**UNIVERSIDAD TECNOLÓGICA DEL SUROESTE DE GUANAJUATO**

**Nombre y firma del Alumno:**

**David Mendoza Aguirre**

**Esmeralda Morales Conejo**

**Materia:**

**“Programación de Aplicaciones”**

**Carrera:**

**Tecnologías de la Información y Comunicación**

**Nombre del profesor(a):**

**MI. Ariana Gómez Contreras**

**Fecha:**

**Viernes 23 de agosto de 2019**

**Orientación a Objetos**

La programación estructurada tradicional se basa fundamentalmente en la ecuación de Wirth:

Algoritmos + Estructuras de Datos = Programas

Esta ecuación significa que en la programación estructurada u orientada a procedimientos los datos y el código se trata por separado y lo único se realiza son funciones o procedimientos que tratan esos datos y los van pasando de unos a otros hasta que se obtiene el resultado que se desea.

La Programación Orientada a Objetos, POO (OOP, Object Oriented Programming, en inglés), es una técnica de programación cuyo soporte fundamental es el objeto. Un objeto es una extensión de un Tipo Abstracto de Datos (TAD), concepto ampliamente utilizado desde la década de los setenta. Un TAD es un tipo definido por el usuario, que encapsula un conjunto de datos y las operaciones sobre estos datos.

A la hora de definir TAD’s (u objetos) se usa un concepto que nos ayuda a representar la realidad mediante modelos computacionales, la abstracción, que es un proceso mental por el que se evitan los detalles para centrarse en las cosas más genéricas de manera que se facilite su comprensión. De hecho la abstracción no sólo se utiliza en la informática, un arquitecto al que le han encargado realizar los planos de un edificio no comenzará por diseñar los planos con máximo nivel de detalle, sino que comenzará a realzar ciertos esbozos en un papel para posteriormente ir refinando. Por supuesto que cuando está realizando los esbozos no se preocupa de por dónde van a ir las líneas eléctricas ni las tuberías de saneamiento, abstrae esos detalles para atacarlos posteriormente cuando tenga clara la estructura del edificio.

La diferencia entre el concepto de TAD y el de objeto radica en que además del proceso de abstracción que se utiliza para su definición, existen otros dos con los que se forma el núcleo principal de la programación orientada a objetos, estos son la herencia y el polimorfismo.

**Ventajas de la orientación a objetos**

Las ventajas más importantes de la programación orientada a objetos son las siguientes:

* **Mantenibilidad.** (facilidad de mantenimiento). Los programas que se diseñan utilizando el concepto de orientación a objetos son más fáciles de leer y comprender y el control de la complejidad del programa se consigue gracias a la ocultación de la información que permite dejar visibles sólo los detalles más relevantes.
* **Modificabilidad.** (facilidad para modificar los programas). Se pueden realizar añadidos o supresiones a programas simplemente añadiendo, suprimiendo o modificando objetos.
* **Reusabilidad**. Los objetos, si han sido correctamente diseñados, se pueden usar numerosas veces y en distintos proyectos.
* **Fiabilidad.** Los programas orientados a objetos suelen ser más fiables ya que se basan en el uso de objetos ya definidos que están ampliamente testados.

Estas ventajas son directas a los programadores. Estos, se podría decir, que son los ejecutores de un determinado proyecto software. Pero la orientación a objetos no sólo reporta beneficios a los programadores. En las etapas de análisis, previas a la codificación, el utilizar un modelado orientado a objetos reporta grandes beneficios ya estas mismas ventajas son aplicables a todas las fases del ciclo de vida de un proyecto software.

La tendencia actual es a tratar temas conceptuales de primer plano (o sea, en las fases de análisis) y no temas finales de implementación. Los fallos durante la etapa de implementación son más difíciles de corregir y más costosos que si se dan en las etapas previas.

El modelado orientado a objetos tiende al refinamiento sucesivo de manera que se llega a la etapa de implementación con un diseño lo suficientemente explícito para que no existan casos inesperados y todo independientemente del lenguaje de programación (salvo en etapas muy próximas a la implementación donde no hay más remedio que contar con el soporte que se recibe del lenguaje elegido). El desarrollo orientado a objetos es más una manera de pensar y no una técnica de programación.

**Conceptos básicos de la orientación a objeto**

Como ya hemos dicho la orientación a objetos se basa en conceptos como clase, objeto, herencia y polimorfismo, pero también en otros muchos, los cuales se explican a detalle a continuación.

**Abstracción**

La abstracción se enfoca en la vista externa del objeto y sirve para separar el comportamiento esencial del objeto de su implementación. La abstracción se enfoca sobre las características importantes de algún objeto, relativo a la perspectiva de que observa.

Gracias a la abstracción podemos representar las características esenciales de un objeto sin preocuparnos de las restantes características (no esenciales). La 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.

En el sentido más general, una abstracción es una representación concisa de una idea o de un objeto complicado. En un sentido más específico, la abstracción localiza y oculta los detalles de un modelo o diseño para generar y manipular objetos.

Una abstracción tiene un significado más general que la encapsulación, pudiendo hablar de abstracción de datos en lugar de encapsulación de datos.

La abstracción es una estrategia de resolución de problemas en la cual el programador se concentra en resolver una parte del problema ignorando completamente los detalles sobre cómo se resuelven el resto de las partes.

En este proceso de abstracción se considera que el resto de las partes ya han sido resueltas y por lo tanto pueden servir de apoyo para resolver la parte que recibe la atención.

Imaginemos que se quiere aplicar la abstracción a las Computadoras, las personas que compran una, no requieren saber de su funcionamiento interno para poder utilizarla.

Si ahora, la abstracción la aplicamos a los Automóviles, sucede algo muy similar, la persona que adquiere uno, no requiere saber cómo funciona de manera interna, solo tiene que saber manejar y los servicios básicos (gasolina, llevarlo a mantenimiento, agua, etc.).

**Clase y objeto**

Los objetos son las entidades básicas del modelo de objeto. La palabra objeto proviene del latín objectus, donde ob significa hacía, y jacere significa arrojar; o sea que teóricamente un objeto es cualquier cosa que se pueda arrojar.

Ejemplo: Una pelota o un libro se pueden arrojar, por lo tanto estos son objetos. Por otro lado, un avión o un elefante también se consideran objetos, aunque sean bastante pesados para ser arrojados.

Los objetos son más que simples cosas que se puedan arrojar, son conceptos pudiendo ser abstractos o concretos.

Ejemplo: Una mesa es un objeto concreto, mientras que un viaje es un objeto abstracto.

Los objetos corresponden por lo general a sustantivos, pero no a gerundios.

Ejemplo: Mesa y viaje son ambos sustantivos y por lo tanto objetos. Trabajando y estudiando son gerundios por lo cual no se consideran objetos.

Cualquier cosa que incorpore una estructura y un comportamiento se le puede considerar como un objeto.

Ejemplo: Una pelota es sólida y redonda y se le puede arrojar o atrapar. Un libro es rectangular y sólido y se le puede abrir, cerrar, y leer.

Un objeto debe tener una identidad coherente, al que se le puede asignar un nombre razonable y conciso.

Ejemplo: Se consideran manzanas todas las frutas con un sabor, textura, y forma similar.

La existencia de un objeto depende del contexto del problema. Lo que puede ser un objeto apropiado en una aplicación puede no ser apropiado en otra, y al revés. Por lo general, existen muchos objetos en una aplicación, y parte del desafío es encontrarlos.

Ejemplo: La temperatura se puede considerar un objeto abstracto, teniendo propiedades tales como el valor de la temperatura y el tipo de la escala en que se mide (Celsius o Fahrenheit). Por otro lado, si hablamos de un termómetro, la temperatura pasa a ser una propiedad del termómetro.

Los objetos se definen según el contexto de la aplicación.

Ejemplo: Una persona llamada Juan Pérez se considera un objeto para una compañía, mientras que para un laboratorio el hígado de Juan Pérez es un objeto. Una universidad como la BUAP se considera un objeto, mientras que dentro de la BUAP los objetos serían las aulas, los estudiantes y los profesores.

Los objetos deben ser entidades que existen de forma independiente. Se debe distinguir entre los objetos, los cuales contienen características o propiedades, y las propias características.

Ejemplo: El color y la forma de una manzana no se consideran propiamente objetos, sino propiedades del objeto manzana. El nombre de una persona se considera una propiedad de la persona.

Un grupo de cosas puede ser un objeto si existe como una entidad independiente.

Ejemplo: Un automóvil se considera un objeto el cual consiste de varias partes, como el motor y la carrocería.

Los objetos deben tener nombres en singular, y no en plural.

Ejemplo: Un automóvil es un objeto, automóviles son simplemente muchos objetos y no un solo objeto.

Parte de una cosa puede considerarse un objeto.

Ejemplo: La rueda, la cual es parte del automóvil, se puede considerar un objeto. Por otro lado, el lado izquierdo del automóvil sería un mal objeto.

Los objetos deben tener nombren razonables y concisos para evitar la construcción de objetos que no tengan una identidad coherente.

Ejemplo: Datos o información no son nombres concisos de objetos. Por otro lado, un estudiante es un objeto, ya que contiene propiedades como el número de matrícula y nombre del estudiante, además incluye un comportamiento tal como ir a clases, presentar exámenes, y graduarse. Todos los estudiantes cuyos apellidos comiencen con "A" sería un mal objeto ya que el nombre del objeto no es conciso.

El objeto integra una estructura de datos (atributos) y un comportamiento (operaciones).

Una clase describe un grupo de objetos con estructura y comportamiento común. (Clase y tipo no son necesariamente equivalentes, tipo se define por las manipulaciones que se le puede dar a un objeto dentro de un lenguaje y clase involucra una estructura, pudiendo corresponder a una implementación particular de un tipo.)

Las estructuras o propiedades de la clase se conocen como atributos y el comportamiento como operaciones. Una clase define uno o más objetos, donde los objetos pertenecen a la clase, teniendo características comunes.

Ejemplo: Juan Pérez y María López se consideran miembros de la clase persona, donde todas las personas tienen una edad y un nombre. La BUAP y la UDLA pertenecen a la clase universidad, donde todas las universidades tienen una dirección y un grado máximo. Chrysler y Microsoft pertenecen a la clase compañía, donde todas las compañías tienen una dirección, un número de empleados, y una ganancia al año.

Una clase se considera un "molde" del cual se crean múltiples objetos.

Ejemplo: La clase es como un molde de una cerámica de la cual se pueden crear múltiples cerámicas, todas con exactamente las mismas características. Para modificar las cerámicas hay que primero construir un nuevo molde.

Al definir múltiples objetos en clases se logra una abstracción del problema. Se generaliza de los casos específicos definiciones comunes, como nombres de la clase, atributos, y operaciones.

Ejemplo: Los objetos impresora a láser, impresora de burbuja, e impresora de punto son todos objetos que pertenecen a la clase impresora.

Una clase como la hemos definido se conoce también como clase básica.

Todos los objetos son instancias de una clase y la clase de un objeto es una propiedad implícita del objeto. Cada objeto conoce su clase y la mayoría de los lenguajes de programación orientados al objeto pueden determinar la clase de un objeto en tiempo de ejecución.

La agrupación en clases de los objetos permite la abstracción de un problema. Las definiciones comunes, tales como nombres de clases y de atributos se almacenan una vez por cada clase. Las operaciones se escriben una vez para cada clase: reutilización de código.

Imaginemos que queremos aplicar la abstracción a las Aves:

El objeto serían el pájaro, y sus características por ejemplo, serían pico, alas, plumas y patas.

El comportamiento sería volar, parar, etc.

**Polimorfismo**

Hace referencia a la posibilidad de que dos métodos implementen distintas acciones, aun teniendo el mismo nombre, dependiendo del objeto que lo ejecuta o de los parámetros que recibe.

La palabra polimorfismo proviene del griego, y significa que posee varias formas diferentes. Este es uno de los conceptos esenciales de una programación orientada a objetos. Así como la herencia está relacionada con las clases y su jerarquía, el polimorfismo se relaciona con los métodos.

El polimorfismo permite referirse a objetos de clases diferentes mediante el mismo elemento. Imaginamos que disponemos de una clase profesor (Teacher) que también hereda de ser persona (Person) y que ambas disponen del método trabaja (work) y que ambos evidentemente lo hacen de una forma distinta.

En general, hay tres tipos de polimorfismo:

* Polimorfismo de sobrecarga.
* Polimorfismo paramétrico (también llamado polimorfismo de plantillas).
* Polimorfismo de inclusión (también llamado redefinición o subtipado).

El polimorfismo de sobrecarga ocurre cuando las funciones del mismo nombre existen, con funcionalidad similar, en clases que son completamente independientes una de otra (éstas no tienen que ser clases secundarias de la clase objeto). Por ejemplo, la clase complex, la clase image y la clase link pueden todas tener la función "display". Esto significa que no necesitamos preocuparnos sobre el tipo de objeto con el que estamos trabajando si todo lo que deseamos es verlo en la pantalla.

Por lo tanto, el polimorfismo de sobrecarga nos permite definir operadores cuyos comportamientos varían de acuerdo a los parámetros que se les aplican. Así es posible, por ejemplo, agregar el operador + y hacer que se comporte de manera distinta cuando está haciendo referencia a una operación entre dos números enteros (suma) o bien cuando se encuentra entre dos cadenas de caracteres (concatenación).

El polimorfismo paramétrico es la capacidad para definir varias funciones utilizando el mismo nombre, pero usando parámetros diferentes (nombre y/o tipo). El polimorfismo paramétrico selecciona automáticamente el método correcto a aplicar en función del tipo de datos pasados en el parámetro.

Por lo tanto, podemos por ejemplo, definir varios métodos homónimos de addition() efectuando una suma de valores.

* El método entero addition(entero, entero) devolvería la suma de dos números enteros.
* real addition(real, real) devolvería la suma de dos flotantes.
* caracter addition(caracter, caracter) daría por resultado la suma de dos caracteres definidos por el autor.

Una signature es el nombre y tipo (estático) que se da a los argumentos de una función. Por esto, una firma de método determina qué elemento se va a llamar.

La habilidad para redefinir un método en clases que se hereda de una clase base se llama especialización. Por lo tanto, se puede llamar un método de objeto sin tener que conocer su tipo intrínseco: esto es polimorfismo de subtipado. Permite no tomar en cuenta detalles de las clases especializadas de una familia de objetos, enmascarándolos con una interfaz común (siendo esta la clase básica).

Imagine un juego de ajedrez con los objetos rey, reina, alfil, caballo, torre y peón, cada uno heredando el objeto pieza.

El método movimiento podría, usando polimorfismo de subtipado, hacer el movimiento correspondiente de acuerdo a la clase objeto que se llama. Esto permite al programa realizar el movimiento de\_pieza sin tener que verse conectado con cada tipo de pieza en particular.

**Encapsulamiento**

La encapsulación es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados. Por lo tanto, la encapsulación garantiza la integridad de los datos que contiene un objeto.

El usuario de una clase en particular no necesita saber cómo están estructurados los datos dentro de ese objeto, es decir, un usuario no necesita conocer la implementación Al evitar que el usuario modifique los atributos directamente y forzándolo a utilizar funciones definidas para modificarlos (llamadas interfaces), se garantiza la integridad de los datos (por ejemplo, uno puede asegurarse de que el tipo de datos suministrados cumple con nuestras expectativas bien que los se encuentran dentro del periodo de tiempo esperado).

La encapsulación define los niveles de acceso para elementos de esa clase. Estos niveles de acceso definen los derechos de acceso para los datos, permitiéndonos el acceso a datos a través de un método de esa clase en particular, desde una clase heredada o incluso desde cualquier otra clase. Existen tres niveles de acceso:

* **público:** funciones de toda clase pueden acceder a los datos o métodos de una clase que se define con el nivel de acceso público. Este es el nivel de protección de datos más bajo.
* **protegido:** el acceso a los datos está restringido a las funciones de clases heredadas, es decir, las funciones miembro de esa clase y todas las subclases.
* **privado:** el acceso a los datos está restringido a los métodos de esa clase en particular. Este es nivel más alto de protección de datos.

Hacer las variables que son innecesarias para el tratamiento del objeto pero necesarias para su funcionamiento privadas, así como las funciones que no necesitan interacción del usuario o que solo pueden ser llamadas por otras funciones dentro del objeto (Como por ejemplo, palpitar)

La encapsulación es muy conveniente y nos permite colocar en funcionamiento nuestro objeto en cualquier tipo de sistema, de una manera modular y escalable (algunas de las reglas de la ingeniería del software).

La técnica del encapsulamiento consiste en reunir todos los elementos que pertenecen a una misma entidad y otorgarles un determinado nivel de aislamiento según convenga. El aislamiento protege a los datos asociados a un objeto contra su modificación por actores que no tengan derecho sobre ellos. Gracias a esto el usuario de la clase solo se ha de concentrar en el uso de la misma y no en como realmente funcionan sus métodos. Una clase representa la encapsulación de una abstracción. Esto, nos viene a decir que cada clase ha de tener dos partes, una Interfaz y una implementación. La interfaz sería una definición de los elementos de la clase y la implementación corresponde al funcionamiento de la misma.

**Operadores**

Un operador es un elemento de código que realiza una operación en uno o más elementos de código que contienen valores. Los elementos de valor incluyen variables, constantes, literales, propiedades, valores devueltos de procedimientos Function y Operator y expresiones.

Una expresión es una serie de elementos de valor combinados con operadores, que produce un nuevo valor. Los operadores actúan sobre los elementos de valor realizando cálculos, comparaciones y otras operaciones.

**Tipos de operadores**

Visual Basic proporciona los tipos de operadores siguientes:

* Operadores aritméticos, realizan los cálculos familiares en valores numéricos, incluido el desplazamiento de sus modelos de bits.
* Operadores de comparación, comparan dos expresiones y devuelven un valor Boolean que representa el resultado de la comparación.
* Operadores de concatenación, combinan varias cadenas en una sola.
* Operadores lógicos y bit a bit en Visual Basic, combinan valores Boolean o numéricos y devuelven un resultado del mismo tipo de datos que los valores.

**Variables**

Una variable es una ubicación temporal de almacenamiento de datos dentro de un programa. En nuestro código podremos utilizar una o más variables y  contener palabras, números, fechas o propiedades. La utilidad de las variables radica en que permiten asignar un nombre corto y fácil de recordar a los datos con los que pensamos trabajar.

Las variables pueden almacenar información introducida por el usuario en tiempo de ejecución, o bien el resultado de un cálculo en específico o una porción de dato que queremos mostrar en el formulario.

La declaración implícita tiene la ventaja de ser más rápida, ya que no perderá tiempo en escribir la variable Dim, pero la gestión es a menudo peor, ya que la declaración implícita no obliga a organizar y listar las variables de antemano e impedirá que Visual Basic mande un mensaje de error cuando, posteriormente en el código del programa, utilice la variable declarada anteriormente pero introduciéndole con algún error ortográfico.