****

**Cámara de Legisladores**

**Practico nº1**

**Programación 2**

Ezequiel Acuña

Gary Pacheco

Profesor: Gonzalo Duarte

2023

# Resumen

En este proyecto, nos adentraremos en el mundo de la programación orientada a objetos utilizando el lenguaje C# en el entorno de Visual Studio. El enfoque central de este trabajo es la implementación de un sistema que simule el funcionamiento de una cámara de legisladores.

Para llevar a cabo esta tarea, comenzaremos creando una clase llamada "Legislador". Esta clase contendrá una serie de atributos esenciales, como el "PartidoPolítico", "DepartamentoQueRepresenta", "NumDespacho", "Nombre", "Apellido", "Edad", y una indicación de si el legislador está casado o no. Además, la clase estará equipada con un método denominado "getCamara" que determinará la cámara a la que pertenece el legislador, ya sea el Senado o la Cámara de Diputados.

A continuación, expandiremos nuestro sistema creando las clases "Senador" y "Diputado", ambas heredando de la clase base "Legislador". Estas subclases incorporarán características específicas, y en particular, la clase "Diputado" tendrá un constructor que inicializará la lista de legisladores, y se implementarán las siguientes funciones:

"RegistrarLegislador": Esta función permitirá la adición de un nuevo legislador a la lista.

"ListarCamaras": Con esta función, se imprimirá en la consola el número de despacho y la cámara en la que trabaja cada legislador, ya sea en el Senado o en la Cámara de Diputados.

Además, crearemos un método dentro de la clase "Parlamento" que calculará y proporcionará la cantidad de senadores y diputados presentes en el Parlamento en un momento dado.

Adicionalmente, abordaremos la implementación de métodos polimórficos, tales como:

"presentarPropuestaLegislativa": Este método permitirá que un legislador presente una propuesta legislativa en la cámara correspondiente.

"votar": Un método que habilitará a un legislador para emitir su voto durante las sesiones legislativas.

"participarDebate": Un último método que permitirá a un legislador involucrarse activamente en debates legislativos.

En conjunto, este proyecto constituirá una sólida representación de cómo se puede utilizar la programación orientada a objetos en C# para simular el funcionamiento de una cámara de legisladores, desde la creación de legisladores individuales hasta su participación en procesos legislativos y debates en la cámara.

ÍNDICE

[Resumen 1](#_Toc147076814)

[Introducción 1](#_Toc147076815)

[Programación orientada a Objetos 1](#_Toc147076816)

[Método 1](#_Toc147076817)

[Herencia 1](#_Toc147076818)

[Superclase 1](#_Toc147076819)

[Subclase 1](#_Toc147076820)

[Reutilización 1](#_Toc147076821)

[Polimorfismo 2](#_Toc147076822)

[Metodología de trabajo 2](#_Toc147076823)

[Clase Legislador 2](#_Toc147076824)

[Clases derivadas de Legislador 2](#_Toc147076825)

[Clase Parlamento 2](#_Toc147076826)

[Program 3](#_Toc147076827)

[Resultado 4](#_Toc147076828)

[Conclusión 7](#_Toc147076829)

[Referencias 8](#_Toc147076830)

# Introducción

## Programación orientada a Objetos

La programación Orientada a objetos se define como un paradigma de la programación, una manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones.

Podemos entender la programación Orientada a objetos (POO) como una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación, que permite diseñar mejor las aplicaciones, llegando a mayores cotas de complejidad, sin que el código se vuelva inmanejable. [[1]](#uno)

Clase

En la programación orientada a objetos, una clase es una plantilla o modelo para crear objetos. Una clase define un conjunto de atributos (datos) y métodos (comportamientos) que los objetos pueden tener. Los atributos de una clase son las características o propiedades que los objetos de esa clase tienen. Los métodos son las acciones que los objetos pueden realizar. [[2]](#dos)

## Método

Un método es una función que se define dentro de una clase y se utiliza para representar el comportamiento de un objeto. Los métodos se utilizan para realizar tareas específicas en un objeto, como cambiar el estado de un objeto, calcular un valor, interactuar con otros objetos, etc. [[3]](#tres)

## Herencia

Es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase. La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados.

Superclase: la clase cuyas características se heredan se conoce como superclase (o una clase base o una clase principal).

Subclase: la clase que hereda la otra clase se conoce como subclase (o una clase derivada, clase extendida o clase hija). La subclase puede agregar sus propios campos y métodos, además de los campos y métodos de la superclase.

Reutilización: la herencia respalda el concepto de “reutilización”, es decir, cuando queremos crear una clase nueva y ya hay una clase que incluye parte del código que queremos, podemos derivar nuestra nueva clase de la clase existente. Al hacer esto, estamos reutilizando los campos/atributos y métodos de la clase existente. [[4]](#cuatro)

## Polimorfismo

El polimorfismo es una relajación del sistema de tipos, de tal manera que una referencia a una clase (atributo, parámetro o declaración local o elemento de un vector) acepta direcciones de objetos de dicha clase y de sus clases derivadas (hijas, nietas, ...). [[5]](#cinco)

# Metodología de trabajo

## Clase Legislador

La clase Legislador obtendrá los siguientes atributos, estos mismos serán definidos en la creación de cada objeto de esta clase en el program.

* Nombre (tipo string)
* Apellido (tipo string)
* Edad (tipo int)
* Casado (tipo string)
* Departamento (tipo string)
* Despacho (tipo int)

Esta clase contara con los siguientes métodos:

* getCamara(). Este método devolverá la cámara del legislador, dependiendo del número de su despacho, los legisladores con el número de despacho del 1 al 50, van a la cámara de senadores, y los posean despachos con números mayor a 50 a la cámara de diputados. Los senadores irán a la cámara alta y los diputados a la cámara baja.

También contara con estos metodos polimorficos:

* PresentarPropuesta (int despacho)
* Votar (int despacho)
* ParticiparDebate (int despacho)

Estos métodos polimórficos usaran el número de despacho para encontrar al legislador que este ejecutando la acción

## Clases derivadas de Legislador

La clase legislador tendrá 2 clases derivadas, siendo estas la clase Diputado y la clase Senador. Estas 2 clases usaran el constructor de la clase padre.

La clase Diputado tendrá la propiedad exclusiva NumAsientoCamaraBaja, y la clase Senador la propiedad NumAsientoCamaraAlta.

## Clase Parlamento

Esta clase contara con una propiedad, la cual será una lista en la que se irán agregando todos los legisladores, ya sean diputados o senadores.

El constructor de esta clase inicializara la lista de legisladores.

La clase Parlamento contara con los siguientes métodos:

* RegistrarLegislador(). Este método será llamado desde program para registrar a un nuevo legislador, creará un objeto de la clase Legislador y lo guardará en la lista de legisladores.
* ListarCamaraBaja(). Este método usara un foreach para recorrer toda la lista de legisladores, para luego listar a todos los legisladores utilizando el método getCamera().
* ListarCamaraAlta(). Este método tendrá la misma función que el método anterior con la diferencia de que listará los legisladores que se encuentren en la cámara alta, siendo estos los senadores.
* MostrarLegisladores(). Recorrerá toda la lista de legisladores para imprimir sus datos en pantalla, este método será utilizado también para elegir el legislador a la hora de votar, presentar una propuesta, o participar de un debate.
* CantidadLegisladores(). Este método tendrá un contador para senadores y otro para diputados, recorrerá toda la lista de legisladores, y contará los mismos, para luego devolver el número final que se encuentra en cada cámara.

## Program

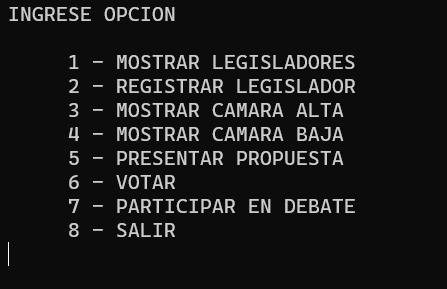
En primera instancia se crearán algunos objetos de las clases Senador y Diputado, se creará la lista legisladores y se añadirán los mismos.

El programa contará con una interfaz que se repetirá en bucle al ingresar una opción hasta que el usuario dese salir del programa, para interactuar con las diferentes utilidades. Podrán ser elegidas con un switch y las opciones disponibles serán:

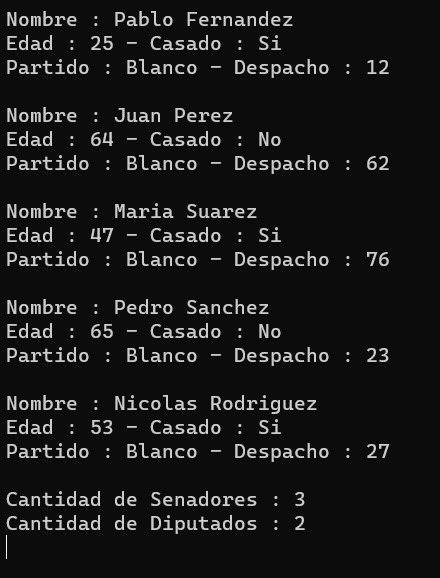
* Mostrar Legisladores. Esta opción mostrara el listado de todos los legisladores, y también la cantidad de senadores y diputados que se encuentren en la lista.
* Registrar Legislador. Esto pedirá al usuario ingresar los datos del legislador a registrar, para luego añadirlo a la lista de legisladores.
* Mostrar cámara alta. Esta opción mostrara la lista de senadores, indicando su nombre completo, cámara y numero de despacho
* Mostrar cámara baja. Mostrará la lista de diputados, indicando los mismos datos indicados anteriormente.
* Presentar propuesta. Esta opción le mostrará al usuario la lista completa de legisladores en el parlamento, y le pedirá ingresar el número de despacho del legislador con el cual se quiere presentar la propuesta.
* Votar. Esta opción realizara la misma acción que la opción anterior habilitando al legislador a votar.
* Participar en debate. Con esta opción un legislador podrá participar en un debate, también inscribiéndose por su número de despacho
* Salir. Esta opción pondrá fin al programa.

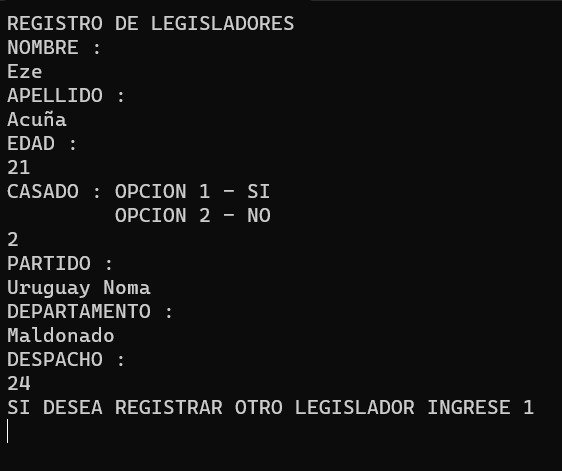
# Resultado

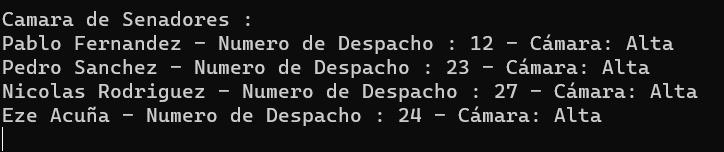
* Menú Principal



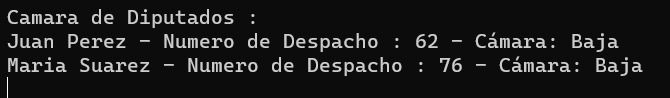
* Listado de legisladores, cantidad de diputados y senadores



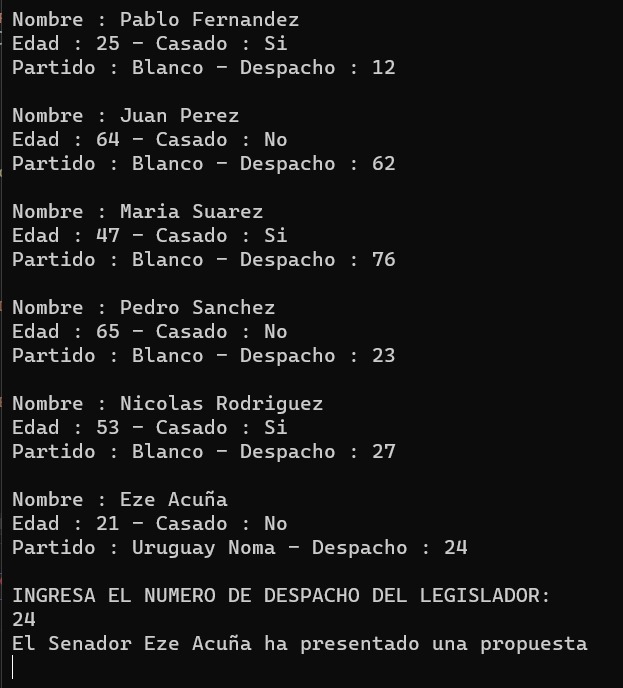
* Registro de legisladores  
  
* Listado de la cámara de senadores



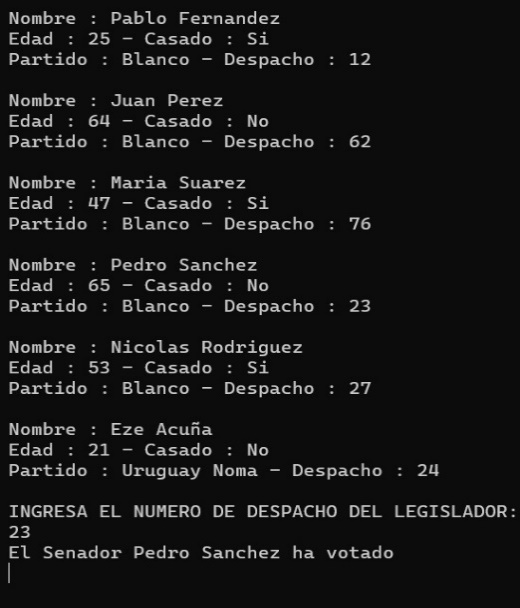
* Listado de cámara de diputados



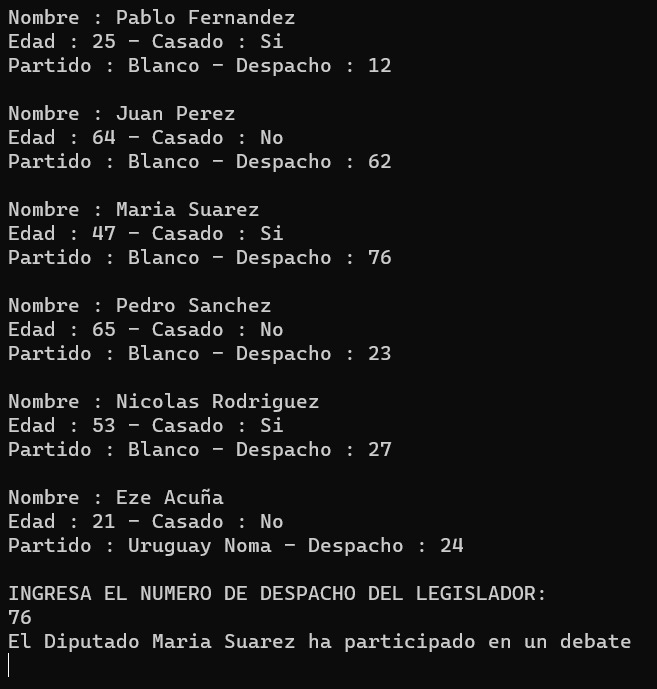
* Presentar propuesta



* Ingresar voto



* Participar en debate



# Conclusión

En base al ejercicio planteado se pudo crear un sistema que permite simular el comportamiento de una Cámara de legisladores y a la vez teniendo en cuenta las funcionalidades pedidas que no presentaron ser un desafío como la creación de clases y el uso de listas para llevar un conteo de la cantidad de senadores y legisladores que fueron agregados por el usuario.

Métodos como mostrar la cámara a la que pertenecen los legisladores fueron realizados con éxito, a la vez cuenta con un apartado que muestra la cantidad de senadores y diputados que hay en el parlamento.

El uso de Métodos Polimórficos como el de presentar una propuesta, votar y participar de un debate se realizaron sin complicaciones.

# Referencias

[1]<https://desarrolloweb.com/articulos/499.php>

[2]<https://programacion.top/orientada-objetos/clase/>

[3]<https://programacion.top/orientada-objetos/metodo/?expand_article=1>

[4]<https://ifgeekthen.nttdata.com/es/herencia-en-programacion-orientada-objetos>

[5]<https://desarrolloweb.com/articulos/polimorfismo-programacion-orientada-objetos-concepto.html>