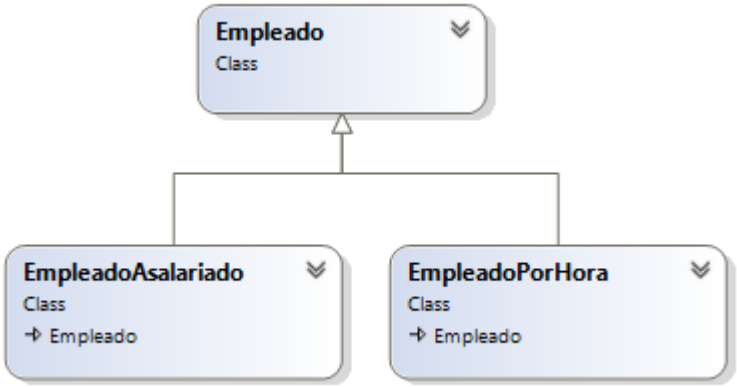
```
class ClaseDerivada : ClaseBase
{
    //elementos heredados de la clase base
    //agregar aquí los elementos nuevos de la clase derivada
}
```

- Ejemplo:

```
class EmpleadoAsalariado : Empleado
{
    //elementos heredados de la clase Empleado
    //agregar aquí los elementos nuevos de EmpleadoAsalariado
}
```

DEMO: EMPLEADOS

- **Clase base** Empleado:

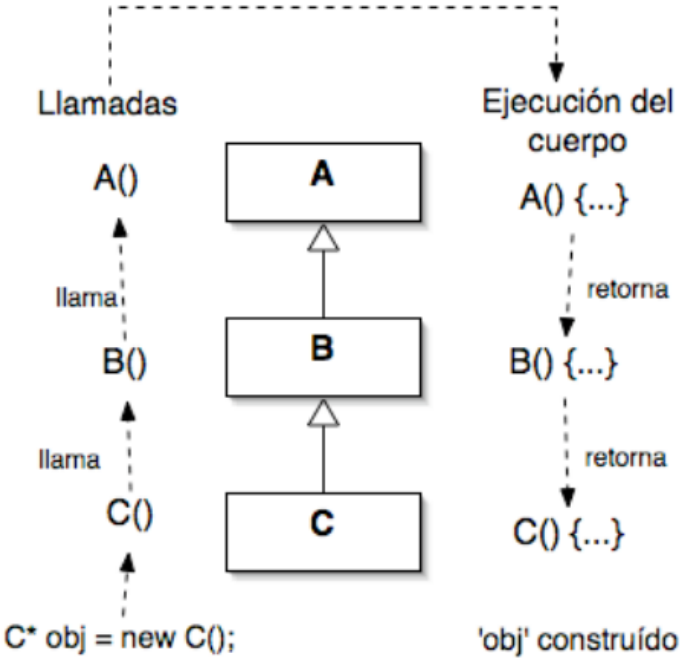



```
class Empleado
{
    private string nombre;
    private string dui;
    private string nit;

    2 references
    public Empleado(string nombre, string dui, string nit)
    {
        this.nombre = nombre;
        this.dui = dui;
        this.nit = nit;
    }

    0 references
    public string Nombre { get => nombre; set => nombre = value; }
    0 references
    public string Dui { get => dui; set => dui = value; }
    0 references
    public string Nit { get => nit; set => nit = value; }
}
```

LOS CONSTRUCTORES NO SE HEREDAN

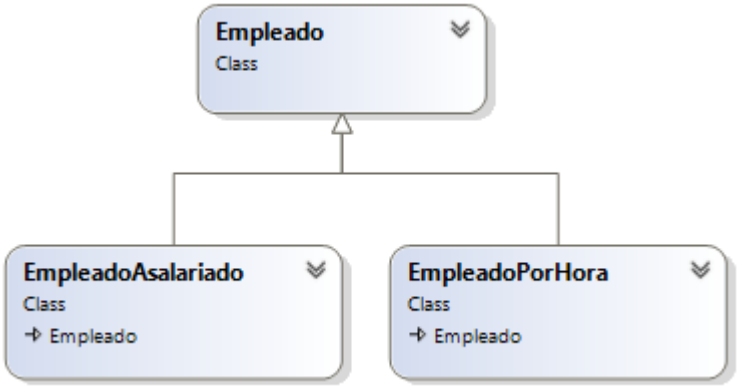
- Cada clase debe definir sus propios constructores.

- **Creación de un objeto de clase derivada:**
  - Se invoca a todos los constructores de la jerarquía.
  - Orden de ejecución de constructores: primero se ejecuta el constructor de la clase base y luego el de la derivada.



DEMO: EMPLEADOS (CONT.)

- **Clase derivada** EmpleadoAsalariado:

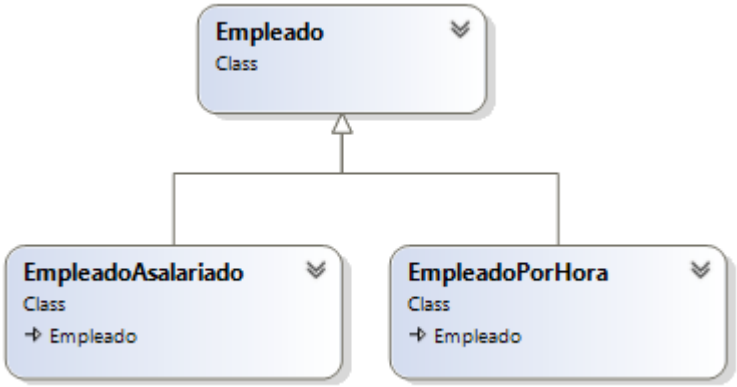



```
class EmpleadoAsalariado : Empleado
{

    private decimal sueldo;

    0 references
    public EmpleadoAsalariado(string nombre, string dui, string nit, decimal sueldo)
        : base(nombre, dui, nit)
    {
        this.sueldo = sueldo;
    }

    0 references
    public decimal Sueldo { get => sueldo; set => sueldo = value; }
}
```

La palabra clave **base** se utiliza para obtener acceso a los elementos de la clase base desde una clase derivada
En este ejemplo base hace referencia al método constructor de la clase Empleado pasándole los argumentos

DEMO: EMPLEADOS (CONT.)

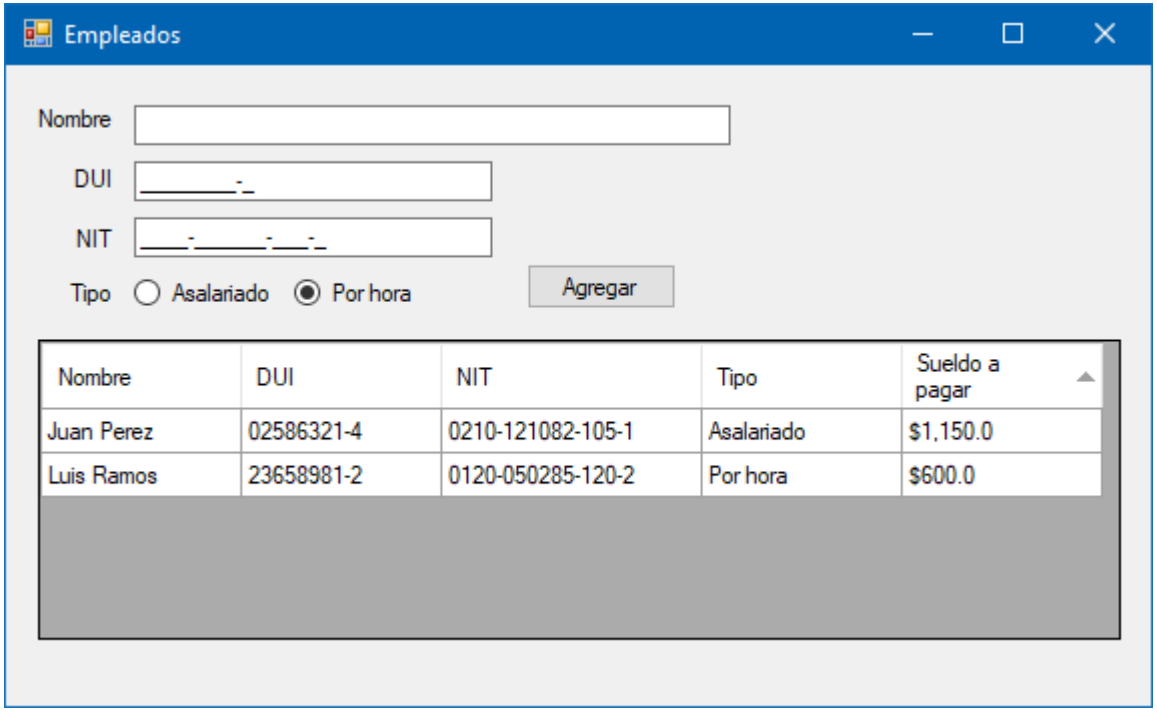
- **Clase derivad** EmpleadoPorHora:

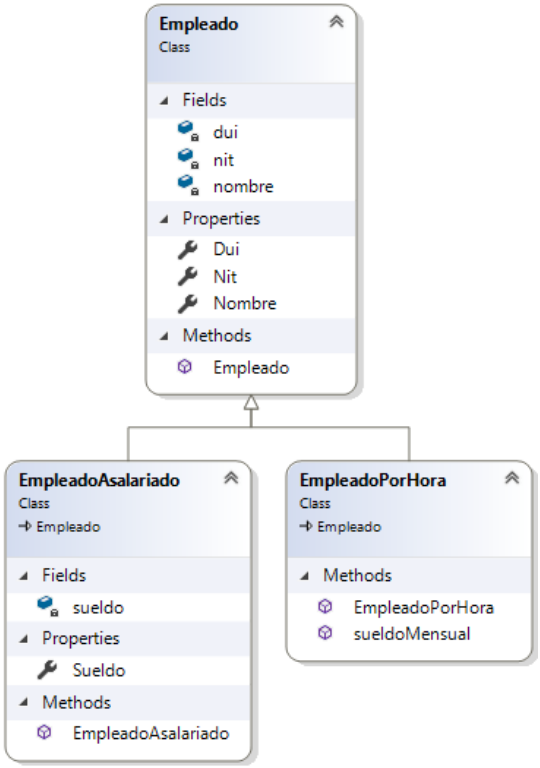



```
class EmpleadoPorHora : Empleado
{
    0 references
    public EmpleadoPorHora(string nombre, string dui, string nit)
        : base(nombre, dui, nit)
    {

    }

    0 references
    public decimal sueldoMensual(int horas, decimal valor)
    {
        return horas * valor;
    }
}
```

DEMO: EMPLEADOS (CONT.)

- Implementación:





MODIFICADORES DE ACCESO

- Modificadores de acceso para elementos de clase:

Accesibilidad declarada	Significado
public	Acceso no restringido.
protected	Acceso limitado a la clase contenedora o a los tipos derivados de esta clase.
internal	Acceso limitado al ensamblado actual.
protected internal	Acceso limitado al ensamblado actual o a los tipos derivados de la clase contenedora.
private	Acceso limitado al tipo contenedor.

- Modificadores de acceso para clases:

Accesibilidad declarada	Significado
public	Una clase public no posee restricciones.
abstract	Una clase abstract (abstracta) no se puede instanciar, sirve como clase base para herencia.
sealed	Una clase sealed (sellada) no se puede heredar, únicamente se puede instanciar.

BIBLIOGRAFÍA

- Asad, A. (2017). The C# Programmer's Study Guide (MCSD) Exam: 70-483. Pakistan: Apress.
- Ceballos, F. (2013). Microsoft C# Curso de programación. Segunda edición. México: Alfaomega.
- Putier, S. (2018). C# 7 y Visual Studio 2017. España: ENI.

- Microsoft Docs:
- https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/object-oriented-programming
- https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/inheritance