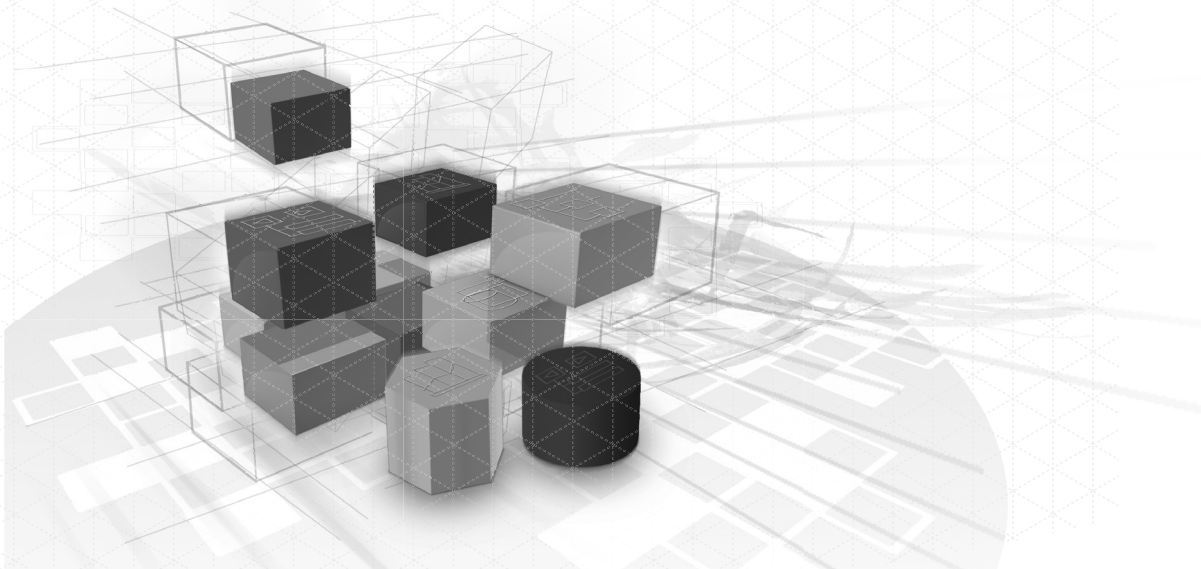
EMPRESA EDITORA MACRO®

PROGRAMACIÓN ORIENTADA A OBJETOS

Visual Basic 2012



PROGRAMACIÓN ORIENTADA A OBJETOS CON VISUAL BASIC 2012
Autor: Manuel Angel Torres Remon

Edición a cargo de:
Empresa Editora Macro E.I.R.L.
Av. Paseo de la República 5613 - Miraflores
Lima - Perú
(511) 719-9700
ventas@editorialmacro.com
http://www.editorialmacro.com

Primera edición: Octubre 2012 - 1000 ejemplares

Impreso en los Talleres Gráficos de
Empresa Editora Macro E.I.R.L.
Lima - Perú

ISBN Nº 978-612-304-085-7
Hecho el Depósito Legal en la Biblioteca Nacional del Perú Nº 2012-12377

MACRO®
EMPRESA EDITORA

MANUEL ANGEL TORRES REMON

Manuel Torres es un consultor dedicado a la docencia de cursos de tecnología como Visual NET y Microsoft SQL Server; ha brindado capacitación en las instituciones más importantes de Lima-Perú.

Recibió su formación tecnológica en el Instituto Superior Tecnológico Publico Manuel Arévalo Cáceres y también en la Universidad Alas Peruanas de la republica del Perú. En ambas instituciones recibió una buena formación profesional, demostrándolo en las diferentes instituciones que labora.

Actualmente se desempeña como consultor tecnológico de grandes empresas como TopyTop, Nestlé y como docente en instituciones educativas como el instituto Manuel Arévalo Cáceres, Cibertec y Unimaster de la Universidad Nacional de Ingeniería; en todas ellas impartiendo cursos de tecnología especialmente en Análisis, Programación y Base de Datos.

Ha publicado libros como Programación Transact con SQL Server 2012, Programación VBA con Excel y la Guía Practica de VBA con Excel.

Se le puede localizar al email manuel.torresr@hotmail.com, o a los teléfonos (511)-982360540 / (511)-996396023.

Agradecimientos

Cada libro desarrollado es una labor donde se sacrifica muchas veces tiempo con la familia, pero tiene sus recompensas cuando el material se vuelve una realidad. No cabe duda que me complace compartir con ustedes los casos expuestos ya que me facilitará en el desempeño de mis labores como docente, por eso este agradecimiento en primer lugar a la familia MACRO por confiar nuevamente en mi persona para el desarrollo del libro Programación Orientada a Objeto con Visual Net 2012.

En segundo lugar agradecer enormemente a las personas que me acompañan siempre como lo son mis pequeñas hijas Ángela Victoria y Fernanda Ximena Torres Lázaro y mi esposa Luz que con mucha paciencia me han dado el tiempo necesario para la elaboración de este material, lo considero un gran sacrificio que lo agradeceré por siempre. Gracias de nuevo.

Introducción

Visual Net 2012 es un lenguaje de programación orientado a objetos, bastante intuitivo para los que recién se inician en el mundo de la programación puesto que cuando empecé a desarrollar aplicaciones lo realizaba en Pascal o C++, pero cuando salió la versión Visual Basic 5 me resultó bastante cómodo entender su entorno y su desarrollo puesto que ya había pasado por los fuertes como Pascal o C++ que algunos compañeros de labores los llaman la base de programación.

Desde ese tiempo hasta hoy la evolución de Visual Net ha sido más beneficiosa para aquellos desarrolladores que se inician puesto que las herramientas, las propiedades y muchas herramientas que propone NET son intuitivitas, pero siempre hay algo que les recomiendo a mis estudiantes y es algo que usted también debe suponer que la lógica en la programación la pone usted no el lenguaje. Net podrá ser sencillo, pero tiene una arquitectura basada en un patrón complejo de comprender y más aún si tratamos de manejar clases, objeto y términos que sólo se veían en un análisis orientado a objetos.

Para aquellos programadores fanáticos de Visual Basic 6 que aún abundan les recomiendo que desarrollen sus aplicaciones en Visual Net 2012 ya que no es tan complejo su desarrollo; sin embargo, siempre hay que leer la sintaxis de la clase que se compone puesto que todas las aplicaciones en Net son clases.

El libro de Programación Orientada a Objetos con Visual Net 2012 consta de 8 capítulos. En el capítulo 1 se explica algunos conceptos básicos de clase y objeto ya que dentro del material se encuentran términos como instancias, atributos o métodos que resultarán un poco complejos de entender si no conocen conceptualmente de que se tratan, en el capítulo 2 se explican los espacios de nombres que proporciona Net para el manejo de Clases y base de datos que usaremos a lo largo del material y también se proponen casos desarrollados sobre clases y herencias que sugiero que los practiquen, puesto que el objetivo es comprender como crear una clase y usarla en una aplicación. En el capítulo 3 se expone el tema de colecciones que pretende hacer comprender al usuario que las variables pueden evolucionar a una colección y que puede ser controlada por datos que siempre manejo, pero esta vez podrá manipularlas al mismo tiempo empezando por los arreglos, pilas y colas. En el capítulo 4 se aplica la serializacion de archivos XML con el objetivo de poder implementar aplicaciones que permitan generar archivos XML a partir de datos. En el capítulo 5 es donde usted se debe concentrar, puesto que se explica todas las formas de poder consultar datos desde el motor de base de datos SQL Server; para este material use la versión 2012, si usted no cuenta con el software use cualquier versión. En el capítulo 6 se implementan aplicaciones que permitirán conectarse a una base por medio de las transacciones que Net las controla de manera efectiva descargando un poco el trabajo a las transacciones que ofrece SQL Server. En el capítulo 7 se implementan las aplicaciones haciendo uso del TableAdapter mediante el uso de la clase DataSet y a la vez se explica el manejo de la tecnología LinQ to SQL que se incorpora con fuerza en la programación. Y en el capítulo 8, el último, se exponen dos casos donde resume todo lo desarrollo de este material con una implementación a N-Capas que es el objeto principal del libro.

En los primeros capítulos se explica paso por paso las líneas de programación de cada caso desarrollado ya que este libro no trata de funciones básicas de Visual Basic sino más bien está dirigido a usuarios intermedio-avanzados.

Espero le sirva el material como me sirve a mí en el desarrollo de mis labores académicas.

A quién está dirigido este libro

Si usted se inicia en la programación recomiendo que implemente casos básicos donde se use estructuras como If, Select Case, For y While que le serán muy útil en este material.

El usuario de este material debe tener conocimiento básico sobre la creación y administración de los datos de una base de datos ya que a partir del capítulo 5 se exponen aplicaciones Cliente-Servidor que hacen uso de SQL Server. El tema más usado en este material es el procedimiento almacenado nos referimos a la implementación y ejecución.

Finalmente, el libro está dirigido a desarrolladores de Visual Basic .Net desde principiantes a profesionales que desean ampliar sus conocimientos con ayuda de .Net FrameWork 4.5. En este material aprenderá todas las técnicas necesarias para diseñar aplicaciones de calidad profesional.

Índice

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

CAP.

1

Introducción a la Programación Orientada a Objetos

CAPACIDAD:

Usted podrá entender conceptos básicos de la programación orientada a objetos visionando el desarrollo de aplicaciones .NET 2012.

1.1. INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

Hasta hace unos años se hablaba de la programación estructurada como técnica para escribir aplicaciones. La cual sólo contaba con tres estructuras que hasta hoy la implementamos: la estructura secuencial, estructura de selección (If) e iteración (For, While).

Hoy en día las aplicaciones informáticas en cualquier lenguaje de programación son mucho más ambiciosas que las necesidades de programación hacia algunos años, principalmente debido al boom de las aplicaciones visuales y mucho más si hablamos de la programación móvil, ello ha llevado a una evolución en el desarrollo de nuevas técnicas, tales como la programación orientada a objetos y el desarrollo de entornos de programación que facilitan la programación de grandes aplicaciones.

La programación orientada a objetos reúne un conjunto de técnicas para obtener calidad interna como medio para obtener calidad externa (Reutilización y Extensibilidad), se basa en dividir un programa en pequeñas unidades lógicas de código. A estas pequeñas unidades lógicas de código se les llama objetos.

El software se organiza como una colección de objetos que contienen tanto estructura como comportamiento. Por lo tanto, un programa orientado a objetos es una colección de clases. Necesitará una función principal que cree objetos y comience la ejecución mediante la invocación de sus funciones miembro.

Los lenguajes de programación orientados a objetos tratan a las aplicaciones como conjuntos de objetos que se ayudan entre sí para realizar acciones. Permitiendo que los programas sean más fáciles de escribir, mantener y reutilizar en el tiempo.

Los objetos en los lenguajes de programación siempre tomarán ambigüedad si no definen en concepto de la mejor manera, estos tienen información (atributos) que los diferencia de otros pertenecientes a otra clase. Por medio de métodos se comunican los objetos de una misma o diferente clase produciendo el cambio de estado de los objetos. Esto hace que a los objetos se les trate como unidades indivisibles en las que no se separan la información ni los métodos usados en su tratamiento.

Los lenguajes de programación orientados a objetos tienen su origen en un lenguaje que fue diseñado por los profesores Ole-Johan Dahl y Kristen Nygaard en Noruega. Este lenguaje de programación orientado a objetos fue el “Simula 67” que fue un lenguaje creado para hacer simulaciones de naves.

La programación orientada a objetos se fue convirtiendo en el estilo de programación dominante a mediados de los años ochenta, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las Interfaces gráficas de usuario (GUI), para las cuales la programación orientada a objetos está particularmente bien adaptada.

Las características de orientación a objetos fueron agregadas a muchos lenguajes existentes durante ese tiempo, incluyendo Ada, BASIC, Lisp, Pascal, entre otros. La adición de estas características a los lenguajes que no fueron diseñados inicialmente para ellas condujo a menudo a problemas de compatibilidad y en la capacidad de mantenimiento del código. Los lenguajes orientados a objetos "puros", por su parte, carecían de las características de las cuales muchos programadores habían venido a depender. Para saltar este obstáculo, se hicieron muchas tentativas para crear nuevos lenguajes basados en métodos orientados a objetos, pero permitiendo algunas características imperativas de maneras "seguras". El Eiffel de Bertrand Meyer fue un temprano y moderadamente acertado lenguaje con esos objetivos pero ahora ha sido esencialmente reemplazado por Java, en gran parte debido a la aparición de Internet, y a la implementación de la máquina virtual de Java en la mayoría de navegadores.

Un ejemplo de lenguaje de programación orientada a objetos completo es PHP en su versión 5, ya que ahora soporta una orientación completa a objetos, cumpliendo todas las características propias de la orientación a objetos.

Como lenguajes de programación que permiten manejar objetos tenemos:

- ActionScript
- Ada
- C++, C#
- Clipper
- Gambas
- Eiffel
- Fortran 90/95
- Java
- JavaScript
- PHP
- PowerBuilder
- Python
- Ruby
- Smalltalk
- VB.NET
- Visual FoxPro 6 - Superior
- Visual Basic 6.0

Muchos de estos lenguajes de programación no son puramente orientados a objetos, sino que son híbridos ya que combinan la POO con otros paradigmas.

1.2. Características de la POO

Existe un acuerdo acerca de qué características contempla la "orientación a objetos", las características siguientes son las más importantes:

- **Abstracción**: denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

- **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. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

- **Modularidad**: se denomina Modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la Modularidad de diversas formas.

- **Principio de ocultación**: cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

- **Polimorfismo**: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

- **Herencia**: 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. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

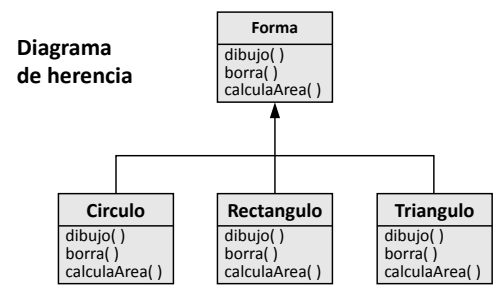


Fig. 1.1

- **Recolección de basura**: la recolección de basura o garbage collector es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto, desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse manualmente.

1.3. Conclusión

Cuando se ejecuta un programa orientada a objetos, ocurren tres acciones:

1. Se crean los objetos cuando se necesitan.
2. Los mensajes se envían desde uno objetos y se reciben en otros.
3. Se borran los objetos cuando ya no son necesarios y se recupera la memoria ocupada por ellos.

Entonces podemos concluir que la programación orientada a objetos estructura el programa en objetos que cooperan entre sí, cada objeto es instancia de alguna clase y las clases están relacionadas casi siempre mediante herencias.

1.4. Los Objetos

Siempre hay conceptos que no tienen mucho concepto explicativo como es el caso de los objetos, se ha preguntado ¿Qué es un objeto en nuestras vidas? Ya que desde que nace el hombre se involucra con ellos condicional e incondicionalmente. Un concepto que leí alguna vez es que para que un objeto sea catalogado como objeto debe poder sostenerlo y poder soltarlo. Pero en nuestro caso no se adapta mucho ya que si hablamos que todo lo lógico proporcionado por las computadoras estaríamos hablando que existen objetos abstractos que no podrán sostener y soltarlo pero cumplirá con los principios de los objetos.

Entonces un objeto es un concepto o abstracción utilizado para representar algún elemento del sistema, esta contiene características particulares llamado atributos y las funciones llamadas operaciones o métodos que operan sobre ellos.

Veamos un caso, en una tienda comercial el encargado de planilla necesita crear una aplicación en un lenguaje de programación orientada a objetos donde se implemente el pago de los vendedores para lo cual se necesita saber el nombre del vendedor y el número de ventas realizadas en un mes. De acuerdo a estos datos podrá obtener el sueldo mensual de dicho vendedor.

¿Cuántos objetos hay en el problema?, nombraremos los objetos conforme vamos leyendo el caso:

- Tienda Comercial
- Encargado de Planilla
- Pago
- Vendedor
- Venta

Como notará todo lo que se nombra dentro de un caso es un objeto, tiene que tener en cuenta que podría haber atributos o métodos involucrados. Veamos teóricamente todo objeto debe tener atributos y métodos, ahora listaremos los mismos objetos y colocaremos sus respectivos atributos y métodos posibles:

OBJETO	ATRIBUTOS	MÉTODOS
Tienda Comercial	Dirección Teléfono Ciudad País	Registrar nueva sucursal
Encargado de Planilla	Código Nombres Categoría Año Ingreso	Registrar nuevo encargado Actualizar datos del encargado
Pago	Número de Pago Fecha de Pago Monto de Pago	Registrar el pago Anular pago
Vendedor	Código Nombres Categoría Sueldo Mensual	Registrar nuevo vendedor Actualizar datos del vendedor Determinar Sueldo
Venta	Número de Venta Fecha de Venta Monto de Venta Número de Ventas	Registrar nueva venta Anular venta Calcular montos de venta

Vea usted que todos los objetos tienen atributos y métodos la diferencia entre ellos se debe a una fórmula sencilla de detección, los atributos no usan verbos mientras que los métodos son estrictamente verbos analice nuevamente la tabla y observará claramente a que sección le pertenece cada atributo o método encontrado en el caso.

1.5. IDENTIFICAR OBJETOS

La idea principal de la POO consiste en combinar datos y objetos en una sola unidad, esta unidad se llama OBJETO. Entonces como identificar un objeto; los Objetos generalmente se ubican en las siguientes categorías:

- **Cosas tangibles**

  Avión, auto, televisor, entre otros.

- **Roles o papeles que juega la gente o las cosas**

  Gerente, cliente, vendedor, entre otros.

- **Organizaciones o entidades**

  Empresa, departamento.

- **Incidentes o cosas que suceden**

  Vuelos, accidentes, llamada.

- **Interacciones entre dos o más objetos simples de detectar**

  Comprar, vender, boda.

- **Lugares**

  Muelle, carretera, etc.

1.6. Diagrama de Clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

El diagrama de clases incluye mucha más información como la relación entre un objeto y otro, la herencia de propiedades de otro objeto, conjuntos de operaciones/propiedades que son implementadas para una interfaz gráfica.

Se mostrará un ejemplo de diagrama de clase desde la perspectiva .NET 2012.

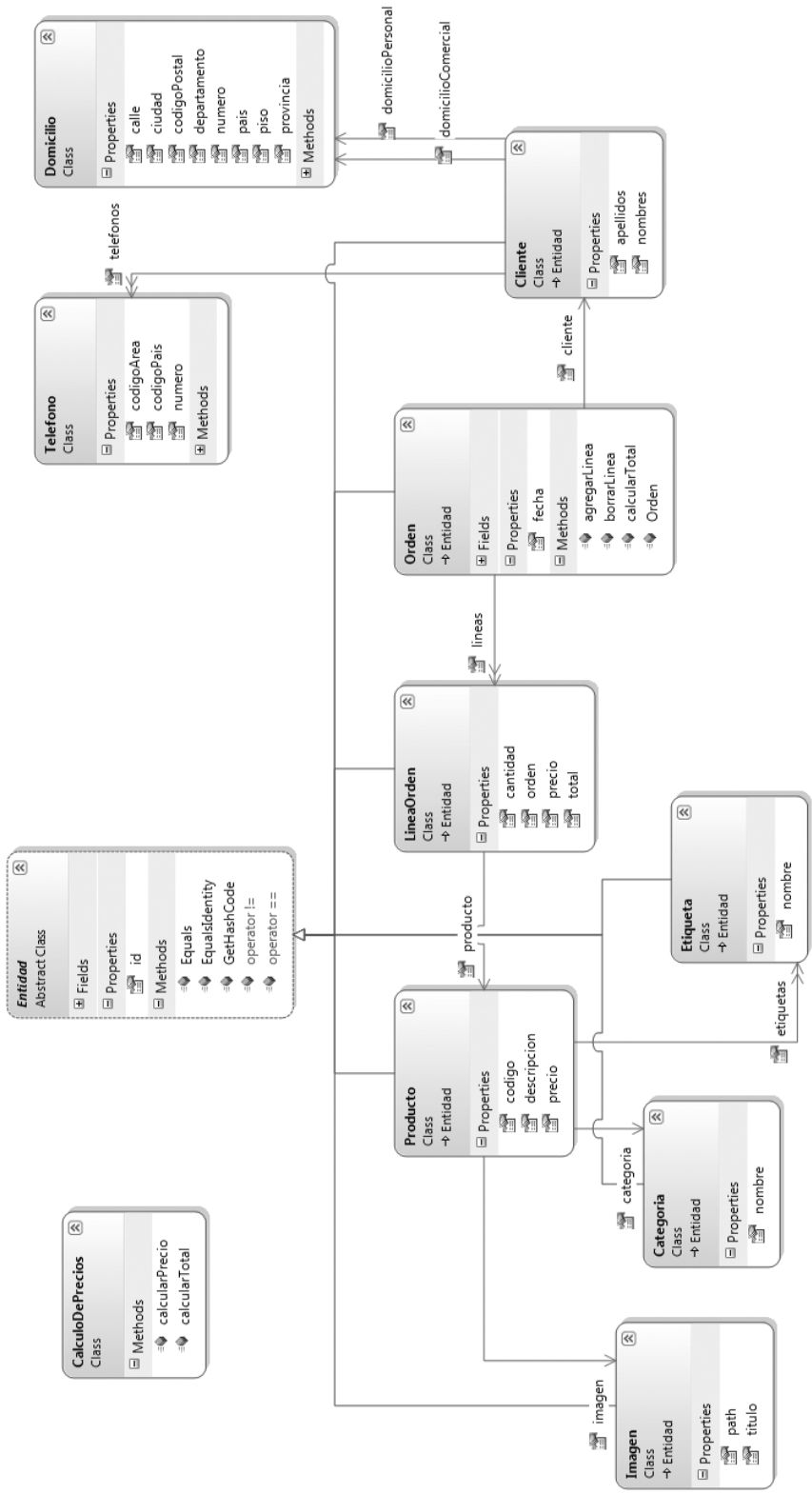

Domicilio
Class
Properties
calle
ciudad
codigoPostal
departamento
numero
pais
piso
provincia
Methods
domicilioPersonal
domicilioComercial
Cliente
Class
Entidad
Properties
apellidos
nombres
telefonos
Telefono
Class
Properties
codigoArea
codigoPais
numero
Methods
cliente
Orden
Class
Entidad
Fields
Properties
fecha
Methods
agregarLinea
borrarLinea
calcularTotal
Orden
lineas
LineaOrden
Class
Entidad
Properties
cantidad
orden
precio
total
Entidad
Abstract Class
Fields
Properties
id
Methods
Equals
EqualsIdentity
GetHashCode
operator !=
operator ==
producto
Etiqueta
Class
Entidad
Properties
nombre
etiquetas
Producto
Class
Entidad
Properties
codigo
descripcion
precio
categoria
Categoria
Class
Entidad
Properties
nombre
CalculoDePrecios
Class
Methods
calcularPrecio
calcularTotal
imagen
Imagen
Class
Entidad
Properties
path
titulo

Fig. 1.2

Impreso en los Talleres Gráficos de

Surquillo
☎ 719-9700