Universidad Mariano Gálvez Guatemala, Campus Jutiapa

Facultad de Ingeniería en Sistemas

Curso: Programación 1

Catedrático: Ing. Ruldin Ayala

Nombre del estudiante: Lester David Payes Méndez

Carné: 0905-24.22750

Documentación Herencia y Polimorfismo en C#

Clase Vehiculo:

Primero, creé la clase Vehiculo como un molde para todos los vehículos. Esta clase contiene las propiedades y métodos comunes que la mayoría de los vehículos tienen. Utilicé virtual para que estos métodos pudieran ser sobrescritos en las clases derivadas y protected para que las propiedades y métodos pudieran ser accesibles desde las clases derivadas. Además, utilicé un constructor para inicializar las propiedades con datos específicos desde el momento de la creación del objeto, es decir, para que el usuario no tuviera que elegir estos datos, sino que fueran definidos por mí como creador.

Propiedades:

- **Color**: El color del vehículo.
- **Modelo**: El modelo del vehículo.
- **Year**: El año de fabricación del vehículo.
- **TipoGasolina**: El tipo de gasolina que utiliza el vehículo.
- **NumeroRuedas**: El número de ruedas del vehículo.
- **CapacidadTanque**: La capacidad del tanque de combustible del vehículo.
- **NivelCombustible**: El nivel actual de combustible en el tanque.
- **Velocidad**: La velocidad actual del vehículo.
- **VelocidadMaxima**: La velocidad máxima que puede alcanzar el vehículo.
- **MarchaActual**: La marcha en la que se encuentra el vehículo.

Métodos:

- **Acelerar**: Método para aumentar la velocidad del vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.
- **CambiarMarcha**: Método para cambiar la marcha del vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.
- **Frenar**: Método para reducir la velocidad del vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.
- **RecargarCombustible**: Método para recargar el combustible del vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.
- **Encender**: Método para encender el vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.
- **Apagar**: Método para apagar el vehículo.

·**VerificarEstado**: Método para verificar el estado general del vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.

- **CambiarAceite**: Método para cambiar el aceite del vehículo. Marcado como virtual para permitir sobrescritura en clases derivadas.

Métodos Protegidos

- **IncrementarVelocidad**: Método protegido para incrementar la velocidad del vehículo.
- **DecrementarVelocidad**: Método protegido para decrementar la velocidad del vehículo.
- **IncrementarCombustible**: Método protegido para incrementar el nivel de combustible del vehículo.
- **DecrementarCombustible**: Método protegido para decrementar el nivel de combustible del vehículo.

Explicación a Detalle

Utilicé virtual en los métodos para permitir que las clases derivadas puedan sobrescribirlos con su propia implementación. Utilicé protected en las propiedades y métodos que deben ser accesibles desde las clases derivadas, pero no desde fuera de la jerarquía de herencia. Esto permite que las clases derivadas como AutoDeCombustion, Camion, y Motocicleta puedan personalizar el comportamiento de estos métodos según sea necesario.

Además, el constructor de la clase Vehiculo inicializa las propiedades con datos específicos desde el momento de la creación del objeto, asegurando que los vehículos tengan valores predeterminados definidos por mí como creador.

Clase AutoDeCombustion

Creé la clase AutoDeCombustion derivada de la clase Vehiculo. Esto permite que AutoDeCombustion utilice las funciones y propiedades de Vehiculo como también modificar funciones o permitirme implementar polimorfismo cuando sea necesario, como por ejemplo en el método de cambiar marchas.

Propiedades Adicionales:

- **EsDescapotable**: Indica si el auto es descapotable.
- **TieneMaletero**: Indica si el auto tiene maletero.

- **PuertasBloqueadas**: Un array que indica el estado de bloqueo de las puertas del auto.

Ejemplos de Encapsulación

- **EsDescapotable**: Utilicé private para proteger esta propiedad.
- **TieneMaletero**: Utilicé private para proteger esta propiedad.
- **PuertasBloqueadas**: Utilicé private para proteger esta propiedad.

Métodos Adicionales

- **Descapotar**: Método para descapotar el auto si es descapotable.
- **UsarMaletero**: Método para usar el maletero del auto si tiene uno.
- **BloquearPuertas**: Método para bloquear todas las puertas del auto.
- **DesbloquearPuertas**: Método para desbloquear todas las puertas del auto.
- **Acelerar**: Sobrescribe el método Acelerar de la clase base para incluir lógica específica de autos de combustión.
- **Frenar**: Sobrescribe el método Frenar de la clase base para incluir lógica específica de autos de combustión.
- **RecargarCombustible**: Sobrescribe el método RecargarCombustible de la clase base para incluir lógica específica de autos de combustión.
- **InformacionAutoDeCombustion**: Método para mostrar información específica del auto de combustión.
- **CambiarMarcha**: Sobrescribe el método CambiarMarcha de la clase base para incluir lógica específica de autos de combustión.

Explicación a detalle

La clase AutoDeCombustion hereda de Vehiculo y añadí propiedades y métodos específicos para autos de combustión. Utilicé el modificador override para sobrescribir los métodos de la clase base y proporcionar implementaciones específicas para esta clase. Además, añadí métodos adicionales como Descapotar, UsarMaletero, BloquearPuertas, y DesbloquearPuertas que son exclusivos para los autos de combustión.

El constructor de AutoDeCombustion inicializa las propiedades específicas del auto de combustión, además de las propiedades heredadas de la clase Vehiculo. Esto asegura que cada instancia de AutoDeCombustion tenga valores predeterminados definidos por mí como creador.

Clase Camion

Luego, creé la clase Camion derivada de la clase Vehiculo. Esto permite que Camion utilice las funciones y propiedades de Vehiculo y también modifique funciones o implemente polimorfismo cuando sea necesario, como por ejemplo en el método de cambiar marchas. Esta clase específica para camiones incluí propiedades y métodos adicionales que son exclusivos para este tipo de vehículo.

Propiedades Adicionales

- **CapacidadCarga**: La capacidad máxima de carga del camión.
- **CargaActual**: La carga actual que lleva el camión.
- **TamañoPlataforma**: El tamaño de la plataforma del camión.
- **TieneContenedor**: Indica si el camión tiene un contenedor.

Ejemplos de Encapsulación

- **CapacidadCarga**: Utilicé private para proteger esta propiedad.
- **CargaActual**: Utilicé private para proteger esta propiedad.
- **TamañoPlataforma**: Utilicé private para proteger esta propiedad.

Métodos Adicionales

- **Cargar**: Método para cargar peso en el camión.
- **Descargar**: Método para descargar peso del camión.
- **Acelerar**: Sobrescribe el método Acelerar de la clase base para incluir lógica específica de camiones.
- **Frenar**: Sobrescribe el método Frenar de la clase base para incluir lógica específica de camiones.
- **Encender**: Sobrescribe el método Encender de la clase base para incluir lógica específica de camiones.
- **CambiarMarcha**: Sobrescribe el método CambiarMarcha de la clase base para incluir lógica específica de camiones.
- **InformacionDelVehiculo**: Sobrescribe el método InformacionDelVehiculo de la clase base para mostrar información específica del camión.

Explicación más a Detalle

La clase Camion hereda de Vehiculo añade propiedades y métodos específicos que solo camiones tiene. Utilicé el modificador override para sobrescribir los métodos de la clase base y proporcionar implementaciones específicas para esta clase. Además, añadí métodos adicionales como Cargar y Descargar que son exclusivos para los camiones.

El constructor de Camion inicializa las propiedades específicas del camión, además de las propiedades heredadas de la clase Vehiculo. Esto asegura que cada instancia de Camion tenga valores predeterminados definidos por mí como creador.

Utilicé polimorfismo en los siguientes métodos:

- **Acelerar**: Para ajustar la lógica de aceleración específica de los camiones.
- **Frenar**: Para ajustar la lógica de frenado específica de los camiones.
- **Encender**: Para ajustar la lógica de encendido específica de los camiones.
- **CambiarMarcha**: Para ajustar la lógica de cambio de marchas específica de los camiones.
- **InformacionDelVehiculo**: Para mostrar información específica del camión.

Clase Motocicleta

Luego, creé la clase Motocicleta derivada de la clase Vehiculo. Esto permite que Motocicleta utilice las funciones y propiedades de Vehiculo y también modifique funciones o implemente polimorfismo cuando sea necesario, como por ejemplo en el método de cambiar marchas. En esta clase específica para motocicletas incluí propiedades y métodos adicionales que son exclusivos para este tipo de vehículo.

Propiedades Adicionales

- **TipoManillar**: El tipo de manillar que tiene la motocicleta.
- **TieneMaletero**: Indica si la motocicleta tiene maletero.
- **Cilindrada**: La cilindrada del motor de la motocicleta.

Ejemplos de Encapsulación

- **TipoManillar**: Utilicé private para proteger esta propiedad.
- **TieneMaletero**: Utilicé private para proteger esta propiedad.

- **Cilindrada**: Utilicé private para proteger esta propiedad.

Métodos Adicionales

- **ArrancarConPata**: Método para arrancar la motocicleta con la pata de arranque.
- **UsarMaletero**: Método para usar el maletero de la motocicleta si tiene uno.
- **Acelerar**: Sobrescribe el método Acelerar de la clase base para incluir lógica específica de motocicletas.
- **Frenar**: Sobrescribe el método Frenar de la clase base para incluir lógica específica de motocicletas.
- **RecargarCombustible**: Sobrescribe el método RecargarCombustible de la clase base para incluir lógica específica de motocicletas.
- **InformacionDelVehiculo**: Sobrescribe el método InformacionDelVehiculo de la clase base para mostrar información específica de la motocicleta.
- **CambiarMarcha**: Sobrescribe el método CambiarMarcha de la clase base para incluir lógica específica de motocicletas.

Explicación a Detalle

La clase Motocicleta se hereda de Vehiculo y añadí propiedades y métodos específicos para motocicletas. Utilicé el modificador override para sobrescribir los métodos de la clase base y proporcionar implementaciones específicas para esta clase. Además, añadí métodos adicionales como ArrancarConPata y UsarMaletero que son exclusivos para las motocicletas.

El constructor de Motocicleta inicializa las propiedades específicas de la motocicleta, además de las propiedades heredadas de la clase Vehiculo. Esto asegura que cada instancia de Motocicleta tenga valores predeterminados definidos por mí como creador.

Program.cs

Documentación de la Clase Program

La clase Program ha sido modularizada para mejorar la claridad y la organización del código. A continuación, se detallan las funciones creadas y su propósito:

Funciones del Menú Principal

1. **MostrarMenuPrincipal**
    - Esta función muestra el menú principal y maneja la selección del usuario.

- Llama a MostrarOpcionesMenuPrincipal para mostrar las opciones del menú.
- Llama a ProcesarOpcionMenuPrincipal para procesar la opción seleccionada por el usuario.
- Contiene un bucle do-while que se ejecuta hasta que el usuario selecciona la opción de salir (opción 4).

2. **MostrarOpcionesMenuPrincipal**

   Esta función muestra las opciones del menú principal.

3. **ProcesarOpcionMenuPrincipal**

   - Esta función procesa la opción seleccionada por el usuario en el menú principal.
   - Llama a las funciones UsarAutoDeCombustion, UsarCamion o UsarMoto según la opción seleccionada.

Funciones del Menú de Vehículos

4. **MostrarMenuVehiculo**

   - Esta función muestra el menú de opciones para un vehículo específico y maneja la selección del usuario.
   - Llama a MostrarOpcionesMenuVehiculo para mostrar las opciones del menú del vehículo.
   - Llama a ProcesarOpcionMenuVehiculo para procesar la opción seleccionada por el usuario.

5. **MostrarOpcionesMenuVehiculo**

   - Esta función muestra las opciones del menú para un vehículo específico.
   - Muestra opciones adicionales según el tipo de vehículo (AutoDeCombustion, Camion, Motocicleta).

6. **ProcesarOpcionMenuVehiculo**

   - Esta función procesa la opción seleccionada por el usuario en el menú del vehículo.
   - Llama a los métodos correspondientes del vehículo según la opción seleccionada.

Funciones para Usar Vehículos

7. **UsarAutoDeCombustion**
    - Crea una instancia de AutoDeCombustion y llama a MostrarMenuVehiculo con esta instancia.

8. **UsarCamion**
    - Crea una instancia de Camion y llama a MostrarMenuVehiculo con esta instancia.

9. **UsarMoto**
    - Crea una instancia de Motocicleta y llama a MostrarMenuVehiculo con esta instancia.

Resumen de Program.cs

En resumen, he modularizado la clase Program creando funciones específicas para mostrar y procesar las opciones del menú principal y del menú de vehículos. Esto lo hice para mejorar la organización del código. Las funciones que cree permiten manejar de manera clara y estructurada las diferentes opciones y acciones disponibles para los vehículos en el sistema. La función principal que encierra a todas las demás funciones es MostrarMenuPrincipal, que contiene un bucle do-while que se ejecuta hasta que el usuario selecciona la opción de salir.

Conclusión General

Clases Derivadas

Cree tres clases derivadas de la clase Vehiculo:

- **AutoDeCombustion**
- **Motocicleta**
- **Camion**

En cada clase cumplí con lo que decían las instrucciones, añadiendo propiedades adicionales y aplicando encapsulación para proteger los datos.

Propiedades Adicionales y Encapsulación

En cada clase derivada, añadí tres propiedades adicionales o más y se aplicó encapsulación para proteger los datos.

Sobreescritura de Métodos

Sobrescribí varios métodos de la clase base Vehiculo en las clases derivadas para implementar comportamientos específicos.

Modularización del Código en la Clase Programa

Modularice la en la clase Program del Program.cs para mejorar la claridad y la organización del código:

- **MostrarMenuPrincipal**: Muestra el menú principal y maneja la selección del usuario.
- **MostrarOpcionesMenuPrincipal**: Muestra las opciones del menú principal.
- **ProcesarOpcionMenuPrincipal**: Procesa la opción seleccionada por el usuario en el menú principal.
- **UsarAutoDeCombustion**, **UsarCamion**, **UsarMoto**: Crean instancias de los vehículos correspondientes y llaman a MostrarMenuVehiculo.
- **MostrarMenuVehiculo**: Muestra el menú de opciones para un vehículo específico y maneja la selección del usuario.
- **MostrarOpcionesMenuVehiculo**: Muestra las opciones del menú para un vehículo específico.
- **ProcesarOpcionMenuVehiculo**: Procesa la opción seleccionada por el usuario en el menú del vehículo.

En resumen, esta tarea me permitió aplicar y reforzar los conceptos de herencia y polimorfismo en C#, así como mejorar mis habilidades en la organización y a la hora de modularizar código. La creación de una jerarquía de clases derivadas de Vehiculo y la implementación de métodos específicos para cada tipo de vehículo han sido fundamentales para que pudiera entender de mejor manera cómo se pueden extender y personalizar las funcionalidades en la programación orientada a objetos.