ANALISTA PROGRAMADOR

PRACTICO 1

RAMIRO AMARAL, FRANCISCO
MACHADO, LOIS MORALES

2023

Resumen:

En este trabajo se desarrolló un sistema legislativo mediante la programación orientada a objetos Se implementaron conceptos como: objetos, clases, métodos, herencia y polimorfismo. Se crearon tres clases para representar a los legisladores: Legislador, Senador y Diputado, las dos últimas heredan de la clase Legislador. Cada clase incluye atributos que tienen información importante de un legislador como partido político, departamento que representa, numero de despacho, nombre y apellido, edad y estado civil. Se utilizaron métodos en cada clase para establecer y obtener los valores de sus atributos, así como para determinar a qué cámara legislativa pertenece el legislador. Se crearon constructores para inicializar las estancias de las clases con valores ingresados por el usuario.

Además, se creó la clase Parlamento, que funciona como un contenedor para guardar legisladores en una lista. Esta se inicializa con el constructor de la clase y es accedida por distintos métodos para realizar operaciones como registrar nuevos legisladores, listar los legisladores con sus datos, incluyendo a que cámara pertenecen y calcular la cantidad de senadores y diputados en la lista.

Para la interacción con el usuario, se creó un menú que ofrece diversas opciones como el ingreso de legisladores (senadores y diputados), para este se implementó un proceso de entrada de datos detallados, solicitando al usuario nombre, apellido, edad, partido político, número de despacho, departamento representado y estado civil de los legisladores. ver información de las cámaras, lo cual nos mostrara datos de senadores y diputados. Otras opciones del menú son ver información de las cámaras, lo cual desplegara información de los legisladores, y obtener la cantidad de legisladores por tipo. El menú también cuenta con la opción de continuar por el programa, desde ahí podremos introducir el numero despacho del legislador que queramos, después se abrirá un menú que nos dejará elegir que acción queremos realizar con ese legislador: participar en un debate, votar, realizar una propuesta legislativa.

Índice

1. Introducción

1.1. Clases y Objetos

En la programación orientada a objetos (POO), una clase actúa como un patrón o un modelo para la creación de objetos. Establece un conjunto de atributos y métodos que pueden ser aplicados a cualquier objeto derivado de dicha clase. Un objeto representa una instancia de una clase que tiene un estado, es decir, los valores de sus atributos, y un comportamiento, que son las acciones que puede llevar a cabo mediante sus métodos.

Las clases y los objetos son componentes esenciales de la POO, y se emplean para simular y representar objetos del mundo real y sus interacciones en el software. Al definir una clase, se está definiendo una plantilla para la creación de objetos que comparten propiedades y funcionalidades comunes. Los objetos se crean a partir de estas clases y pueden colaborar entre sí y con otros objetos del sistema para ejecutar tareas específicas. [1]

1.2. Herencia

La herencia en C# es un mecanismo que posibilita la creación de nuevas clases tomando como base una clase preexistente, que se denomina clase base o superclase. A la nueva clase generada se le llama clase derivada o subclase.

La clase derivada hereda todos los elementos de la clase base y tiene la capacidad de agregar nuevos elementos o modificar los elementos heredados.

Cuando se hereda de una clase, la subclase adquiere todos los campos, métodos y propiedades de la clase base, pero tiene la flexibilidad de proporcionar implementaciones diferentes de estos elementos si es necesario.

Además de permitir la reutilización de código, la herencia también tiene el beneficio de reducir la cantidad de código que se necesita escribir al aprovechar los miembros y comportamientos de una clase existente en lugar de crearlos desde cero. [2]

1.3. Polimorfismo

El polimorfismo es un concepto clave en la programación orientada a objetos que se refiere a la capacidad de una clase hija (o subclase) de proporcionar una implementación específica de un método que ha sido definido en su clase padre (o superclase). Esto permite que un mismo método tenga múltiples comportamientos dependiendo del contexto en el que se llame.

En otras palabras, el polimorfismo nos permite definir un método en la clase base y luego sobrescribir ese mismo método en una clase derivada, de manera que ambas clases tengan un método con el mismo nombre y firma, pero con implementaciones diferentes. Este proceso de redefinir el método en la clase hija se conoce como "sobreescritura de métodos" o "anulación de métodos". [3]

2. Metodología

Para crear un programa de un sistema legislativo con (POO) se aplicaron los diferentes conceptos de esta. Las clases se utilizaron para crear los objetos que se necesitaban con sus respectivos atributos. Estuvieron presentes otros conceptos como el de herencia, utilizado para crear una clase Padre llamada Legislador, esta contenía todos los atributos (nombre, apellido, numero de despacho, etc.) y métodos (getters y setters de los atributos) en común que comparten las clases Senador y Diputado. Se hace uso del Polimorfismo para crear métodos únicos para cada tipo de legislador, como el método getCamara() que sirve para saber qué tipo de legislador es, y métodos para presentar propuestas, debatir y votar que al ser llamados debían actuar de distinta forma dependiendo de que tipo de legislador sea.

Se crea además una clase parlamento, que contendrá la lista de legisladores y métodos para registrar legisladores y para ver la cantidad de cada uno que existe en la lista.

2.1. Clase Legislador

2.1.1. Atributos

Se definen atributos para guardar los datos de los objetos que estamos creando. Para este programa se requerian los siguientes:

partidoPolitico (string)

departamentoQueRepresenta (string)

numDespacho (int)

nombre (string)

apellido (string)

edad (int)

casado (bool)

2.1.2. Constructores

En los constructores utilizamos el concepto de sobrecarga por lo que se llaman igual que la clase pero con atributos diferentes.

Legislador(string partidoPolitico, string departamentoQueRepresenta, int numDespacho, string nombre, string apellido, int edad, bool casado)

Legislador(): vacío para que podamos crear objetos a partir del ingreso del usuario.

2.1.3. Métodos

Se utilizan métodos getters y setters para todos los atributos, estos nos permiten obtener (getters) y modificar (setters) cada atributo de la clase por separado.

El polimorfismo fue utilizado en estos 4 métodos: getCamara(), presentarPropuestaLegislativa(), votar(), participarDebate(). Estos se declaran en la clase padre (Legislador) y están vacíos, para después en las clases hijos hacer lo que se llama override para que los métodos actúen diferente dependiendo la clase que los llame.

2.2. Clase Diputado y Clase Senador

Las clases Diputado y Senador heredan de la clase Legislador todos sus atributos y se les agrega uno llamado numAsientoCamaraBaja en la clase Diputado y numAsientoCamaraAlta en la clase Senador, que son únicos de cada clase.

En este caso, al haber utilizado herencia, el constructor de las clases hijas está llamando al constructor de la clase padre (Legislador) y pasándole los parámetros que se han recibido en el constructor de las hijas. Esto se hace para inicializar adecuadamente los atributos de la clase padre con los valores dados al crear un objeto de las clases Diputado y Senador.

Se utiliza además un constructor vacío para que podamos crear objetos a partir del ingreso del usuario.

2.2.1. Constructores

-Diputado(string partidoPolitico, string departamentoQueRepresenta, int numDespacho, string nombre, string apellido, int edad, bool casado, int numAsientoCamaraBaja):base(partidoPolitico, departamentoQueRepresenta,numDespacho, nombre, apellido, edad, casado)

-Diputado()

-Senador(string partidoPolitico, string departamentoQueRepresenta, int numDespacho, string nombre, string apellido, int edad, bool casado, int numAsientoCamaraBaja):base(partidoPolitico, departamentoQueRepresenta,numDespacho, nombre, apellido, edad, casado)

-Senador()

2.2.2. Métodos

En estas clases existen los métodos getNumAsiento() y setNumAsiento() que sirven para obtener y modificar el valor del atributo numAsientoCamaraAlta para Senador y numAsientoCamaraBaja para Diputado. Además existen los métodos polimórficos ya mencionados que permiten realizar diferentes acciones dependiendo que clase los utilice, estos son: getCamara(), presentarPropuestaLegislativa(), votar() y participarDebate().

2.3. Clase Parlamento

La clase parlamento es la entidad que contiene una lista que actúa como un almacén para mantener un registro de legisladores. Esta lista se configura inicialmente mediante el constructor de la clase y se utiliza a través de varios métodos para llevar a cabo diferentes tareas. Además, se declaran dos atributos de solo lectura que cuentan cuantos legisladores de cada cámara existen en la lista.

2.3.1. Métodos

En esta clase se hicieron diferentes métodos que acceden y editan la lista creada:

listarCamaras(): recorre la lista e imprime en consola el nombre, apellido, numero de despacho y cámara de cada legislador.

registrarLegislador(): añade un nuevo legislador a la lista.

cantidadLegislador(): utilizando los atributos para contar ya mencionados, imprime en consola la cantidad de senadores y la cantidad de diputados.

3. Resultados y Discusión

3.1. Programa por Consola

El programa presenta un menú principal para que el usuario elija que quiere hacer.

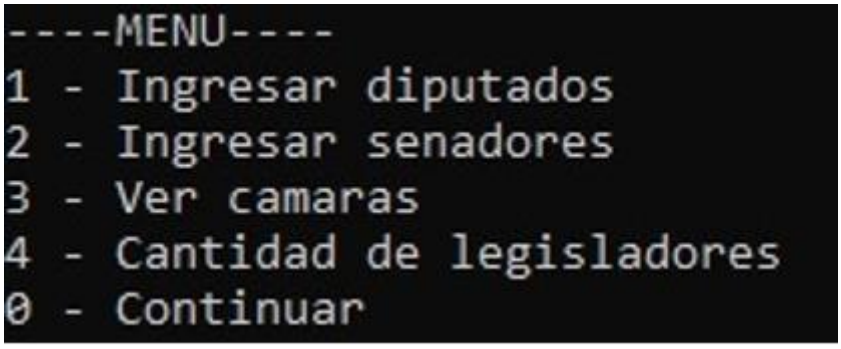

Figura 3.1.1.

La opción 1 permite al usuario ingresar un legislador tipo diputado y comienza a pedir los datos necesarios. La opción 2 actúa de la misma manera, solo que ingresa legisladores tipo senador.

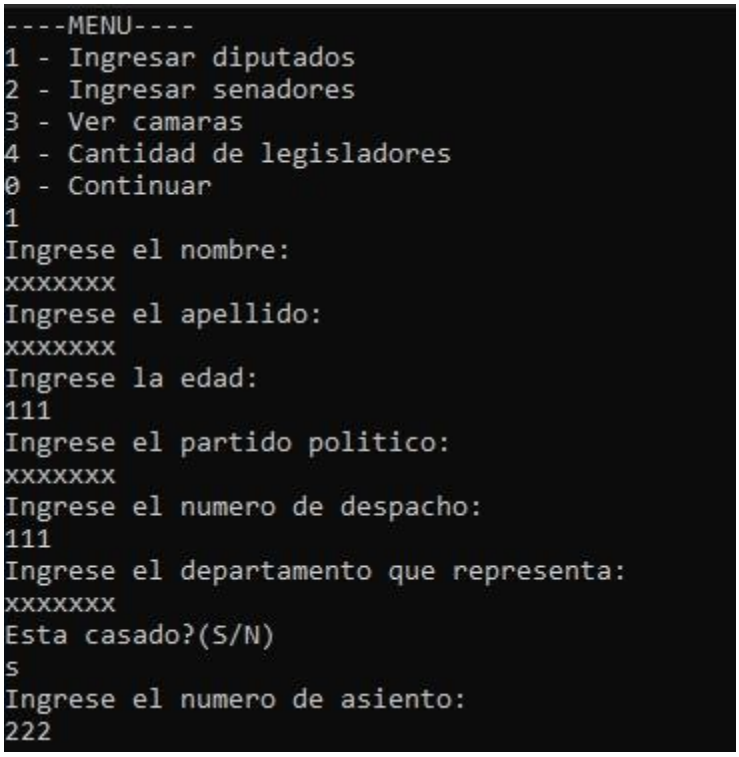

Figura 3.1.2.

La opción 3 permite al usuario ver los legisladores que están actualmente guardados en la lista.

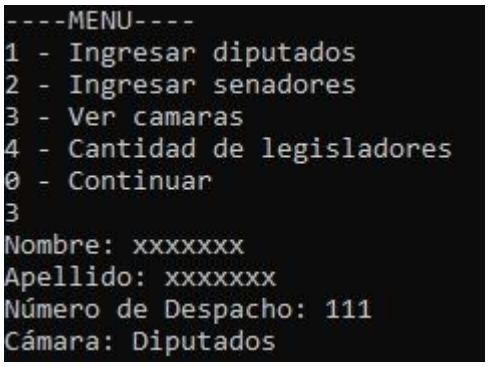

Figura 3.1.3.

La opción 4 imprime en pantalla la cantidad de legisladores por tipo.

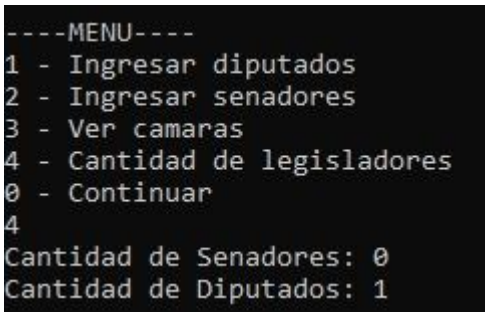

Figura 3.1.4.

La opción 0 imprime la lista de los legisladores y lleva al usuario al siguiente menú,

En este se pide el número del despacho del legislador que se elija para posteriormente desplegar las opciones de presentar una propuesta legislativa, votar y participar en un debate.

En la siguiente imagen se da un ejemplo eligiendo la opción presentar una propuesta con el legislador que se había creado.

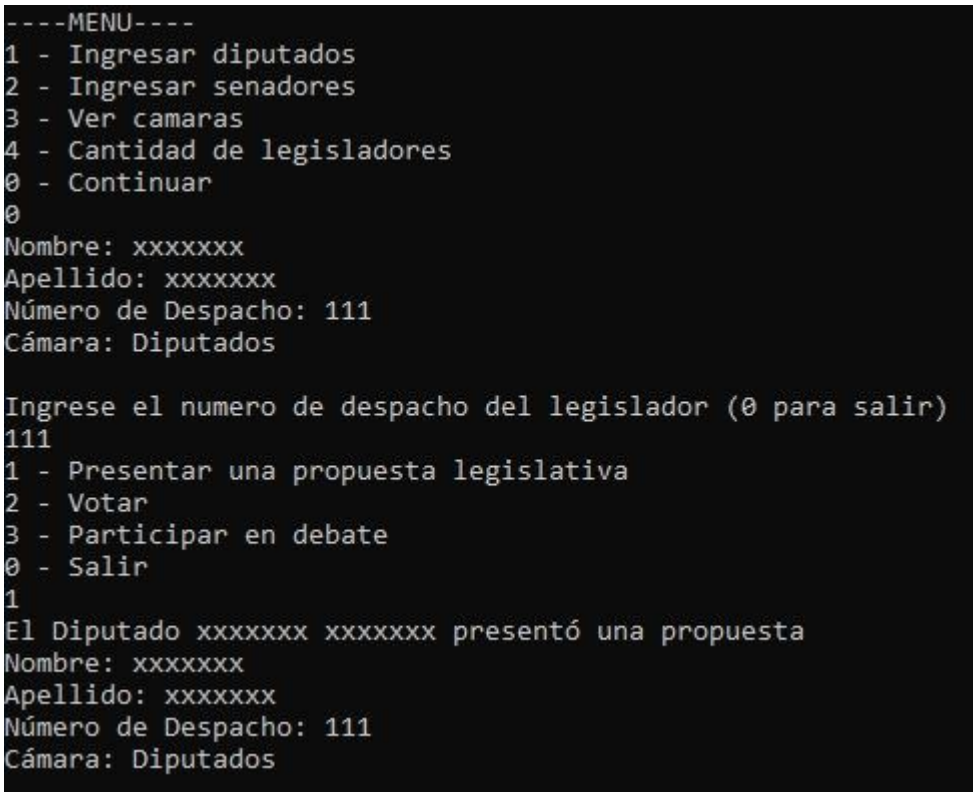

Figura 3.1.5.

3.2. Diagrama de Clases

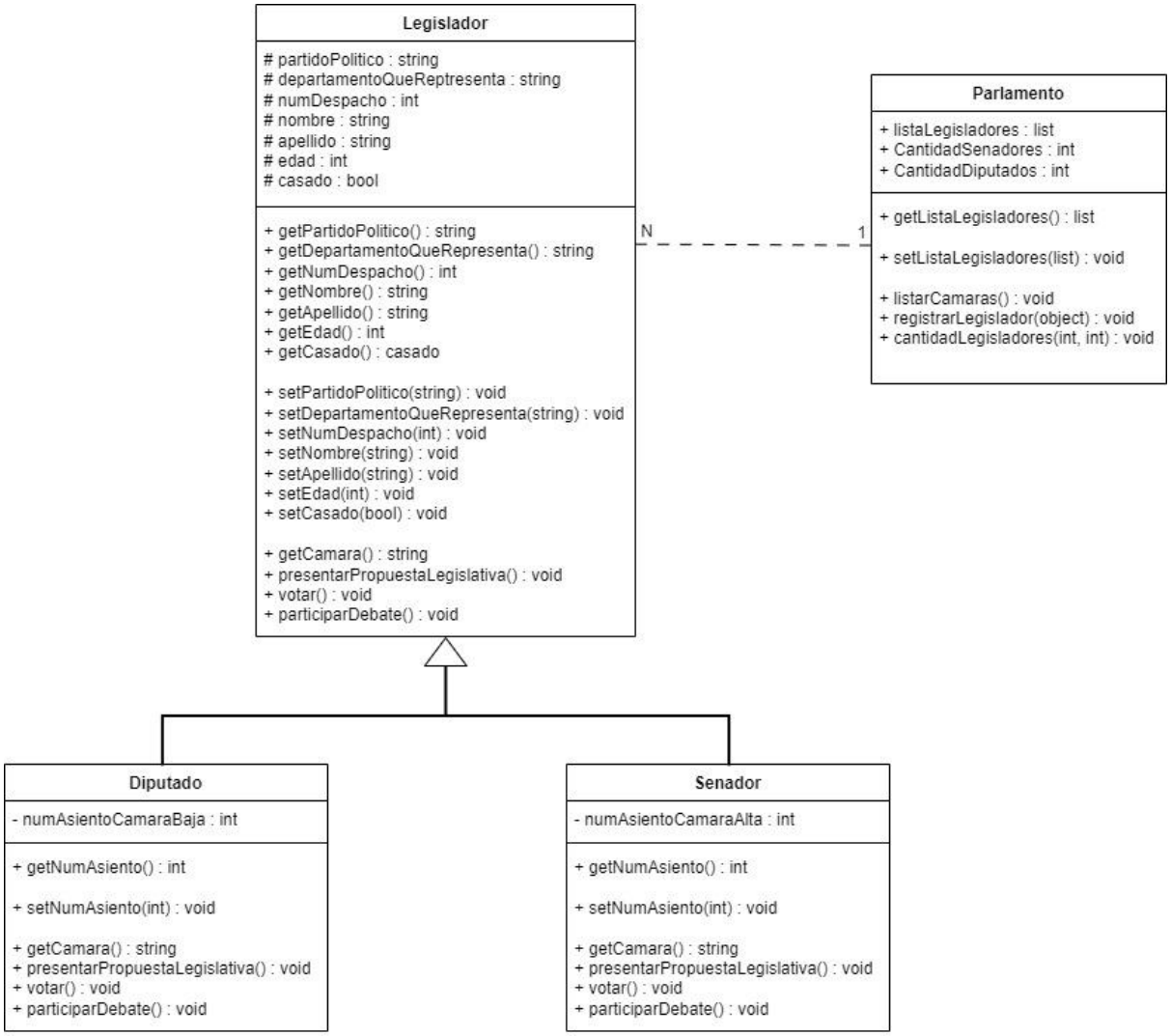

Figura 3.2.1.

4. Conclusiones

La Programación Orientada a Objetos nos permite crear entidades como los legisladores con sus atributos y métodos únicos. La herencia y las clases nos dan la posibilidad de organizar a estas entidades en diferentes categorías como Senadores y Diputados, simplificando su manejo. Las listas y colecciones funcionan como hojas de registro, donde guardamos quiénes están en el Parlamento y cuántos son de cada tipo. El menú interactivo es como la guía de usuario que ayuda a comunicarnos con el programa de manera fácil. Además, verificamos que la información ingresada por el usuario sea correcta para que no haya problemas. Los métodos polimórficos fueron de gran utilidad para asignar acciones únicas que pueden hacer los legisladores, como presentar propuestas, votar y participar en debates.

5. Referencias

[1] https://oregoom.com/c-sharp/poo/

[2] https://oregoom.com/c-sharp/herencias/

[3] https://www.netmentor.es/entrada/polimorfismo-poo