

Practica N°1 Carrera Analista Programador- Programación 2

**Analista Programador**

**Practica N°1**

**Ejercicio Parlamento**

**Alumnos: Juan Bordabehere**

**Miguel Gaitán**

**Docente: Gonzalo Ariel Duarte**

CEI Formación y Cursos Intensivos

**Contenido**

**1. Resumen**

**2. Introducción**

1. Historia C#.........................................................................................................1
2. Lenguaje tipado................................................................................................
3. Getters y Setters................................................................................................
4. Clases.................................................................................................................
5. Herencia..............................................................................................................2
6. Polimorfismo......................................................................................................

**3. Desarrollo del programa**

1. Creación de clases “Padre e Hijos” y sus atributos................................................3
2. Armado de Getters, Setters y métodos...................................................................5
3. Implantación de lógica en el programa..................................................................10

**5. Conclusión**.................................................................................................21

**Resumen:**

El proyecto se desarrolla en el entorno de programación C#, una plataforma altamente versátil y potente que brinda la base sólida necesaria para crear (por ej.) un sistema de gestión parlamentaria. En su esencia, este proyecto se concibe como un sistema integral de gestión parlamentaria diseñado para manejar y detallar la cantidad de legisladores que se encuentran en un parlamento.

Ofrece la capacidad de cuantificar, de manera precisa y organizada los legisladores que se incorporan a una lista específica. Además, el proyecto agrega un valor significativo al facilitar el proceso de votación en un entorno parlamentario simulado.

Uno de los aspectos más destacados de este proyecto es su capacidad para permitir que legisladores específicos presenten propuestas legislativas virtuales. Esta funcionalidad brinda a los representantes la oportunidad de expresar sus opiniones y argumentar en un entorno simulado

Para garantizar la accesibilidad y usabilidad del sistema, se ha diseñado una interfaz de usuario intuitiva y amigable. Esta interfaz presenta una variedad de menús que simplifican la navegación y el uso del sistema, incluso para aquellos usuarios que no poseen experiencia técnica previa.

Su capacidad para registrar legisladores, cuantificar su presencia, facilitar votaciones simuladas y fomentar propuestas legislativas virtuales, lo convierte en una herramienta valiosa para comprender el proceso legislativo al tiempo que su diseño centrado en el usuario garantiza una experiencia agradable para todos los usuarios, independientemente de su nivel de habilidad técnica. Principio del formulario

**2.1 Historia C#:**

"C#" (pronunciado cii Sharp en inglés) es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

**2.2 Lenguaje Tipado:**

El término "lenguaje tipado" se refiere a una característica de los lenguajes de programación que impone restricciones sobre cómo se pueden utilizar los tipos de datos. En un lenguaje tipado, cada variable, constante y expresión tiene un tipo de datos asociado, y el lenguaje impone reglas sobre cómo se pueden combinar estos tipos de datos en operaciones.

* 1. **Getters y Setters**

Los métodos "getters" y "setters" en C# son parte de la programación orientada a objetos y se utilizan para acceder y modificar los valores de los campos (variables) de una clase de manera controlada. Estos métodos se conocen comúnmente como propiedades y permiten encapsular el acceso a los campos, lo que es útil para mantener el control sobre el estado interno de un objeto.

* 1. **Clases**

En C#, una clase es un componente fundamental de la programación orientada a objetos (POO). Una clase es un plano o plantilla para crear objetos, y se utiliza para definir la estructura y el comportamiento de esos objetos. Para crear una clase en C#, utiliza la palabra clave class, seguida del nombre de la clase y un bloque de código que define las propiedades y métodos de la clase.

1

* 1. **Herencia**

En C#, la herencia es un concepto fundamental de la programación orientada a objetos que permite que una clase (llamada clase derivada o subclase) herede características y comportamientos de otra clase (llamada clase base o superclase). La herencia permite crear una relación jerárquica entre clases, lo que facilita la reutilización de código y la organización de las clases de manera lógica y coherente. Cuando una clase hereda de otra, la clase derivada adquiere automáticamente los campos, propiedades, métodos y eventos de la clase base. Esto significa que la clase derivada puede acceder y utilizar todos los miembros públicos y protegidos de la clase base como si fueran propios. Además, la clase derivada puede agregar nuevos miembros o modificar el comportamiento de los miembros heredados utilizando la herencia.

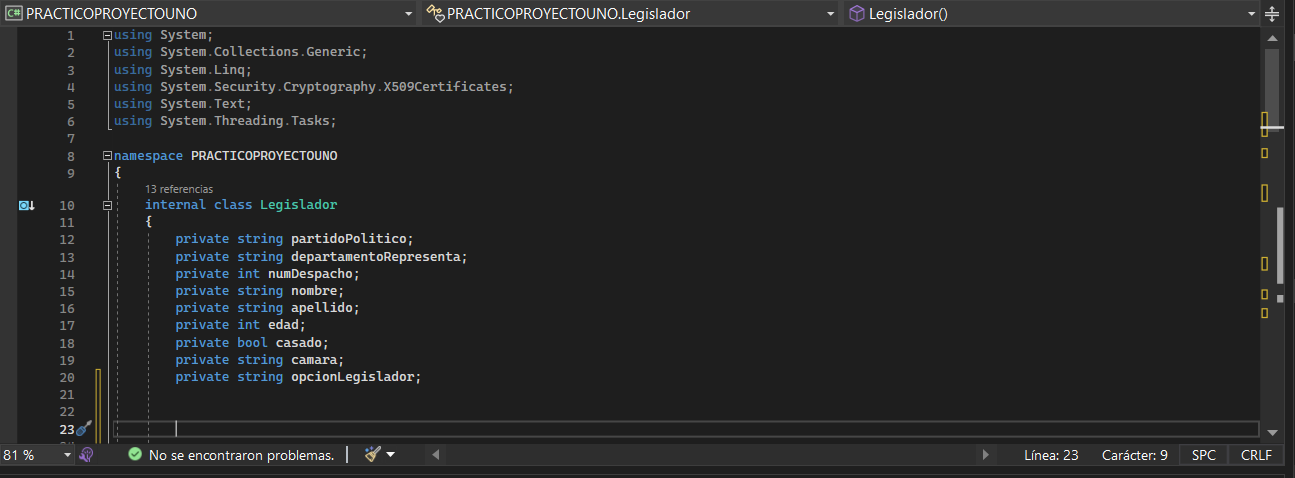
* 1. **Polimorfismo**

Polimorfismo de herencia: Esto implica que una clase derivada (subclase) herede de una clase base (superclase) y reemplace (anule) los métodos de la clase base con sus propias implementaciones. Luego, se pueden crear objetos de la clase derivada y tratarlos como objetos de la clase base. Cuando se llama a un método en un objeto de la clase base, se ejecuta la implementación específica de la clase derivada si está disponible. Esto permite que diferentes tipos de objetos se comporten de manera diferente según sus implementaciones específicas.

2

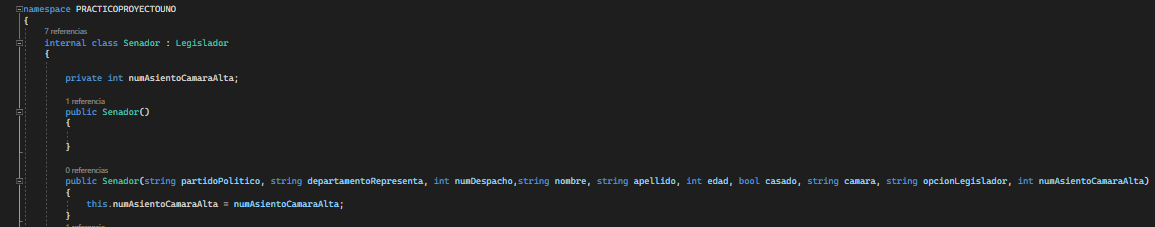
**Desarrollo del programa**

**3.1** Creación de la clase (Legislador). “Clase Padre”

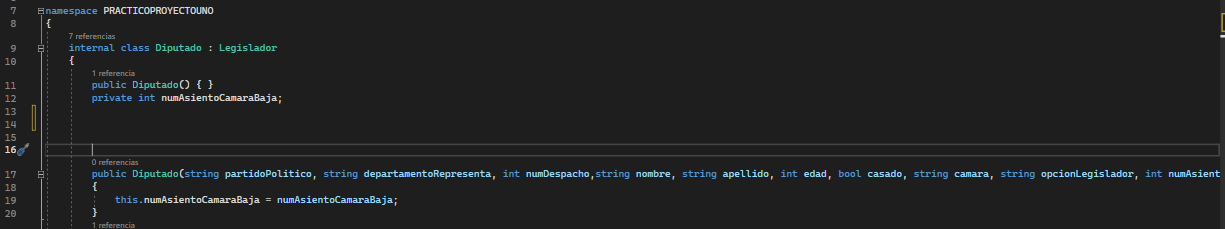
****

En esta instancia comenzamos con la creación de la clase padre de nuestro programa, esta contiene atributos privados ya que trabajaremos sus valores a través de los métodos get y set de cada atributo. Como ilustra la imagen, en la clase se manejan atributos tales como intr., string y bool y sus atributos son: patidoPolitico, DepartamentoRepresenta, numDespacho, nombre, edad, casado, cámara, opcionLegislador.

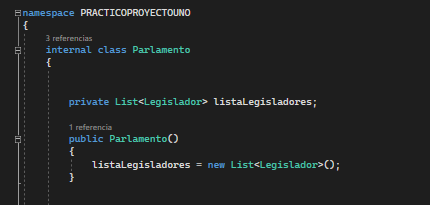
Creación clase (Senador). “Clase Hija”



3

Creación clase (Diputado). “Clase Hija”

Al momento de crear las clases hijas estas heredan todos los atributos de la clase padre “Legislador”, en este momento cada clase hija presenta un atributo único que la nombramos numAsientoCamaraBaja para la clase Diputado y numAsientoCamaraBaja para la clase Senador. Por otro lado creamos un constructor para cada clase con sus respectivos parámetros que heredaran de la clase “legislador” (Padre).

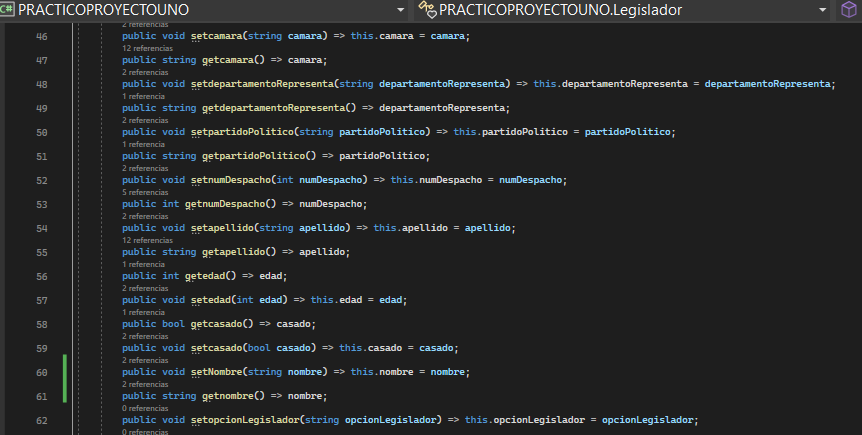
****

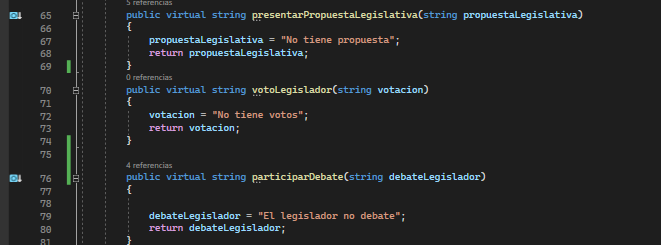
En cuanto a la creación de la clase parlamento esta no heredara ningún atributo, en ella se crea como atributo la lista “legislador” donde se almacenaran tanto diputados como senadores registrados previamente.

4

**3.2 Armado de Getters, Setters y Métodos.**

**LEGISLADOR:**

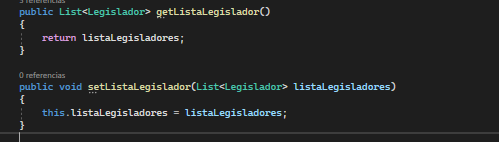
****

****

En estas imágenes se muestra la creación de getters y setters de la clase **“Legislador”** que luego usaremos para setear valores y poder llamarlos para ser representados, esta acción se realizó también para las clases “Diputado”, “Senador” y “Parlamento”. También aquí como se muestra en la imagen se crean los métodos polimórficos “presentarPropuestaLegislativa”, “votoLegislador”, “participarDebate” que luego durante el programa serán modificados a necesidad.

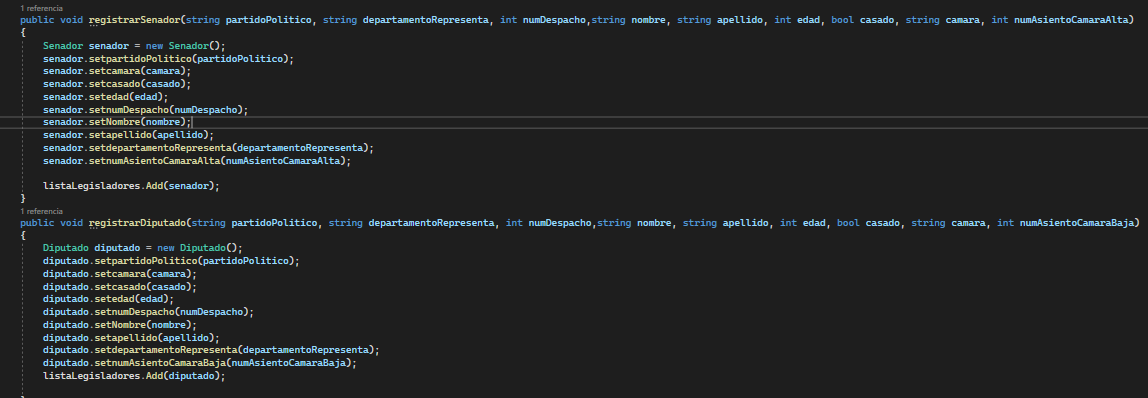
5

**PARLAMENTO:**



Acerca de la clase **Parlamento** esta contiene un método get (getListaLegislador ()) que su función principal es permitir a otros objetos obtener la lista de legisladores almacenada en esta clase, y un método set (setListaLegislador ()) que su función principal es modificar es establecer o actualizar la lista de legisladores.

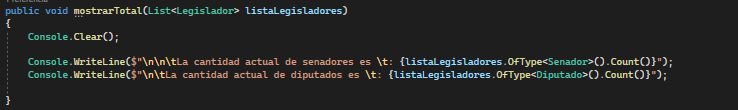
En cuanto a los métodos que la clase parlamento contiene son diversos a continuación mediante imágenes se explicaran detalladamente.

****

Public void registrarSenador (): Este método se utiliza para registrar un senador en la lista de legisladores. Los parámetros proporcionados se utilizan para configurar las propiedades del objeto Senador, como el partido político, el departamento que representa, el nombre, el apellido, la edad, etc. Luego, se crea un objeto Senador, se configura con los valores proporcionados y se agrega a la lista listaLegisladores.

Public void registrarDiputado (): Este método se utiliza para registrar un diputado en la lista de legisladores. Al igual que el método anterior, toma varios parámetros para configurar las propiedades del objeto Diputado, como el partido político, el departamento que representa, el nombre, el apellido, la edad, etc. Luego se crea el objeto Diputado y se configura con los valores proporcionados previamente y se agrega a la lista listaLegisladores.

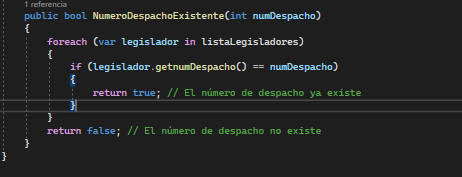
6



listaLegisladores.OfType<Senador> ().Count (): Esto cuenta la cantidad de elementos en la lista listaLegisladores que son del tipo Senador. OfType <Senador> () filtra la lista para seleccionar solo los elementos que son instancias de la clase Senador, y luego Count () cuenta cuántos elementos cumplen con ese criterio.

listaLegisladores.OfType <Diputado> ().Count (): Similar al caso anterior, esto cuenta la cantidad de elementos en la lista listaLegisladores que son del tipo Diputado

Este método luego será utilizado para representar mediante un opción en menú de consola la cantidad de Senadores y Diputados que se van registrando.



public bool NumeroDespachoExistente(int numDespacho): Este es un método público que toma un número de despacho como argumento y devuelve un valor booleano (true o false). El objetivo principal de este método es determinar si el número de despacho proporcionado ya existe en la lista de legisladores.

foreach (var legislador in listaLegisladores): Este bucle foreach recorre todos los elementos de la lista de legisladores (listaLegisladores). En cada iteración, el elemento actual se almacena en la variable legislador.

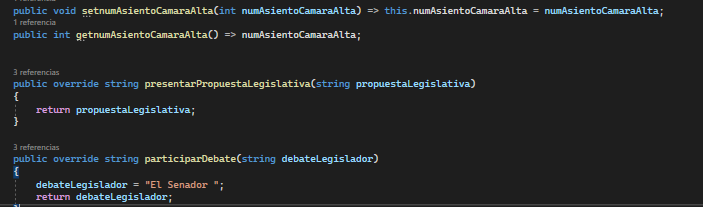
if (legislador.getnumDespacho() == numDespacho): En cada iteración del bucle, se verifica si el número de despacho del legislador actual es igual al número de despacho proporcionado como argumento al método.

Si la condición legislador.getnumDespacho() == numDespacho se cumple para algún legislador en la lista, significa que el número de despacho ya existe en la lista. En ese caso, el método devuelve true, indicando que el número de despacho es existente.

7

Si el bucle foreach completa su ejecución sin encontrar ningún legislador con el número de despacho proporcionado, el método devuelve false, indicando que el número de despacho no existe en la lista.

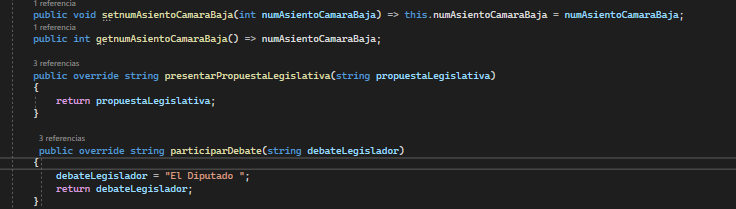
**SENADOR:**

****

En la clase Senador se crea el método set y get del número de asiento de cámara alta ya que este atributo (numAsientoCamaraAlta) solamente lo posee la clase hijo “Senador”. En cuanto a los métodos presentarPropuestaLegislativa y participarDebate, estos anulan a sus respectivos métodos en la clase padre y devuelven un valor diferente, este será introducido por el usuario en el caso de presentarPropuestaLegislativa(). Estos modifican el valor al método polimórfico que se encuentra en la clase padre (Legislador).

8

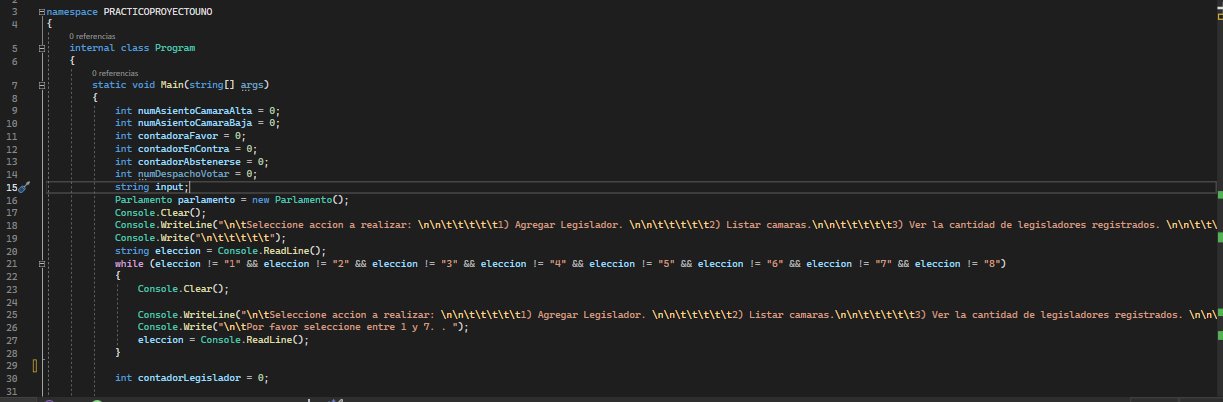
**DIPUTADO:**

****

En la clase Senador se crea el método set y get del número de asiento de cámara Baja ya que este atributo (numAsientoCamaraBaja) solamente lo posee la clase hijo “Diputado”. En cuanto a los métodos presentarPropuestaLegislativa y participarDebate, estos anulan a sus respectivos métodos en la clase padre y devuelven un valor diferente, este será introducido por el usuario en el caso de presentarPropuestaLegislativa(). Estos modifican el valor al método polimórfico que se encuentra en la clase padre (Legislador).

9

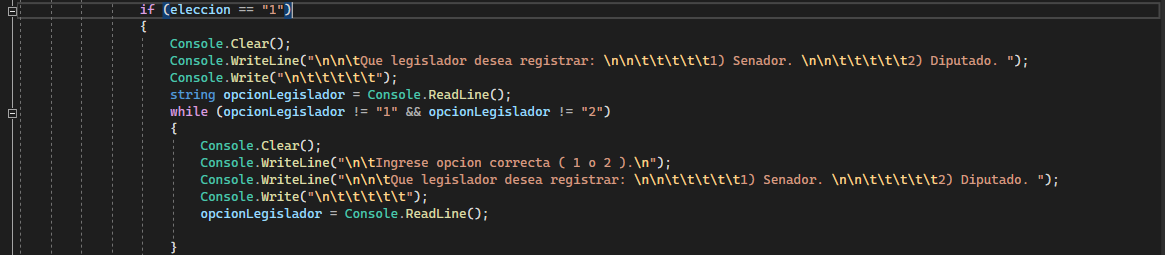
**3.3 Implementación de lógica del programa**

****

Se declaran varias variables, incluyendo numAsientoCamaraAlta, numAsientoCamaraBaja, contadoraFavor, contadorEnContra, contadorAbstenerse, numDespachoVotar, input, y se crea el objeto parlamento. Estas variables se utilizan para almacenar datos y controlar el flujo del programa.

Se muestra un menú en la consola que le permite al usuario seleccionar una acción entre las opciones del 1 al 8. El usuario ingresa su elección a través de (eleccion).

Hay un bucle while que valida que la elección del usuario esté dentro del rango válido (1 a 8). Si el usuario ingresa una elección inválida, se le pedirá que lo haga nuevamente hasta que ingrese una elección válida. Una vez que el usuario ha seleccionado una opción válida, el programa procederá a realizar la acción correspondiente según la elección.



Este código maneja la opción 1 del menú principal, que es "Agregar Legislador".

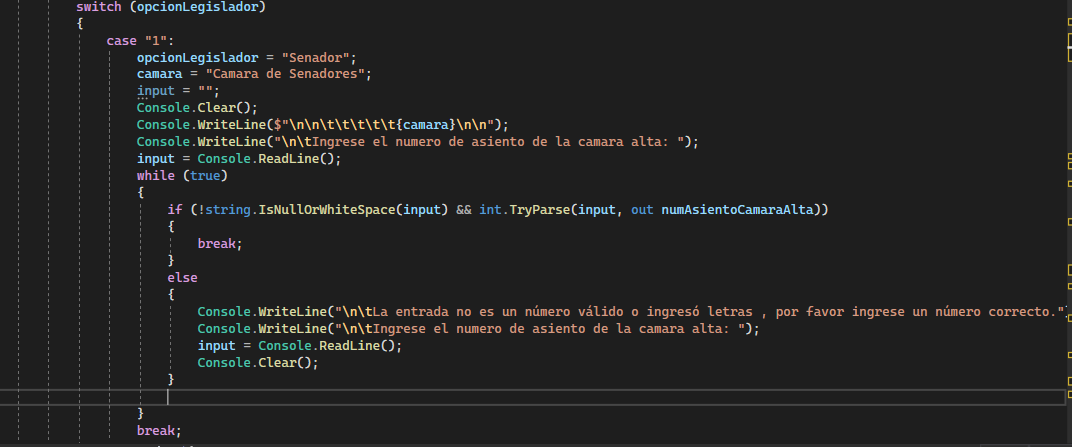
Se verifica si la elección del usuario (eleccion) es igual a "1", lo que indica que el usuario desea registrar un legislador.

Si la elección es igual a "1", se muestra un submenú en la consola que le permite al usuario seleccionar qué tipo de legislador desea registrar: Senador (1) o Diputado (2).

10

Se utiliza un bucle while para asegurarse de que la elección del tipo de legislador (opcionLegislador) sea válida. Mientras el usuario no haya ingresado "1" o "2", se mostrará un mensaje de error y se le pedirá que ingrese una opción válida.

Una vez que el usuario ha seleccionado una opción válida (1 o 2), el programa procederá a registrar el tipo de legislador elegido según la elección del usuario.



Se utiliza un switch para evaluar la variable opcionLegislador, que se estableció previamente en "1" cuando el usuario eligió registrar un Senador.

Cuando opcionLegislador es igual a "1", se ejecuta el código dentro del caso "1". Aquí se realiza el registro de un Senador.

Se establece opcionLegislador en "Senador" y camara en "Camara de Senadores" para reflejar la elección del usuario.

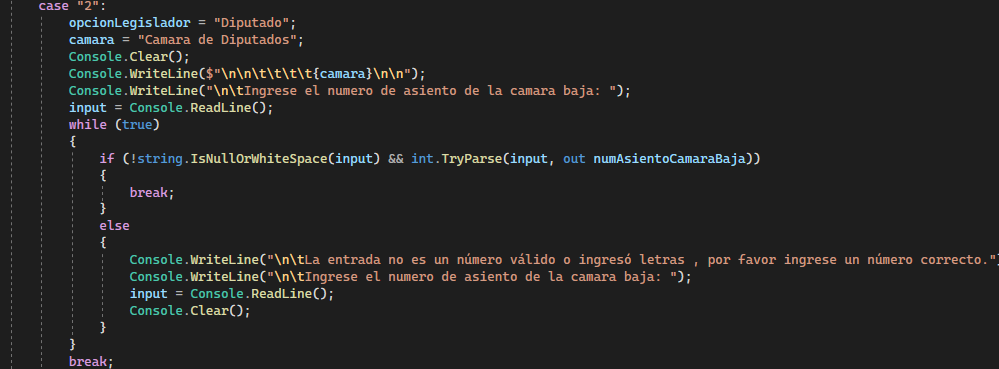
Se utiliza una variable input para almacenar la entrada del usuario, y se borra la pantalla con Console.Clear() para proporcionar una interfaz de usuario limpia.

Luego, se utiliza un bucle while (true) para validar la entrada del usuario. El bucle se repetirá hasta que se ingrese un número válido.

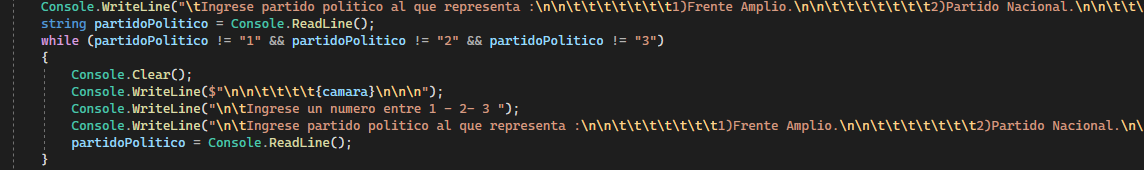
Dentro del bucle, se verifica si la entrada no está vacía y si se puede analizar como un número entero utilizando int.TryParse(). Si ambas condiciones se cumplen, el número se almacena en la variable numAsientoCamaraAlta, y el bucle se rompe con break.

Si la entrada no es válida (por ejemplo, si el usuario ingresó letras o una cadena vacía), se muestra un mensaje de error, se solicita al usuario que ingrese el número nuevamente y se borra la pantalla con Console.Clear() para dar una mejor experiencia al usuario.

11



Esta imagen refleja el mismo procedimiento que para el caso anterior solo que este registra un diputado y le da un valor a el numero de asiento de camara baja. Tambien se realiza la validación para dicho ingreso.



Este fragmento de código maneja la entrada del partido político al que representa un legislador (en este caso, un Senador).

Se muestra un mensaje en la consola que solicita al usuario que ingrese el partido político al que representa el legislador. Los usuarios pueden seleccionar entre tres opciones numeradas: "1) Frente Amplio", "2) Partido Nacional" y "3) Partido Colorado".

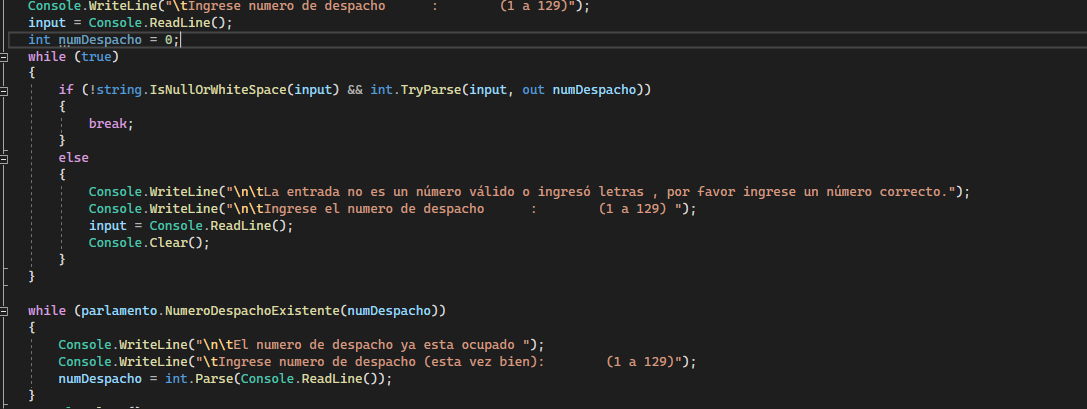
Se utiliza la variable partidoPolitico para almacenar la elección del usuario mediante Console.ReadLine().

Luego, se inicia un bucle while para verificar si la elección del usuario es válida. El bucle se ejecutará mientras partidoPolitico no sea "1", "2" ni "3".

12



Este fragmento de código representa un fragmento de la selección del departamento al que representa el legislador a registrar.



Este fragmento de código maneja la entrada del número de despacho de un legislador en el programa.

Se muestra un mensaje en la consola que solicita al usuario que ingrese un número de despacho, con un rango válido entre 1 y 129.

Se utiliza la variable input para almacenar la entrada del usuario mediante Console.ReadLine ().

Luego, se inicia un bucle while para verificar si la entrada del usuario es válida. El bucle se ejecutará mientras input no sea una cadena no vacía y no se pueda analizar como un número entero válido utilizando int.TryParse().

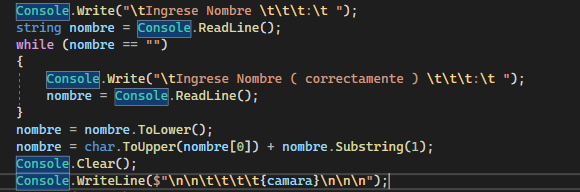
Si la entrada del usuario no es válida (por ejemplo, si ingresó letras en lugar de un número o un número fuera del rango permitido), se muestra un mensaje de error, se solicita al usuario que ingrese nuevamente el número de despacho y se borra la pantalla con Console.Clear() para proporcionar una interfaz de usuario limpia.

Una vez que el usuario ha ingresado un número de despacho válido, se almacena en la variable numDespacho y el bucle se rompe con break.

13

Luego, se utiliza otro bucle while para verificar si el número de despacho ingresado ya existe en el parlamento. El método parlamento.NumeroDespachoExistente (numDespacho) está diseñado para verificar la existencia del número de despacho en la lista de legisladores . Si el número de despacho ya existe, se muestra un mensaje de error y se le pide al usuario que ingrese un número de despacho diferente.

Este código garantiza que se ingrese un número de despacho válido y no duplicado antes de continuar con la ejecución del programa.

****

Este fragmento de código maneja la entrada del nombre de un legislador en el programa y realiza algunas validaciones y formateo.

Se muestra un mensaje en la consola que solicita al usuario que ingrese un nombre.

Se utiliza la variable nombre para almacenar la entrada del usuario mediante Console.ReadLine().

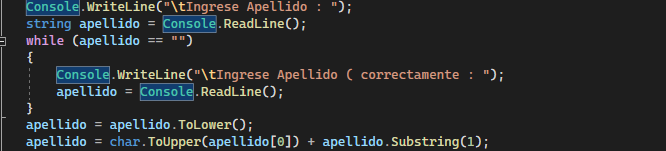
Se inicia un bucle while para verificar si la entrada del usuario está vacía (es decir, si el usuario no ingresó ningún nombre).

Si la entrada del usuario está vacía (es decir, nombre es una cadena vacía), se muestra un mensaje de error y se le pide al usuario que ingrese el nombre correctamente.

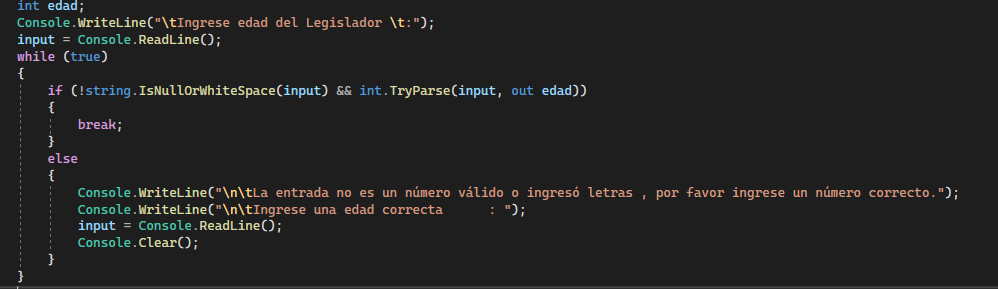
Luego, se realiza un formateo del nombre. Primero, se convierte la cadena nombre a minúsculas con nombre.ToLower().

Luego, se capitaliza la primera letra del nombre para que esté en mayúscula utilizando char.ToUpper(nombre[0]) para convertir la primera letra a mayúscula y nombre.Substring(1) para conservar el resto de la cadena en minúscula.

14

****

En el ingreso del apellido del legislador el código presenta las mismas características que en la imagen previa y su funcionamiento se repite.



Para el ingreso de la edad se declara una variable edad para almacenar la edad del legislador.

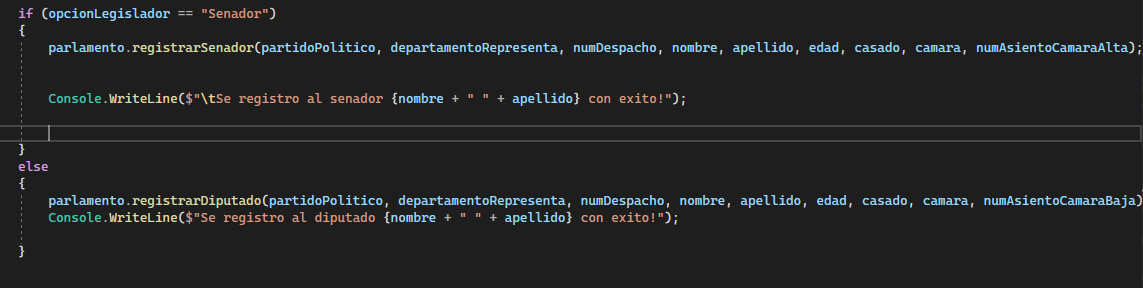
Se muestra un mensaje en la consola que solicita al usuario que ingrese la edad del legislador.

Se inicia un bucle while (true) para verificar si la entrada del usuario es válida. El bucle se ejecutará mientras input no sea una cadena no vacía y no se pueda analizar como un número entero válido utilizando int.TryParse().

Si la entrada del usuario no es válida (por ejemplo, si ingresó letras en lugar de un número o un número inválido), se muestra un mensaje de error, se solicita al usuario que ingrese la edad nuevamente y se borra la pantalla con Console.Clear() para proporcionar una interfaz de usuario limpia.

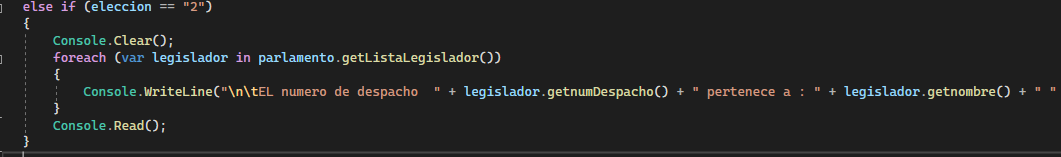
Una vez que el usuario ha ingresado una edad válida, se almacena en la variable edad y el bucle se rompe con break.

15

****

Luego de finalizar con el ingreso de datos mediante un if corroboramos si es un senador o un diputado que estamos registrando, si es un senador llamamos al método registrarSenador que tiene como propiedades todos los datos ingresados previamente y este ingresa al objeto Senador a una lista de legisladores. Si es un diputado este método realiza la misma acción.

Al finalizar cada una de las opciones se muestra en pantalla que el legislador ya sea diputado o senador se registraron con éxito.

****

Este fragmento de código maneja la opción "2" del menú principal, que es "Listar cámaras".

Se verifica si la elección del usuario (eleccion) es igual a "2", lo que indica que el usuario eligió la opción "Listar camaras".

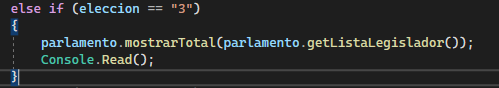
Se utiliza Console.Clear() para borrar la pantalla y proporcionar una interfaz limpia.

Se utiliza un bucle foreach para iterar a través de la lista de legisladores almacenados en el objeto parlamento utilizando parlamento.getListaLegislador().

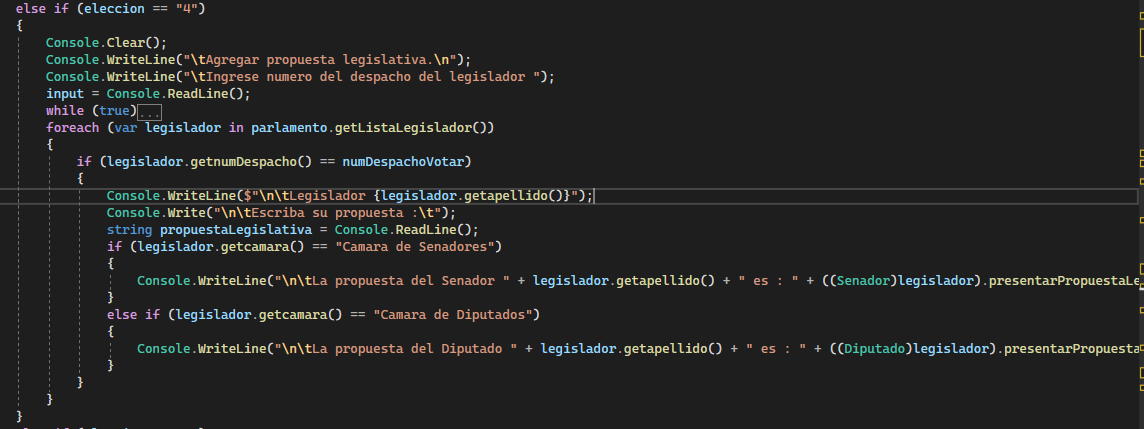
Para cada legislador en la lista, se muestra información relevante en la consola, como el número de despacho, el nombre, el apellido y la cámara a la que pertenece.

El programa espera a que el usuario presione Enter con Console.Read() antes de continuar.

16

****

Si el usuario ingresa en el menú principal la opción “3”, como muestra la imagen se llama al método mostrarTotal que se le pasa como parámetro la lista de legisladores al momento registrados y este muestra en consola la cantidad de senados y la cantidad de diputados registrados por separado.

****

Este fragmento de código maneja la opción "4" del menú principal, que es "Agregar propuesta legislativa" en el programa.

Se verifica si la elección del usuario (eleccion) es igual a "4", lo que indica que el usuario eligió la opción "Agregar propuesta legislativa".

Se utiliza Console.Clear() para borrar la pantalla y proporcionar una interfaz limpia.

Se solicita al usuario que ingrese el número de despacho del legislador al que desea agregar una propuesta legislativa.

La validación del ingreso de datos es exactamente igual a las usadas en las validaciones previas.

Luego, se utiliza un bucle foreach para iterar a través de la lista de legisladores almacenados en el objeto parlamento utilizando parlamento.getListaLegislador().

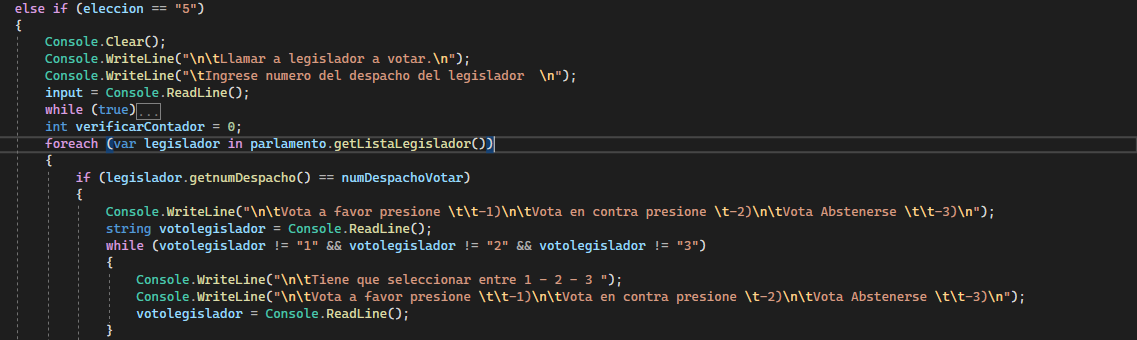
Para cada legislador en la lista, se verifica si el número de despacho del legislador coincide con numDespachoVotar. Si coincide, significa que se ha encontrado al legislador al que el usuario desea agregar una propuesta legislativa.

17

Se solicita al usuario que ingrese la propuesta legislativa.

Dependiendo de si el legislador es un Senador o un Diputado (determinado por legislador.getcamara()), se llama al método presentarPropuestaLegislativa específico de ese tipo de legislador para procesar la propuesta.

Se muestra un mensaje que indica la propuesta legislativa presentada por el legislador.



Este fragmento de código maneja la opción "5" del menú principal, que es "Llamar a legislador a votar" en el programa.

Se verifica si la elección del usuario (eleccion) es igual a "5", lo que indica que el usuario eligió la opción "Llamar a legislador a votar".

Se utiliza Console.Clear() para borrar la pantalla y proporcionar una interfaz limpia.

Se solicita al usuario que ingrese el número de despacho del legislador al que desea llamar a votar.

Se utiliza un bucle foreach para iterar a través de la lista de legisladores almacenados en el objeto parlamento utilizando parlamento.getListaLegislador().

Para cada legislador en la lista, se verifica si el número de despacho del legislador coincide con numDespachoVotar. Si coincide, significa que se ha encontrado al legislador al que el usuario desea llamar a votar.

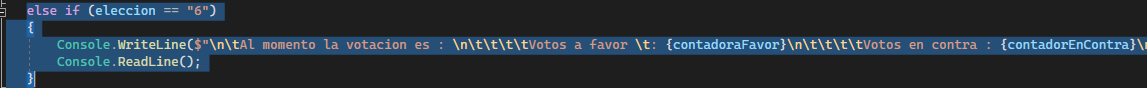
Se muestra un mensaje que permite al usuario seleccionar su voto ("1" para votar a favor, "2" para votar en contra o "3" para abstenerse) utilizando Console.ReadLine().

Se utiliza un bucle while para verificar si la elección del usuario (votolegislador) es válida. El bucle se ejecutará mientras votolegislador no sea "1", "2" ni "3".

18

Si la elección del usuario no es válida (por ejemplo, si no ingresó "1", "2" ni "3"), se muestra un mensaje de error y se le pide al usuario que ingrese su voto nuevamente.

Posteriormente a esto se abre un menú en consola en el cual el usuario elige en un menú de 3 opciones la decisión de si vota a favor, en contra o se abstiene el legislador.



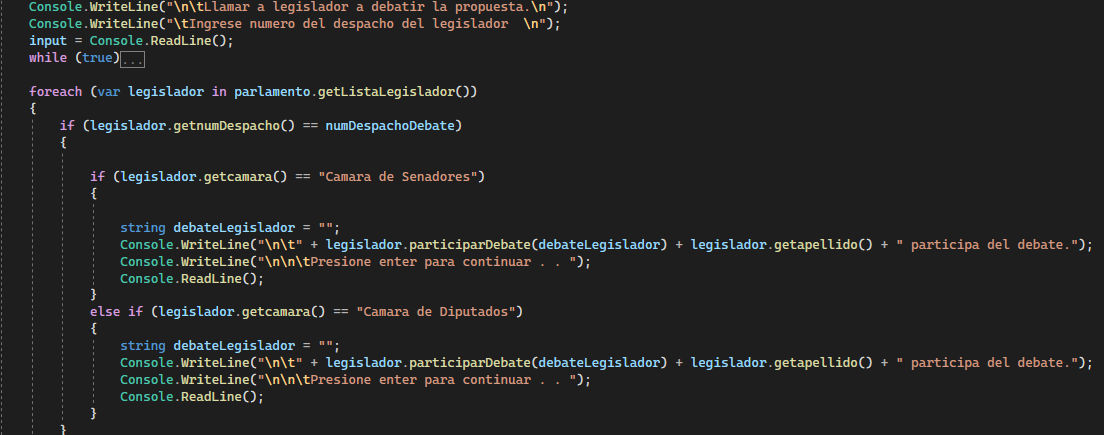
Este fragmento de código maneja la opción "6" del menú principal, que es "Ver votación" en el programa.

Se verifica si la elección del usuario (eleccion) es igual a "6", lo que indica que el usuario eligió la opción "Ver votación".

Se utiliza Console.WriteLine() para mostrar un mensaje que proporciona información sobre el estado actual de la votación. Muestra el número de votos a favor (contadoraFavor), el número de votos en contra (contadorEnContra) y el número de abstenciones (contadorAbstenerse).

El programa espera a que el usuario presione Enter con Console.ReadLine() antes de continuar. Esto permite al usuario ver los resultados de la votación antes de que el programa continúe ejecutándose.

19



Este fragmento de código maneja la opción "7" del menú principal, que es "Debate" en el programa.

Se verifica si la elección del usuario (eleccion) es igual a "7", lo que indica que el usuario eligió la opción "Debate".

Se utiliza Console.WriteLine() para mostrar un mensaje que indica que se está llamando a un legislador para debatir una propuesta.

Se solicita al usuario que ingrese el número de despacho del legislador al que desea llamar para el debate.

Luego, se utiliza un bucle foreach para iterar a través de la lista de legisladores almacenados en el objeto parlamento utilizando parlamento.getListaLegislador().

Para cada legislador en la lista, se verifica si el número de despacho del legislador coincide con numDespachoDebate. Si coincide, significa que se ha encontrado al legislador al que el usuario desea llamar para el debate.

Se verifica la cámara del legislador (legislador.getcamara()) para determinar a qué cámara pertenece y se realiza una simulación de participación en el debate utilizando el método participarDebate(debateLegislador).

Se muestra un mensaje que indica que el legislador participa en el debate y se le pide al usuario que presione Enter para continuar.

En resumen, este fragmento de código permite al usuario llamar a un legislador específico para participar en un debate y muestra un mensaje que indica que el legislador está participando en el debate.

20

**Conclusión:**

Este proyecto ha sido una experiencia desafiante y valiosa para nosotros. Nos ha permitido no solo aplicar los conocimientos adquiridos en clase, sino también profundizar en la programación orientada a objetos (POO) de una manera práctica y concreta.

A lo largo de este proyecto, hemos podido experimentar de primera mano cómo se llevan a cabo los principios de la POO en el desarrollo de software. Desde la creación de clases y objetos hasta la implementación de herencia y polimorfismo, cada aspecto de la programación orientada a objetos ha sido una parte esencial de nuestro trabajo. Esto nos ha ayudado a consolidar nuestras habilidades y comprender cómo estas técnicas pueden ser aplicadas en proyectos del mundo real.

Además de la parte técnica, también hemos aprendido sobre la importancia de la colaboración y la comunicación en un equipo. Trabajar juntos en este proyecto nos ha enseñado a coordinar esfuerzos, resolver problemas de manera conjunta y mantener una comunicación efectiva para asegurar que todos estábamos alineados en cuanto a los objetivos y las tareas a realizar.

Repositorio GitHub: https://github.com/JoseMiguelGaitan/Juan-bordabehere-Miguel-Gaitan.git

21