Conceptos varios de Programación OO

La programación orientada a objetos

- ¿Por qué Orientación a Objetos (OO)?
  - Se parece más al mundo real
  - Permite representar modelos complejos
  - Muy apropiada para aplicaciones de negocios
  - Las nuevas plataformas de desarrollo la han adoptado (Java / .NET)

Visual Studio

¿Qué es un Objeto?

- Informalmente, un objeto representa una entidad del mundo real
- Entidades Físicas
  - (Ej.: Vehículo, Casa, Producto)
- Entidades Conceptuales
  - (Ej.: Proceso Químico, Transacción Bancaria)
- Entidades de Software
  - (Ej.: Lista Enlazada, Interfaz Gráfica)

Visual Studio

¿Qué es un Objeto?

- Definición Formal (Rumbaugh):
  - "Un objeto es un concepto, abstracción o cosa con un significado y límites claros en el problema en cuestión"
- Un objeto posee (Booch):
  - Estado
  - Comportamiento
  - Identidad

Visual Studio

Un objeto posee Estado

- ***Lo que el objeto sabe***
- El estado de un objeto es una de las posibles condiciones en que el objeto puede existir
- El estado normalmente cambia en el transcurso del tiempo
- El estado de un objeto es implementado por un conjunto de propiedades (atributos), además de las conexiones que puede tener con otros objetos

Visual Studio

Un objeto posee Comportamiento

- ***Lo que el objeto puede hacer***
- El comportamiento de un objeto determina cómo éste actúa y reacciona frente a las peticiones de otros objetos
- Es modelado por un conjunto de mensajes a los que el objeto puede responder (operaciones que puede realizar)
- Se implementa mediante métodos

Visual Studio

¿Qué es una Clase?

- Una clase es una descripción de un grupo de objetos con:
  - Propiedades en común (atributos)
  - Comportamiento similar (operaciones)
  - La misma forma de relacionarse con otros objetos (relaciones)
  - Una semántica en común (significan lo mismo)
- Una clase es una abstracción que:
  - Enfatiza las características relevantes
  - Suprime otras características (simplificación)
- Un objeto es una instancia de una clase

Visual Studio

Ejemplo de una Clase

- Clase: Curso
- Estado (Atributos)
  - Nombre
  - Ubicación
  - Días Ofrecidos
  - Horario de Inicio
  - Horario de Término
- Comportamiento (Métodos)
  - Agregar un Alumno
  - Borrar un Alumno
  - Entregar un Listado del Curso
  - Determinar si está Completo

Visual Studio

Modificadores de Acceso

- Permiten definir el nivel de acceso (visibilidad) de los miembros (atributos o métodos) de una clase
  - Publico: Cualquier clase puede "ver" los miembros públicos de otra clase
  - Privado: Sólo la clase puede ver sus propios miembros privados
- Existen otros dos modificadores para propósitos específicos (Paquete, Protegido)

Visual Studio

Pilares de la Orientación a Objetos

- Abstracción
- Polimorfismo
- Herencia
- Encapsulamiento

Visual Studio

Abstracción

- Ignorancia Selectiva
  - La abstracción nos ayuda a trabajar con cosas complejas
  - Se enfoca en lo importante
  - Ignora lo que no es importante (simplifica)
- Una clase es una abstracción en la que:
  - Se enfatizan las características relevantes
  - Se suprimen otras características
- Una clase debe capturar una y solo una abstracción clave

Visual Studio

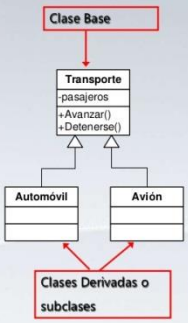
Polimorfismo

- Es la propiedad que tienen los objetos de permitir invocar genéricamente un comportamiento (método) cuya implementación será delegada al objeto correspondiente recién en tiempo de ejecución
- El polimorfismo tiende a existir en las relaciones de herencia, pero no siempre es así

Visual Studio

Herencia

- Es una relación entre clases en la cual una clase comparte la estructura y comportamiento definido en otra clase
- Cada clase que hereda de otra posee:
  - Los atributos de la clase base además de los propios
  - Soporta todos o algunos de los métodos de la clase base
- Una subclase hereda de una clase base



Visual Studio

Encapsulamiento

- Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.

Visual Studio

Introducción a la Programación Orientada a Objetos

Introducción

Emprender un proyecto de desarrollo de software, es un reto que nos lleva a pensar en la elaboración de un producto final que cumpla con las características de un software exitoso: Solido, robusto, confiable, escalable, interoperable y lo mejor de todo para quienes estamos desarrollando esgenerar código que pueda ser reutilizado.

Bajo esta premisa surge la necesidad de trabajar con una filosofía de programación orientada a objetos. Ahora bien, en este momento se preguntaran, ¿una filosofía?, la respuesta es sencilla, antes de aprender a programar en un lenguaje orientado a objeto como C# o Vb.net, es importante aprender a pensar, bajo un modelo de desarrollo, con su teoría y su metodología, como lo es la programación orientada a objetos, para encontrar una solución a un problema que se plantee.

Conceptos Fundamentales

Clase:

Vamos a iniciar este concepto con un ejemplo: Para construir un modelo de automóvil se deben haber plasmado una serie de características y funcionalidades que estarán guiando la construcción del mismo. Cada vez que se procede a crear este mismo modelo de auto, se deberá cumplir con las características y funcionalidades predefinidas.

En programación orientada a objeto, una clase define las características y comportamientos comunes de los objetos, en otras palabras la clase es el molde para la creación de los mismos. Para nuestro ejemplo, un plano del modelo del auto es el símil de la clase auto. Aunque la clase especifica las características propias del objeto cada implementación es única tal como sucede en el mundo real pues el valor de sus atributos puede variar, por ejemplo el color, kilómetros recorridos, etc.

Objeto:

Los objetos tienen características y comportamientos que están definidas de la clase de donde se instancian, sin embargo, aunque varios objetos provengan de una clase pueden tener identidad propia; en otras palabras: La identidad es el valor o estado de la propiedad que permite a un objeto diferenciarse de otros. En programación orientada a objetos cada Auto se conocerá como una instancia de la clase Auto.

Clase	Automóvil
Objeto	Auto
Estado	No. Puertas, maleta, tipos de faros, año, color.
Comportamiento	Acelerar, frenar, girar.

Pilares de la Programación Orientada a Objetos

Es importante entender los cuatro pilares de la programación orientada a objetos: abstracción, encapsulación, herencia y polimorfismo.

Abstracción:

Como se explicó anteriormente, los objetos tienen atributos o características que representan los datos asociados al mismo, estos atributos y sus valores en un momento dado, determinan el estado de un objeto. De igual forma los objetos tienen funcionalidades o comportamientos llamados métodos en la programación orientada a objetos. Con estos métodos accedemos a los atributos de una manera predefinida y se implementan el comportamiento del objeto.

Cuando desarrollemos un software, crearemos muchos objetos, que en algún momento vamos a requerir para resolver una situación planteada. Es aquí donde entra el concepto de abstracción. Con la abstracción podremos tomar lo que hace falta de un objeto del mundo real para el sistema en un momento dado, es captar las características esenciales de un objeto, así como su comportamiento.

Veamos como lo aplicamos a un ejemplo. En un taller mecánico se desea registrar los automóviles que ingresan al mismo. Es decir que vamos a tomar para registro de ingreso Marca, Modelo, Año, Cliente,

Rif, Dirección Fiscal, teléfono del cliente. En este Caso tomamos del objeto automóvil solo los datos que necesitamos Marca, Modelo y Año. Del Objeto Cliente, el cliente su RIF, dirección fiscal y teléfono.

Encapsulamiento
El objetivo es “meter todo en una capsula”, juntar las piezas que hacen que funcione como un todo. Ejemplo meto el motor dentro del auto.
Polimorfismo
Lograr que un objeto se comporte como si fuese una implementación de otra clase. Ejemplo: Un carro comportándose como una grúa.

Herencia
Tomar características y funcionalidades definidas en otras clases. Ejemplo: Auto hereda de vehículo motorizado. Como grúa también hereda de vehículo automotor.
Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes.