
3


Análisis
Orientado
a Objetos

3. Análisis orientado a objetos

Durante los últimos años ha ido creciendo de forma considerable el análisis orientado a objetos. Presentándose un interés creciente en el campo debido a sus enormes ventajas que se explican más adelante.

El Análisis Orientado a Objetos (AOO) va progresando como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis. En lugar de examinar un problema mediante el modelo clásico de entrada-proceso-salida (flujo de información) o mediante un modelo derivado exclusivamente de estructuras jerárquicas de información, el AOO introduce varios conceptos nuevos como son: las clases, una instancia, objetos, atributos y métodos (Figura 3.1).

- Una *clase.* Es una plantilla para objetos múltiples con características similares, comprenden todas esas características de un conjunto particular de objetos.

- Una *instancia* de una clase. Es otro término para un objeto real. Si la clase es la representación general de un objeto, una instancia es su representación concreta. A menudo se utiliza indistintamente la palabra objeto o instancia para referirse, precisamente, a un objeto.

- Un *objeto.* Es la instancia de una clase. Una clase es la representación abstracta de un concepto en el mundo real, y proporciona la base a partir de la cual creamos instancias de objetos específicos. Cada objeto es un elemento único de la clase en la que se basa. Si una clase es como un molde, entonces un objeto es lo que se crea a partir del molde. La clase es la definición de un elemento; el objeto es el elemento. El molde para una figura de cerámica en particular, es como una clase; la figura es el objeto.

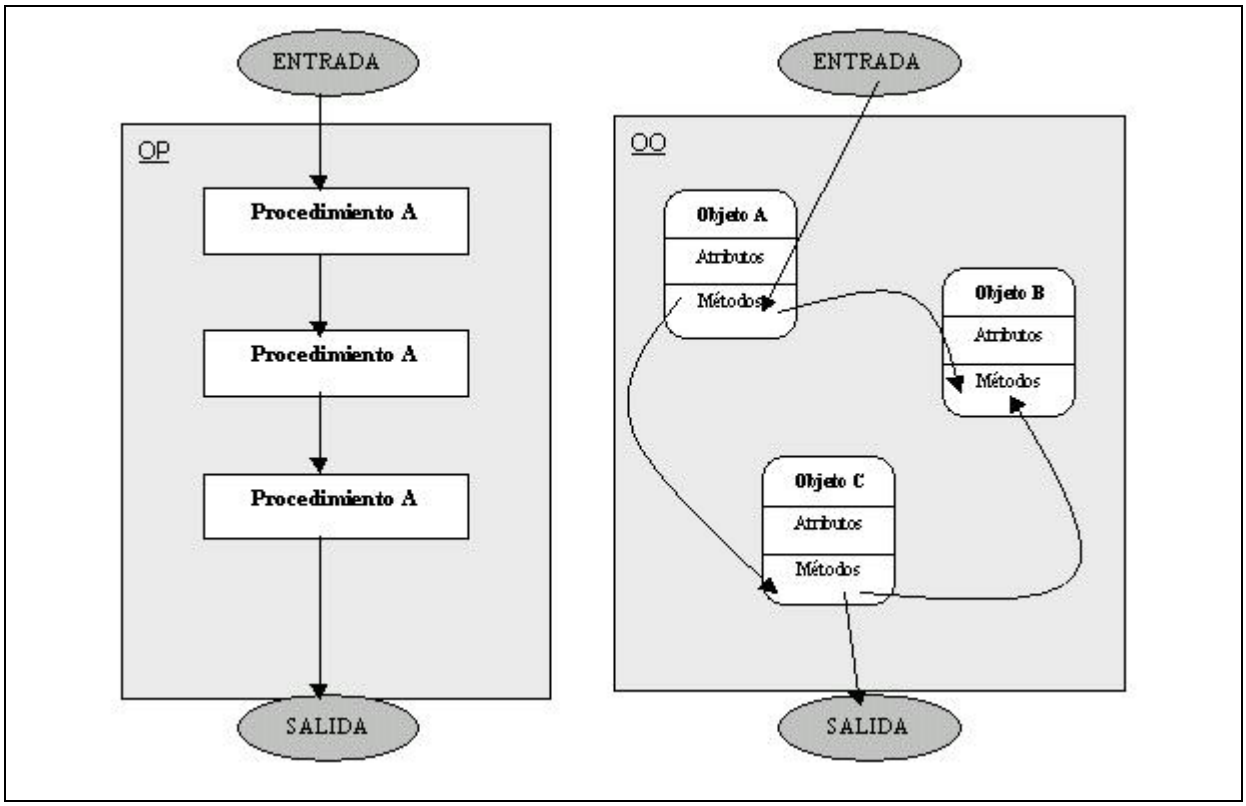

Figura 3.1 Diferencia del flujo de información en sistemas basados en modelos clásico (OP) y orientado a objetos (OO)

En los lenguajes orientados a objetos, cada clase está compuesta de dos cualidades: *atributos* (estado) y *métodos* (comportamiento o conducta). Los atributos son las características individuales que diferencian a un objeto de otro (ambos de la misma clase) y determinan la apariencia, estado u otras cualidades de ese objeto. Los atributos de un objeto incluyen información sobre su estado.

El enfoque particular del análisis orientado a objetos, modela la forma en que las personas comprenden y procesan la realidad a través de los conceptos que adquieren. Mismo que se pueden implantar por diversos medios, como máquinas, computadoras y personas. Así, la implantación puede incluir válvulas, tuberías, dispositivos de medición, circuitos, compuertas, controladores, flujo de materiales, reglas de dirección, entre otros.

La orientación a objetos puede describirse como el conjunto de disciplinas que desarrollan y modelan software que facilita la construcción de sistemas complejos a partir de componentes.

El atractivo intuitivo de la orientación a objetos es por que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible, Estos conceptos y herramientas orientados a objetos son tecnologías que permiten que los problemas del mundo real sean expresados de modo fácil y natural.

Las técnicas orientadas a objetos proporcionan mejoras y metodologías para construir sistemas de software complejos a partir de unidades de software modularizadas y reutilizables. Se necesita un nuevo enfoque para construir software en la actualidad, capaz de manipular tanto sistemas grandes como pequeños y debe crear sistemas fiables que sean flexibles, mantenibles y capaces de evolucionar para cumplir las necesidades del cambio.

La orientación a objetos trata de cubrir las necesidades de los usuarios finales, así como las propias de los desarrolladores de productos de software. Estas tareas se realizan mediante el modelado del mundo real. El soporte fundamental es el modelo objeto.

El AOO, permite al ingeniero del software modelar un problema a través de la representación de objetos, atributos y operaciones como las componentes primarias del modelado.

El propósito del AOO es definir todas las clases, relaciones y comportamientos asociadas con ellas, que son relevantes al problema que se va a resolver. Para cumplirlo se debe ejecutar las siguientes tareas:

- Identificar las clases.
- Definir la jerarquía de clases.
- Representar las relaciones de objeto a objeto.
- Modelar el comportamiento del objeto.
- Repetir iterativamente las tareas anteriores hasta completar el modelo.

Una amplia variedad de métodos de análisis orientado a objeto han sido propuestos, pero todos poseen un conjunto de características comunes:

- Representación de clases o jerarquía de clases.

- Creación de modelos de objeto-relación.
- Derivación de modelos objeto-comportamiento.

3.1 Historia del análisis orientado a objetos

La primera vez que se propuso un enfoque orientado a objetos para el desarrollo de software fue a finales de los 60´s. Sin embargo, las tecnologías de objetos han necesitado casi veinte años para llegar a ser ampliamente usadas. Durante los 90´s, la ingeniería del software orientada a objetos se convirtió en el paradigma de elección para muchos productores de software y para un creciente número de sistemas de información y profesionales de la ingeniería.

Vivimos en un mundo de objetos. Estos objetos existen en la naturaleza, en entidades hechas por el hombre, en los negocios y en los productos que usamos. Pueden ser clasificados, descritos, organizados, combinados, manipulados y creados. Por eso no es sorprendente que se proponga una visión orientada a objetos para la creación de software de computadora, una abstracción que modela el mundo de forma tal que nos ayuda a entenderlo y gobernarlo mejor.

Las tecnologías de objetos llevan a una reutilización de componente de software que da como resultado un desarrollo de software más rápido y a programas de mejor calidad. El software orientado a objetos es más fácil de mantener debido a que su estructura es inherentemente poco acoplada, esto lleva a menores efectos colaterales cuando se deben hacer cambios. Los sistemas orientados a objetos son más fáciles de adaptar y más fácilmente escalables (pueden crearse grandes sistemas ensamblando subsistemas reutilizables).

Hacia mediados de los 80´s, los beneficios de la programación orientada a objetos empezaron a obtener reconocimiento, y el diseño de objetos pareció ser un enfoque sensato para la gente que deseaba utilizar el lenguaje de programación orientada a objetos. Un enfoque orientado a objetos para programar ofrece muchos beneficios sobre un enfoque estructurado.

El análisis orientado a objetos y su diseño se enfocan en los objetos. Los objetos tienen ciertos comportamientos y atributos que determinan la manera en que

interactúan y funcionan. No se intenta proporcionar un orden para las acciones al momento del diseño debido a que los objetos funcionan basados en la manera en que funcionan otros objetos. La programación orientada a objetos ayuda a los desarrolladores a crear objetos que reflejan escenarios del mundo real.

Las implementaciones orientadas a objetos ocultan datos, lo cual significa que muestran únicamente los comportamientos a los usuarios y ocultan el código subyacente de un objeto. Los comportamientos que el programador expone son únicamente aquellos elementos que el usuario de un objeto puede afectar.

El enfoque orientado a objetos permite que los objetos estén autocontenidos. Los objetos existen por sí mismos, con una funcionalidad para invocar comportamientos de otros objetos. Al utilizar un enfoque orientado a objetos, los desarrolladores pueden crear aplicaciones que reflejan objetos del mundo real, como rectángulos, elipses y triángulos, además de dinero, números de partes y elementos en un inventario.

En un enfoque orientado a objetos, los objetos, por definición, son modulares en su construcción. Esto quiere decir que son entidades completas y, por lo tanto, tienden a ser altamente reutilizables. Las aplicaciones orientadas a objetos se construyen sobre el paradigma de los mensajes o de los eventos en donde los objetos envían mensajes a otros objetos, como el sistema operativo Microsoft Windows.

La programación orientada a objetos representa para los años 90´s lo que la programación estructurada fue para los 70´s: un nuevo e importante paradigma para mejorar la construcción, mantenimiento y utilización de software. Los métodos tradicionales de programación tienden a ver los programas como un conjunto de procedimientos que se llaman unos a otros. Cada procedimiento tiene asociados unos datos pasivos sobre los que opera. La programación orientada a objetos cambia esta visión por otra en la que una aplicación está compuesta por objetos con estado propio dotados de funcionalidad. Los objetos se comunican entre sí y tienen cada uno una forma propia de respuesta, que viene determinada por una serie de procedimientos que son asociados a cada objeto.

El objetivo de esta tecnología es obtener un software más consistente, robusto y reutilizable, más fácil de verificar, mantener, refinar y extender. El paradigma orientado a objetos representa un paso más en la dirección de acercar el lenguaje de las soluciones informáticas al lenguaje en que se plantean los problemas.

La programación orientada a objetos promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

3.2 Programación orientada a objetos (POO)

Proporciona conceptos y herramientas con las cuales modela y representa el mundo real, para ser expresados de modo fácil y natural.

Paradigma de la programación orientada a objetos

El Paradigma de programación representa un enfoque particular o filosofía para el desarrollo de software. Unos no son mejores que otros sino que hay situaciones donde un paradigma resulta más apropiado para la resolución del problema.

En la actualidad podemos dividir los paradigmas de programación con base en la forma y tipo de programación:

- Orientados a procedimientos: Algoritmos.
- Orientados a objetos: Clases y Objetos.
- Orientados a lógica*:* Expresado en cálculo de predicados.
- Orientados a reglas: Reglas if-then*.*

Existen cuatro conceptos importantes dentro de la programación orientada a objetos:

- Objetos
- Clases
- Herencia
- Envío de mensajes

Objetos. La Programación orienta a objetos (POO) está modelada a partir de objetos del mundo real. Un objeto del mundo real es cualquier cosa que vemos a nuestro alrededor, como por ejemplo una computadora, un árbol o un automóvil. Todos los objetos del mundo real tienen dos componentes esenciales:

- Características.
- Comportamiento.

Por ejemplo, los automóviles tienen características como marca, modelo, color, etc. y comportamiento: frenar, acelerar, retroceder, llenar combustible, cambiar llantas, etc.
Los Objetos de Software, al igual que los objetos del mundo real, también tienen características y comportamientos. Un objeto de software mantiene sus características en una o más variables, e implementa su comportamiento con métodos (función o subrutina asociada a un objeto).

Cuando a las características del objeto le ponemos valores decimos que el objeto tiene estados. Las variables almacenan los estados de un objeto en un determinado momento.

Un objeto de software lo podemos definir como una unidad de código compuesto de variables y métodos relacionados.

Clases. La clase es un modelo o prototipo que define las variables y métodos comunes a todos los objetos de cierta clase. También se puede decir que una clase es una plantilla genérica para un conjunto de objetos de similares características (Figura 3.2).

Es un tipo definido por el usuario que determina las estructuras de datos y las operaciones asociadas con ese tipo. Cada vez que se construye un objeto de una clase, se crea una instancia de esa clase. Una clase es una colección de objetos similares y un objeto es una instancia de una definición de una clase.

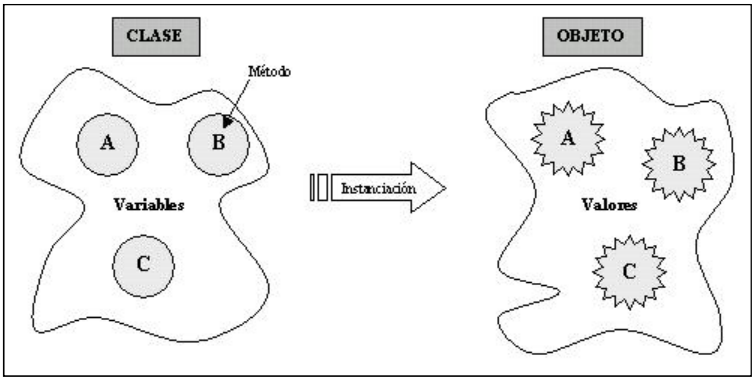

Figura 3.2 Representación de clases y objetos (instancias de clases)

Mensajes. Los mensajes son el medio a través del cual interactúan los objetos. Un mensaje estimula la ocurrencia de cierto comportamiento en el objeto receptor. El comportamiento se realiza cuando se ejecuta un método (Figura 3.3).

Estructuralmente un mensaje consta de tres partes:

- Objeto receptor o destino.
- Método miembro del receptor cuya ejecución se ha solicitado.
- Cualquier otra información adicional que el receptor pueda necesitar para ejecutar el método requerido.

Una operación dentro de un objeto emisor genera un mensaje de la forma:

destino.operación (parámetros)

Donde *destino* define al objeto receptor el cual es estimulado por el mensaje, *operación* se refiere al método que recibe el mensaje, y *parámetros* proporciona información requerida para el éxito de la operación.

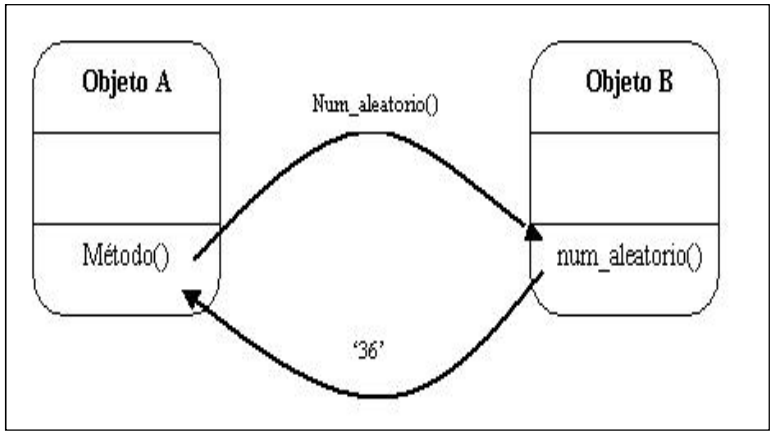

Figura 3.3. Comunicación entre objetos

Herencia. La herencia es la propiedad que permite a los objetos construirse a partir de otros objetos. Una clase se puede dividir en subclases, la clase original se denomina clase base; las clases que se definen a partir de la clase base, comparten sus características y añaden otras nuevas, a estas se les denominan clases derivadas.

Las clases derivadas pueden heredar métodos y datos de su clase base añadiendo su propio comportamiento y datos a la misma.

La herencia puede ser de dos tipos, simple (si sólo es posible heredar características de una sola clase (Figura 3.4)) o múltiple (si se pueden heredar características de varias clases).

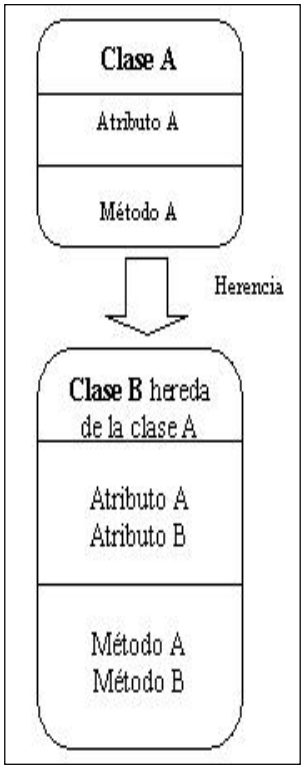

Figura 3.4. Herencia simple

Principios de la Programación Orientada a Objetos

Existen una serie de principios fundamentales para comprender cómo se realiza el modelado de la realidad al crear un programa bajo el paradigma de la orientación a objetos. Estos principios o propiedades son:

- Abstracción.
- Encapsulamiento.
- Modularidad.
- Jerarquía.
- Polimorfismo.

Abstracción. Es la propiedad que permite representar las características esenciales de un objeto, sin preocuparse de las restantes características (no esenciales).

Una abstracción se centra en la vista externa de un objeto, de modo que sirva para separar el comportamiento esencial de un objeto de su implementación. Definir una abstracción significa describir una entidad del mundo real, no importando la

complejidad que pueda tener y, a continuación, utilizar esta descripción en un programa.

El elemento clave de la POO es la clase. Una clase se puede definir como una descripción abstracta de un grupo de objetos, cada uno de los cuales se diferencia por su estado específico y por la posibilidad de realizar una serie de operaciones. Por ejemplo, una pluma estilográfica es un objeto que tiene un estado (llena de tinta o vacía) y sobre la cual se pueden realizar algunas operaciones (escribir, poner o quitar la tapa, llenar de tinta si está vacía, etc.).

Las mejores abstracciones de software hacen que las cosas complejas sean simples, logrando ocultar por completo los aspectos no esenciales de una clase. Estos aspectos no se pueden ver, ni usar, ni depender de ellos. Este principio es llamado dependencia mínima y es lo que hace que la abstracción sea tan importante.

Encapsulamiento. Es la propiedad que permite asegurar que el contenido de la información de un objeto está oculta al mundo exterior, es decir un objeto A no conoce lo que hace un objeto B, y viceversa. La encapsulación también es conocida como ocultación de la información (ocultamiento).

La encapsulación permite controlar la forma en que se utilizan los datos y los métodos. Puede utilizar modificadores de acceso para evitar que los métodos externos ejecuten métodos de clase o lean y modifiquen sus atributos. Para permitir que otros objetos consulten o modifiquen los atributos de los objetos, las clases suelen presentar métodos de acceso.

Con el encapsulado de los datos se consigue que las personas que utilicen un objeto sólo tengan que comprender su interfaz, olvidándose de cómo está implementada, y reduciendo la complejidad de utilización.

Modularidad. Es la propiedad que permite dividir una aplicación en partes más pequeñas (módulos), cada una de las cuales debe ser tan independiente de la aplicación en sí y de las partes restantes.

La dependencia a veces se conoce como acoplamiento. Un sistema con muchas dependencias tiene fuerte acoplamiento. Los buenos sistemas tienen débil acoplamiento, porque en ese caso los cambios en una parte son menos probables de propagarse.

Los módulos correctos a menudo tienen la propiedad de que sus interfaces proporcionan una abstracción de algún elemento conocido de manera intuitiva. Este tipo de módulos se dice que tienen una fuerte cohesión.

Si un módulo, de cualquier tamaño y complejidad es una buena abstracción (tiene fuerte cohesión y débil acoplamiento), puede ser factible reutilizarlo en sistemas posteriores, o sustituirlo en el sistema existente.

Jerarquía. Es una propiedad que permite ordenar las abstracciones. Las dos jerarquías más importantes de un sistema complejo son:

- Estructura de clases (jerarquía “es-un” (is-a): generalización/especialización)
- Estructura de objetos (jerarquía “parte-de” (part-of): agregación).

a) *Las jerarquías de generalización/especialización.* Se conocen como herencia. La herencia define una relación entre clases, en donde una clase comparte la estructura o comportamiento definido en una o más clases.

b) *Las jerarquías de agregación*. Es el concepto que permite el agrupamiento físico de estructuras relacionadas lógicamente, por ejemplo un camión se compone de ruedas, motor, sistema de transmisión y chasis; en consecuencia, camión es una agregación, y ruedas, motor, transmisión y chasis son agregados de camión.

Polimorfismo. Esta propiedad permite que un objeto presente diferentes comportamientos en función del contexto en que se encuentre. Por ejemplo un método puede presentar diferentes implementaciones en función de los argumentos que recibe, recibir diferentes números de parámetros para realizar una misma operación, y realizar diferentes acciones dependiendo del nivel de abstracción en que sea llamado.

El polimorfismo adquiere su máxima expresión en la derivación *o* extensión de clases, es decir, cuando se obtiene una clase a partir de una clase ya existente, mediante la propiedad de herencia.

3.2.1 Lenguajes orientados a objetos

Los lenguajes de POO tratan a los programas como conjuntos de objetos que se ayudan entre ellos para realizar acciones, permitiendo que los programas sean más fáciles de escribir, mantener y reutilizar.

El primer lenguaje de programación orientado a objetos fue el *Simula 67*, creado por los profesores Ole-Johan Dahl y Kristen Nygaard en Noruega. Fue un lenguaje creado para hacer simulaciones de naves.

Entre los principales lenguajes de este tipo tenemos:

C++. Es un lenguaje versátil, potente y general, mantiene las riquezas de C en cuanto a operadores, expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original. La evolución de C++ ha continuado con la aparición de Java, un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet.

C#. Es un lenguaje de programación simple pero eficaz, diseñado para escribir aplicaciones empresariales. El lenguaje C# es una evolución de los lenguajes C y

C++. Utiliza muchas de las características de C++ en las áreas de instrucciones, expresiones y operadores.

C# presenta considerables mejoras e innovaciones en áreas como seguridad de tipos, control de versiones, eventos y recolección de elementos no utilizados (liberación de memoria).

VB .NET. Visual Basic NET es un lenguaje de programación desarrollado por Microsoft muy apropiado para construir sistemas de información basados en red o mejor aun en Internet.

Se caracteriza por ser un lenguaje de programación fácil de leer y de escribir. Con la herencia visual, simplifica la creación de aplicaciones basadas en Windows,

FoxPro. Es un lenguaje de programación orientado a objetos y procedural, un Sistema Gestor de Bases de datos o Database Management System (DBMS), y desde la versión 7.0, un sistema administrador de bases de datos relacionales, producido por Microsoft.

FoxPro ofrece a los desarrolladores un conjunto de herramientas para crear aplicaciones de bases de datos para el escritorio, entornos cliente/servidor, tablet PC o para la Web.

Delphi. Es sin lugar a dudas el mejor entorno de desarrollo rápido de aplicaciones (RAD), con un potentísimo lenguaje el Object Pascal, un compilador rapidísimo que nos permite crear ejecutables con una velocidad cercana al C++, y con múltiples posibilidades: bases de datos, multimedia, web, etc.

PHP. Es un lenguaje de scripting que permite la generación dinámica de contenidos en un servidor web. El significado de sus siglas es HyperText Preprocessor. Entre sus principales características cabe destacar su potencia, su alto rendimiento, su facilidad de aprendizaje y su escasez de consumo de recursos.

Clarion. Es un Lenguaje de programación 4GL (Lenguaje de cuarta generación), además de ser un entorno integrado de desarrollo de Softvelocity orientado a la programación de aplicaciones de bases de datos. Es compatible con una gran

cantidad de bases de datos incluyendo todas las de formato SQL, ADO, y XML, además puede generar salidas a HTML, XML, archivos de texto y PDF, entre otros.

Python. Es un lenguaje de programación multiparadigma que permite varios estilos: Programación orientada a objetos, programación estructurada, programación funcional y programación orientada a aspectos. Una característica importante de Python es la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa.

Ruby. Es un lenguaje de guiones (scripts) para una programación orientada a objetos rápida y sencilla. Ruby es un lenguaje de programación interpretado su implementación oficial es distribuida bajo una licencia de software libre.

SmallTalk. Es un lenguaje orientado a objetos puro, pues todas las entidades que maneja son objetos. es mucho más que un lenguaje de programación, es un ambiente completo de desarrollo de programas. Integra de una manera consistente características tales como un editor, un compilador, un debugger, utilitarios de impresión, un sistema de ventanas y un manejador de código fuente.

Java. Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90´s. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.
El lenguaje toma en su mayoría la sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros.
Está compuesto de tres plataformas importantes:

- J2SE (StandarEdition) Plataforma diseñada para aplicaciones stand-alone (de escritorio) y applets.
- J2EE (EnterpriceEdition) Plataforma diseñada para aplicaciones web (jsp, servlets, ejb, etc).
- J2ME (MicroEdition) Plataforma diseñada para aplicaciones moviles (midlets).