

**Universidad Abierta y a Distancia**

**de México**

**División de Ciencias Exactas, Ingeniería y Tecnología**

**Desarrollo de software**

**Semestre:** Sexto Semestre

**Asignatura:** Programación net II

**Unidad de aprendizaje: Herencia y polimorfismo en el lenguaje de programación C Sharp**

**Evidencia de Aprendizaje: Herencia, Polimorfismo y sobreescritura.**

**Nombre del estudiante:** Francisco González Antonio

**Matrícula:** ES1821009465

**Grupo: DS-DPRN2-2202-B2-003**

**Docente: José Francisco Rico Gallegos**

**Fecha de entrega: 2 de Noviembre de 2022**

**Ciudad de México, 2 de Noviembre del 2022**

Diseño: DL-CPL

**ÍNDICE**

[**INTRODUCCIÓN** 5](#_Toc80333573)

[**DESARROLLO DE LA ACTIVIDAD** 6](#_Toc80333574)

[**CONCLUSIONES** 7](#_Toc80333575)

[**REFERENCIAS BIBLIOGRÁFICAS** 8](#_Toc80333576)

# **INTRODUCCIÓN**

Como parte del aprendizaje de esta unidad es necesario unir los conceptos de herencia y polimorfismo aprendidos dentro de esta unidad, para ello en esta actividad se simuló una Armadora, la cual tiene una clase Abstracta llamada Vehículo y tres clases Hijas Auto, Camioneta y Van, cada una de ellas tiene asignado un costo total con base en las piezas que requiere para su construcción. Adicional dentro de este desarrolló se creó una Clase llamada Armadora que se encarga de gestionar la creación a conveniencia con base en un presupuesto mensual asignado, el cual aumenta cada mes.

Los conceptos más importantes que utilicé en esta actividad son el de Herencia, Polimorfismo, Sobreescritura y Abstracción, adicional utilicé algunos conceptos vistos en la unidad anterior como el encapsulamiento mediante la sentencia get y set de C-Sharp.

# **DESARROLLO DE LA ACTIVIDAD**

Esta actividad a diferencia de las realizadas anteriormente tiene cierta particularidad y es que no se indica la implementación, sólo se indican los métodos y el cómo se deben relacionar las clases, por lo que la implementación de manera correcta queda a cargo de mí, por ello considero importante explicar algunos puntos importantes sobre el código y el porqué se programó de dicha forma. A continuación muestro y explico cada unos de los apartados a evaluar dentro del código solución, dicho código está incluido en esta carpeta compartida y está igualmente documentado.

**Programa la clase Vehículo**

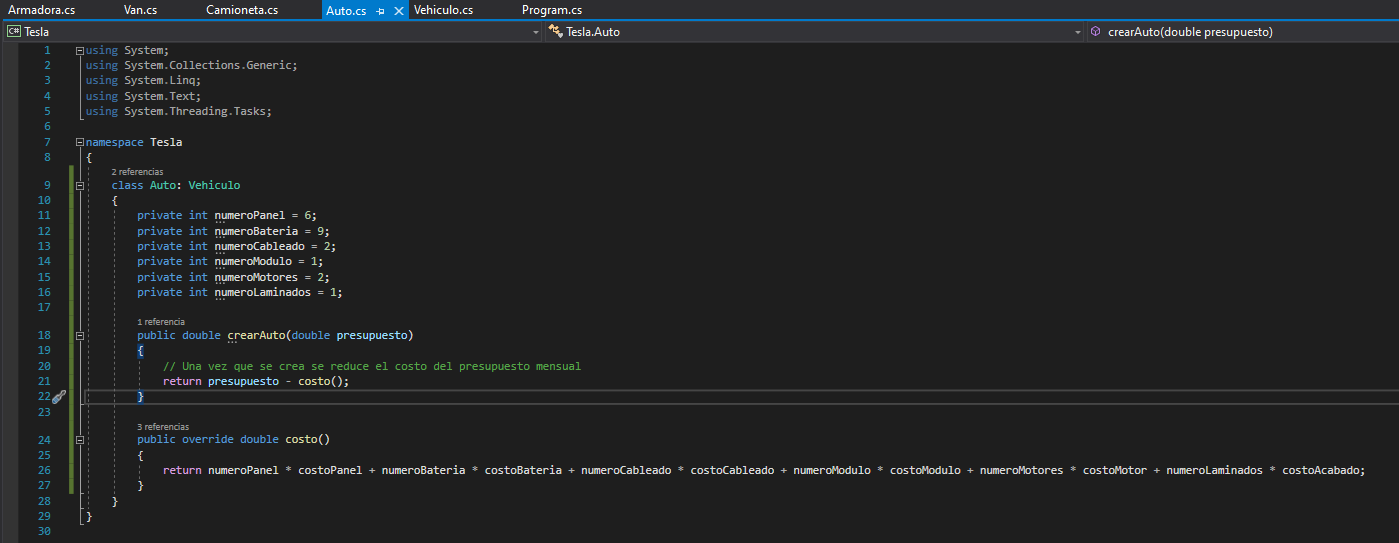
La clase Vehículo guarda los costos de los diversos componentes, lo que ayuda a centralizar dichos costos, en caso de que cambien sólo se mueve esta clase pues todos los Vehículos heredados utilizan estos mismos componentes. Algo en común para todos es el costo, por lo que se permite la abstracción para que cada quien implemente como mejor le convenga.

Text

Description automatically generated

**Clase Auto con sus métodos para Crear y Costo**

Aquí la clase Auto hereda de Vehículo y sobrescribe el costo con base en los datos de costos de la clase Padre



**Clase Camioneta con sus métodos para Crear y Costo**

A screenshot of a computer

Description automatically generated with medium confidenceCaso similar a la clase auto, se sobrescribe costo y se implementa la creación.

**Clase Van con sus métodos para Crear y Costo**

A screenshot of a computer

Description automatically generated with medium confidence

**Clase Armadora que se encarga de la gestión, recalcando el método piezas y explicando el cómo funciona**

La clase Armadora se crea con base en el presupuesto base asignado (para nuestro caso 950,000 el mes de Enero), con ello con base en el mes se va aumentando el presupuesto base en 15%

Text

Description automatically generated

Ahora bien, el método piezas es quizá el más complejo. Para determinar la cantidad óptima de autos identifiqué que es necesario poner un tope mínimo para la creación de ciertos vehículos, ya que en caso de no determinar un mínimo de vehículos el óptimo se transforma inmediatamente en el número máximo de vehículos del que es más barato, por lo cual para resolver dicho dilema consideré que se deben crear los autos en la siguiente secuencia Auto – Camioneta – Van, y adicional para evitar el aumento de complejidad de los cálculos una vez que el presupuesto mensual ya no alcanza para construir más vehículos este se vuelve cero, lo que técnicamente significa que ya no se puede utilizar el siguiente mes, esto para evitar transformar esto en un problema de Evaluación de Costos o Problema de la Mochila cuando viene a ser un problema más de Programación.

Al final el método piezas nos dice cuántos autos se construyeron por mes.

Text

Description automatically generated

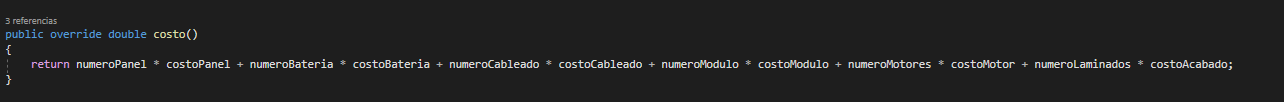
Text

Description automatically generated

Adicional hay un método llamado esPosible el cuál con base en el presupuestoMensual y el costo de fabricación del siguiente Vehículo determina si alcanza o no para su construcción siguiendo la Secuencia Auto – Camioneta – Van.

**Aplicación del comando override dentro de la implementación**

Para cada uno de las clases que heredan de Vehículo se utiliza override para sobrescribir el método de costo:



**Menú informativo**

Más que un menú muestra el cálculo de cuánto dinero habrá este años por mes y cuánto Vehículos de cada tipo se crearán por mes respetando la secuencia de creación Auto – Camioneta – Van

A screenshot of a computer

Description automatically generated

**Ir al Anexo A para revisar la información referente al código, pruebas, implementación y evidencias.**

# **CONCLUSIONES**

Este caso de estudio va muy de la mano con lo aprendido en la unidad de diversas razones, para mí la más sobresaliente es que aquí al hacer uso de la clase Abstracta Vehículo se muestra de forma rápida el cómo podemos encapsular información pues en esta clase se encuentra el costo por cada uno de los componentes utilizados, más aún, cada clase hija determina sus costos con base en esta información y el método abstracto para costo, que cada clase heredada se encarga de sobrescribir. Considero que esta actividad aporta experiencia a mi desarrollo dentro de C# ya que hice uso de los conceptos vistos en la unidad y adicional a ello la implementación fue de cierta forma libre, permitiendo programar los métodos con base en mi interpretación pero con las funcionalidades esperadas, lo que considero se apega mucho a una realidad en el mercado, ya que como desarrollador de software se deben traducir los requerimientos en funcionalidades esperadas.

# **REFERENCIAS BIBLIOGRÁFICAS**

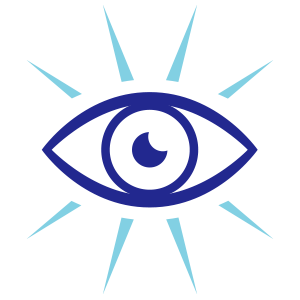
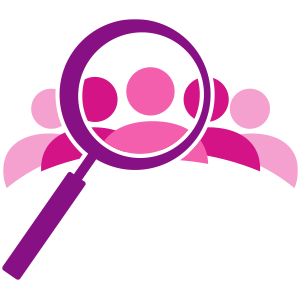
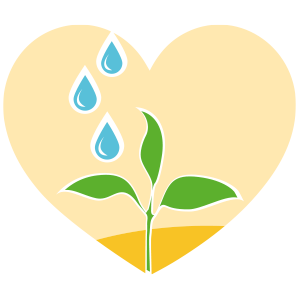
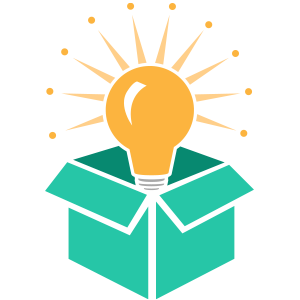
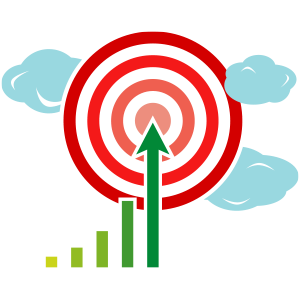
C. Wille; C#, Prentice Hall. 2001. ISBN 0-672-32037-1.

Deitel (2004). “Instructor Manual for Simply C#”. Pearson Education

Tanenbaum, A. (2009). “Sistemas Operativos Modernos”. 3 ed. México. Pearson Education

# **ANEXO A**

**Nombre del software**



La compañía será beneficiada pues ahora puede hacer proyecciones y conocer costos rápidamente.

Los objetivos y metas es saber el costo total y el número de vehículos a fabricar con base en el presupuesto base asignado, el cual se sabe que aumenta en 15% mensual.

Es un Software para armadora.

Lo destaca la automatización de las proyecciones.

Principales funciones son la herencia de la clase abstracta Vehículo, determinar el presupuesto por mes y las proyecciones de fabricación.

Lo diferencia la sencillez de implementación

Estamos intentando solucionar la automatización de la creación de diversos tipos de Vehículos con base en un presupuesto. Además se intenta abstraer lo más posible la creación para que se puedan crear diversos vehículos basados en los mismos componentes

El beneficio del software es la facilidad de creación y el pronóstico de presupuesto y proyección de piezas a crear

Esto va dirigido a quien desea implementar diversos autos con base en los mismo componentes.

Apunta a clientes y usuarios de la Armadora que quieran realizar pronósticos.

El propósito es abstraer los diversos vehículos de Tesla y facilitar su implementación. Adicional determinar cuántos vehículos se van a elaborar por mes

Necesidades

¿Qué problema estamos tratando de solucionar?

¿Qué beneficios les daría el software a las personas que tienen ese problema?

¿A qué Mercado o a qué segmento del Mercado apunta este producto?

¿Quiénes son los clientes y usuarios a los que apunta?

¿Cuál es el propósito de crear este producto?

¿Qué cambios positivos debería brindar?

VISION

Grupo objetivo

¿Cómo se va a beneficiar la compañía por construir este producto?

¿Cuáles son los objetivos y metas?

¿Qué producto es?

¿Qué hace que se destaque?

¿Cuáles son sus principales funciones?

¿Qué lo diferencia?

PRODUCTO

Alcance

**Historias de usuario y criterios de aceptación**

Nota. Generar al menos una historia de usuario por funcionalidad programada

|  |  |  |
| --- | --- | --- |
| **#XX** | Como **Gestor de Armadora** quiero **Crear los diversos Vehículos, realizar proyecciones de presupuesto y saber cuántos vehículos debo armar y de que tipo cada mes** para poder **llevar a cabo una planeación anual** | |
| **CRITERIOS DE ACEPTACIÓN** | | |
| 1. | **Creación de los diversos Vehículos disponibles** | En caso que **necesitar la construcción de un tipo de vehículo** y adicionalmente **la creación de cualquier tipo de vehículo que se disponga**, cuando **la armadora lo solicite**, el sistema **deberá permitir la creación.** |
| 2. | **Proyección del presupuesto por cada mes** | En caso que **se requiera la proyección de presupuesto cada mes** y adicionalmente **se requiera mostrar por pantalla dicha información**, cuando **se requiera la información por año**, el sistema **imprimirá el presupuesto por mes con base en el presupuesto base y el crecimiento del 15% proyectado.** |
| 3. | **Proyección de la creación de Vehículos en un mes específico (en este caso Abril o 4)** | En caso que **se quiera saber el total de Vehículos óptimos por mes**, el sistema **debe decir cuántos son el óptimo con base en las reglas proporcionadas.** |
| 4. | **Proyección anual de la creación de vehículos por mes** | En caso que **se requiera saber cuántos vehículos de cada tipo se van a crear** el sistema **debe mostrar cuánto Autos, Camionetas y Vans se crean por mes** |

**Diagramas**

• Diagrama de clases **(Obligatorio DPRN2)**

Diagram

Description automatically generated

**Herramientas del lenguaje a utilizar y descripción de la solución**

Para resolver este problema se utilizó la Herencia, el polimorfismo y la sobre escritura en el lenguaje C# mediante Visual Studio. Se creó la clase Padre para heredar a tres clases hijas y adicional se creó una clase Armadora que se encarga de determinar el número óptimo de Vehículos a crear por mes, lo que se menciona en los requerimientos como la parte final del desarrollo.

**Pruebas realizadas**

Matriz de pruebas realizadas, se deben incluir casos esperados y no esperados, **basados en los criterios de aceptación de cada historia de usuario**, para mostrar la funcionalidad del código presentado.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TÍTULO DE LA PRUEBA** | | | |  |  | | |  | | | | **FECHA DE LA PRUEBA** |
| Funcionamiento óptimo de la Armadora | | | | | | | | | | | | 02/11/22 |
| **DESCRIPCIÓN DE LA PRUEBA** | | | | | **PRUEBA DISEÑADA POR** | | |  | | | | **FECHA DE EJECUCIÓN** |
| Se realiza la creación de vehículos y se determina la cantidad óptima a producir por mes | | | | | Francisco González Antonio | | | | | | | 02/11/22 |
|  | | | | | | | | | | | | |
| **DESCRIPCIÓN DE LA PRUEBA** | | **DEPENDENCIAS DE PRUEBA** | | | | **CONDICIONES DE PRUEBA** | | | | **CONTROL DE PRUEBAS** | | |
| Se verifica el correcto funcionamiento de la Armadora para verificar la correcta creación de los diversos vehículos, cálculo de costos, proyección anual y presupuesto anual. | | Sólo la dependencia entre las clases del código. | | | | Condiciones de operación simuladas. | | | | Se identifica el resultado esperado. | | |
|  | | | | | | | | | | | | |
| **ID DE HISTORIA / CRITERIO ACEPTACIÓN** | **DESCRIPCIÓN** | **ID IMAGEN** | **RESULTADOS ESPERADOS** | | | | **RESULTADOS REALES** | | **APROBADA/REPROBADA** | | **NOTAS ADICIONALES** | |
| 1 | Creación de diversos vehículos | 001, 002 | Se puedan crear tres vehículos, pero recordemos que es necesario declarar un presupuesto inicial, ya que al ser una armadora si no hay presupuesto no se pueden crear vehículos. | | | | Creación de tres vehículos descontando el presupuesto | | Aprobada | | Éxito | |
| 2 | Proyección de presupuesto durante el año con base en el 15% de aumento | 003,004 | Se proyecta en consola el presupuesto | | | | Presupuesto por mes | | Aprobada | | Éxito | |
| 3 | Proyección de la creación de vehículos en el mes de Abril | 005,006 | Se ingresa el valor de 4 y nos da el resultado de lo que se espera crear para ese mes | | | | Salida de el número de Autos, Camionetas y Vans con base en las reglas determinadas | | Aprobada | | Éxito | |
| 4 | Proyección de la creación de vehículos en el año | 007,008 | Salida en consola de cuántos vehículos por mes. | | | | Se muestra el resultado de vehículos por mes | | Aprobada | | Éxito | |

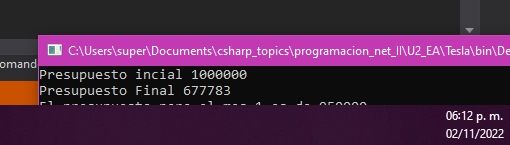
Agregar imágenes aquí con su identificador correspondiente (ID)

001 Código

Text

Description automatically generated

002 Resultado

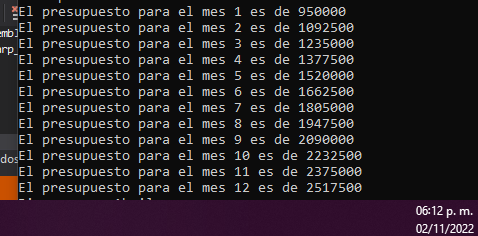


003 Código

Text

Description automatically generated

004 Resultado



005 Código

Text

Description automatically generated

006 Resultado

Text

Description automatically generated with medium confidence

007 Código

Text

Description automatically generated

008 Resultado

Graphical user interface, text

Description automatically generated