CAPÍTULO 3

CONCEPTOS DE PROGRAMACIÓN

3.1. PROGRAMAS DE COMPUTADORA

Un programa es un conjunto de instrucciones secuenciales que definen la realización de una acción en la computadora. Existen diversos tipos, pero podemos dividirlos en forma general de la siguiente manera:

- Software de sistemas.- Son indispensables para que el sistema operativo funcione. Son los programas que hacen posible el uso de la computadora, al darle las instrucciones de partida y para que reconozca todos sus accesorios y puertos.

- Software de aplicación.- Son creados para llevar a cabo tareas de un tema en específico como por ejemplo: procesadores de textos, hojas de cálculos, juegos, programas de animación, etc.

Para crear un programa, se necesitan de los lenguajes de programación. Un lenguaje de programación es el conjunto de reglas sintácticas y semánticas, que nos indica cómo se debe escribir las instrucciones de un programa, para que la computadora las pueda ejecutar.

La computadora es capaz de ejecutar un programa escrito en un lenguaje de programación fijo, llamado lenguaje máquina, pero como éste es muy difícil de escribir para una persona, se utilizan los lenguajes ensambladores de alto nivel, como C, C++, Java, Visual Basic, etc. y el conjunto de instrucciones escritas de manera secuencial en un lenguaje de programación establecido se denomina código fuente; éste es convertido a un lenguaje máquina por un compilador.

3.1.1. Programas de aplicación

A través de un software de aplicación el usuario se comunica con la computadora para llevar a cabo tareas en particular. El programa tiene secuencias e instrucciones internas (algoritmos) escritas en un lenguaje de programación (código fuente) que sirven para poder recolectar los datos del usuario y posteriormente indicar a la computadora que ejecute las acciones y muestre el resultado obtenido.

Las aplicaciones son creadas para facilitar cálculos y procesos que el usuario necesite llevar a cabo con cierta rapidez. En la ingeniería como en muchas otras áreas los programas ayudan a realizar tareas repetitivas de manera más rápida de lo que lo haría una persona con su calculadora.

Los programas de aplicación han ido evolucionando al paso del tiempo debido principalmente a que los usuarios necesitan herramientas más complejas y procesos más potentes para realizar sus tareas. En un inicio los programas eran sólo aplicaciones para los sistemas operativos sin interfaz gráfica (que estrictamente era software basado en texto y operado por comandos, ver figura 3.1); los programas se ejecutaban al teclear su nombre en la línea de comandos y el usuario introducía los datos (escritos en la línea de comandos) que el programa necesitaba para funcionar.

Los programas tuvieron un cambio radical cuando aparecieron los sistemas operativos gráficos, (Windows, Linux, Macintosh Os etc.), pues éstos incluyeron por primera vez objetos gráficos tales como ventanas, botones, textos, imágenes de gran resolución, y permitieron el uso de otro tipo de hardware como el Mouse.

Figura 3.1. Editor de líneas de comando

De la misma manera en que los programas fueron cambiando, los desarrolladores fueron creando programas cada vez más amigables y vistosos para el usuario, basados en gráficos. En la actualidad los programas de aplicación se muestran como ventanas dentro del sistema operativo. Aún existen programas que se ejecutan desde la línea de comandos, pero no es el caso del presente trabajo.

3.2. INTERFAZ GRÁFICA DE USUARIO

Las aplicaciones en la actualidad se caracterizan principalmente por tener una interfaz gráfica de usuario ("GUI" por sus siglas en ingles Graphical User Interface) que les permite interactuar con el usuario. La interfaz, que es el enlace gráfico entre el usuario y el programa, está conformada por diversos controles como formularios, menús, barras de herramientas, botones, etiquetas, cuadros de texto, listas de datos, gráficas, etc.

Por medio de la interfaz gráfica el usuario introduce datos al programa, para que éste ordene una serie de procesos a la computadora, la cual recibe las órdenes y las ejecuta. Al termino del proceso la computadora indica al programa los resultados obtenidos, y el programa a través nuevamente de la interfaz muestra al usuario cuál fue el resultado de las operaciones realizadas.

Cada programa tiene una interfaz de usuario diferente, diseñadas acorde a la tarea que van a realizar pero básicamente todos tienen la forma de una ventana.

3.2.1. Controles

Son todas aquellas formas gráficas que permiten al usuario interactuar con el programa. Existe una gran variedad de controles para una aplicación. Cada lenguaje de programación incluye sus propios controles, que en algunos casos son los mismos aunque con diferentes nombres. El lenguaje de programación usado para este trabajo fue Visual Studio .Net, de aquí que los controles que a continuación se describen pertenecen a este lenguaje.

- Formulario (*Form*)

Es el control más importante de la GUI porque es el único lugar donde se pueden colocar los demás controles; si una aplicación no tiene al menos un formulario simplemente no podrá haber interfaz gráfica de usuario. Por lo general el formulario es una ventana con el mismo aspecto que las contenidas en el sistema operativo Windows (ver figura 3.2)

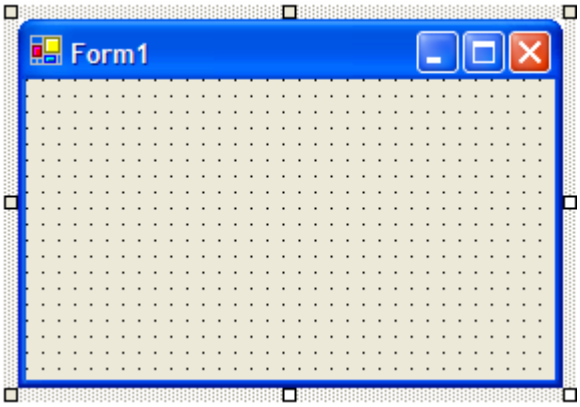


Figura 3.2 Formulario visto en modo diseño en el lenguaje de programación

- Botón (*Button*)

Es tal vez el control más conocido después del formulario, pues los botones fueron uno de los primeros controles que surgieron en la interfaz gráfica. Generalmente son de forma rectangular (figura 3.3a), pero pueden tener cualquier forma geométrica y tamaño, por ejemplo los programas que reproducen música o video tienen en general botones de forma circular (figura 3.3b). Tiene la característica que cuando se hace clic sobre ellos, parecería que se oprime de verdad.

(a) (b)

Figura 3.3 Botones

- Etiqueta (*Label*)

Las etiquetas son controles que permiten escribir texto con formato definido, y colocarlo en cualquier lugar de la interfaz gráfica. La etiqueta es un control que no se percibe visualmente, sólo su contenido que en este caso es el texto.

- Cuadro de imagen (*PictureBox*)

Así como la etiqueta permite ingresar textos a la interfaz gráfica, el *PictureBox* permite colocar imágenes dentro de la interfaz de usuario y de la misma manera este control no se percibe visualmente sólo su contenido.

- Barras de herramientas (*Tool Strip*)

Las barras son controles de forma rectangular alargadas que se encuentran generalmente en las orillas de las ventanas. Está compuesto por botones de forma cuadrada representados con imágenes.

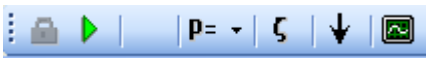


Figura 3.4 Barra de iconos o herramientas

- Barras de menús (Menú *Strip*)

Son similares a las barras de iconos, sólo que en éstas los botones son rectangulares y están representados por textos que al hacer clic en ellos se despliegan menús que son cuadros con varias opciones.

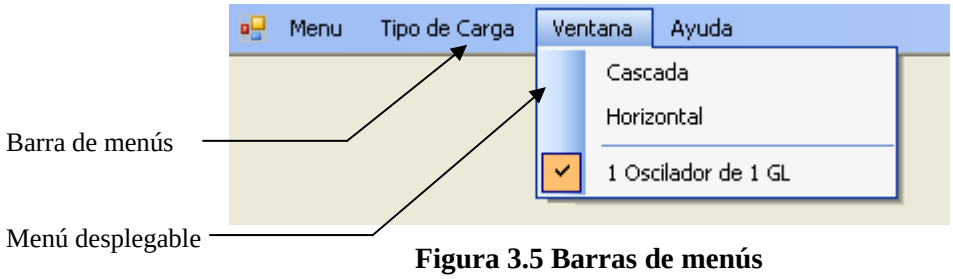


Figura 3.5 Barras de menús

- Menú contextual (*Context Menu Strip*)

Son menús en forma de ayuda emergente que aparecen al hacer clic derecho sobre un control.

- Cuadros de texto (*TextBox*)

Son controles rectangulares en los cuales se pueden escribir datos a través del teclado. Se usan principalmente para ingresar valores al programa.

Figura 3.6 Cuadros de texto y etiquetas

- Cuadros numéricos (*Numeric Up down*)

Son controles con la misma forma que los cuadros de texto, pero presentan una diferencia: que sólo se puede ingresan valores del tipo numérico y además muestran dos flechas en la parte derecha del control para incrementar o disminuir el valor.

Figura 3.7 Cuadro de texto numérico

- Cuadros de texto desplegable (*ComboBox*)

Este control combina las propiedades de un cuadro de texto y menús desplegables, muestra un texto que es una de las opciones del menú que se despliega cuando se hace clic en la flecha que tiene en su lado derecho. En ocasiones las opciones del menú pueden incrementarse con tan sólo escribir nuevos textos en el control, pero en otras, las opciones que se muestran son fijas y sólo se puede seleccionar de entre las predefinidas en el control.

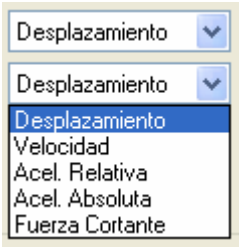


Figura 3.8 Cuadros de texto desplegable

- Cuadros de lista (*ListBox*)

Control de forma rectangular, de dimensiones fijas que muestra una lista de datos que se pueden seleccionar al hacer clic sobre ellos. Cuando el número de datos rebasan el tamaño del control, aparecen barras de desplazamiento en los extremos izquierdo e inferior del control.

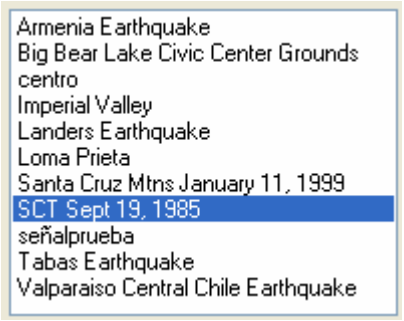


Figura 3.9 Cuadros lista

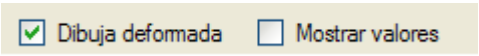
- Cajas de texto (*RichTextBox*)

Permite ingresar texto a la interfaz gráfica de usuario. A diferencia del control etiqueta (*label*) está permitido manipular su contenido; se usa principalmente para abrir archivos de texto, al crear una copia del archivo y mostrar su contenido en el control. También se puede ingresar texto escribiendo directamente sobre él. Su tamaño es ajustable y su apariencia es como el área de trabajo de un editor de texto.

Figura 3.10 *RichTextBox* mostrando el contenido de un archivo de registro sísmico

- Casillas de verificación (*CheckBox*)

Las casillas de verificación son pequeños recuadros con un texto delante de ellos que es el nombre de la opción a la que hace referencia; es posible seleccionar varios controles de este tipo a la vez. Al hacer clic sobre ellos se activa o desactiva mediante la aparición de una "palomita" sobre el recuadro, el control ya está predeterminado de esta manera.



(a) (b)

Figura 3.11 Casillas de verificación (a) Activada y (b) Desactivada

- Botones de opción (*OptionBox*)

Son muy parecidos a las casillas de verificación sólo que en este control el recuadro aparece de forma circular y cuando está activado se rellena de algún color. Sirven para activar o desactivar opciones pero a diferencia de los *Check Box* en un grupo de estos controles sólo se puede seleccionar una opción (ver figura 3.12).

- Grupo de opciones (*GroupBox*)

Su finalidad es crear y organizar a un conjunto de controles dentro de la interfaz gráfica, al que se le denomina sub grupo. Dentro de la interfaz gráfica todos los controles forman parte del grupo principal de controles, pero en algunos casos se requiere establecer sub grupos, por ejemplo para los controles de tipo *OptionBox*, esto se logra al crear un *GroupBox* y agregar los controles dentro de él.

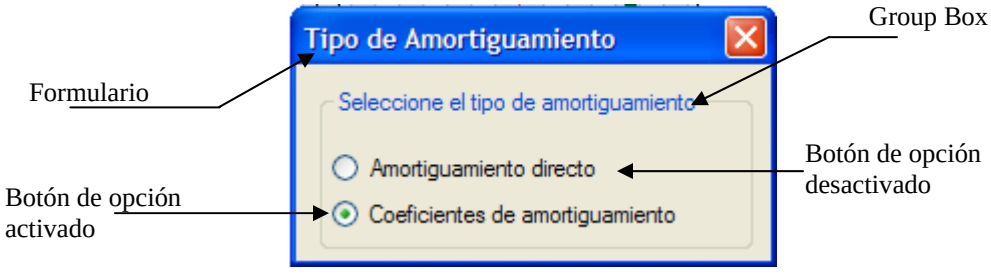
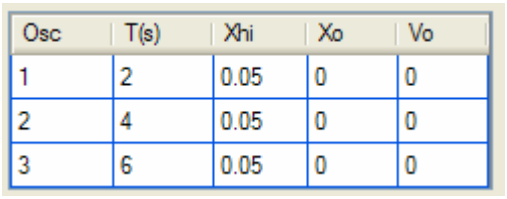


Figura 3.12 Grupo de opciones con controles del tipo *OptionBox*

- Tabla de datos (*DataGridView*)

Es una tabla con una cuadricula en su interior (figura 3.13) muy parecida a las hojas de Excel, y al igual que éstas, permite ingresar valores de cualquier tipo, seleccionar las columnas y filas, copiar su contenido al portapapeles de Windows, etc.



Osc	T(s)	Xhi	Xo	Vo
1	2	0.05	0	0
2	4	0.05	0	0
3	6	0.05	0	0

Figura 3.13 Control del tipo Tabla

- Barras de desplazamiento (*ScrollBar*)

Éstas aparecen comúnmente en los controles que su contenido rebasa las dimensiones del control. La barra asociada a un control permite desplazarse a lo largo y ancho de éste. No obstante, puede colocarse en cualquier parte de la interfaz gráfica y no estar asociadas a ningún control; son útiles en diferentes casos, por ejemplo, si se requiere establecer una relación con algún tipo de escala numérica.

3.3. CODIGO FUENTE

Es el conjunto de sentencias e instrucciones que forman al programa; se escribe por medio de líneas de texto que luego serán convertidas en lenguaje máquina por un compilador. El código fuente es la parte esencial del programa, pues todos sus componentes son creados por medio de líneas de código.

En Visual Studio, como en muchos programas ensambladores, existe un editor (figura 3.14) en el que se crea el código, línea por línea, siguiendo las reglas de sintaxis de cada programa ensamblador.

En el editor de código está escrito, en un lenguaje de programación, cada parte que conforma al programa, desde la interfaz gráfica hasta el procedimiento para obtener la solución al problema (algoritmo).

Como ya se mencionó, la interfaz de usuario es la parte gráfica del programa, pero aun esta parte es necesaria crearla a través de líneas de código; todas las propiedades (tamaño, ubicación, color, texto, valores, etc.) de cada control son definidas de esta manera pues incluso todos los controles tienen que convertirse al lenguaje máquina.

En los inicios de la programación con interfaz gráfica definir cada control directamente en el editor de código era demasiado complejo para los desarrolladores de programas, pero hoy en día los lenguajes ensambladores crean el código de manera automática para cada control una vez que se va formando la interfaz gráfica; esto ahorra mucho tiempo y trabajo a los desarrolladores.

Para formar la interfaz de usuario en Visual Studio, basta con arrastrar el control deseado desde la barra de controles hasta el formulario principal, y ubicarlos en la ventana hasta dar forma a la interfaz gráfica; al mismo tiempo el programa ensamblador va creando el código fuente en el editor de código.

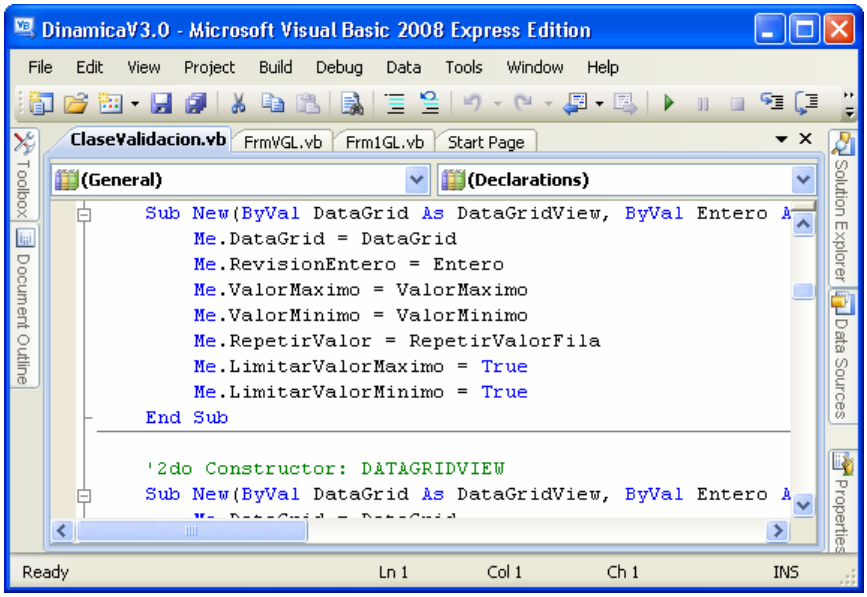


Figura 3.14 Editor de código del programa Visual Basic .Net 2008

Si bien es cierto que los programas ensambladores generan el código de la interfaz gráfica, el desarrollador debe escribir (en un lenguaje de programación) cuál es la relación entre los controles de la interfaz y el comportamiento que cada uno tendrá dentro del programa. Además, debe escribir el código fuente para el algoritmo que da la solución al problema.

Cuando el usuario interactúa con el programa, lo hace a través de la interfaz gráfica de usuario, y por lo general el código fuente de los programas de aplicación no está disponible para el usuario, que muchas veces no le interesa como fue escrito el programa sino la utilidad que tenga éste para resolver su problema.

3.3.1. Algoritmo

Es una secuencia de instrucciones cuyo objetivo es obtener la solución del problema para el que es creado el programa. No hay una única forma de resolver el problema, por lo que la idea de partida y el desarrollo del método para llegar a la solución son fundamentales para lograrlo.

Saber cuál es el algoritmo óptimo para la solución de nuestro problema puede definirse por medio de dos aspectos: un algoritmo es mejor cuanto menos tarde en resolver un problema, o bien, es mejor mientras menos memoria necesite.

El espacio en memoria, en general, tiene menos interés. El tiempo es un recurso mucho más valioso que el espacio en memoria. Así que desde este punto de vista el mejor algoritmo será aquél que resuelva el problema más rápidamente.

3.3.2. Formas de programar

Para programar es fundamental saber abstraer y descomponer los problemas en otros más pequeños, anticiparse y prever todos los posibles casos que ocurrirán. Así como los programas de aplicación han evolucionando, la forma de programar también lo ha hecho, que dicho sea de paso los programas han mejorado gracias a la forma de programar.

A continuación se hace una breve descripción de la evolución de la forma de programar, partiendo de la programación estructurada hasta la programación dirigida por eventos.

3.3.2.1. Programación Estructurada

Como ya se dijo el código fuente está escrito en líneas de texto siguiendo las reglas y estilo de un lenguaje de programación establecido. La programación estructurada surgió a finales de los años 60 y consiste en colocar las líneas de código de manera secuencial, es decir, la escritura de una instrucción debajo de otra, y el modo de procesar el programa será ejecutando las instrucciones línea por línea de arriba hacia abajo.

Para poder hacer una programación estructurada es necesario el uso de tres estructuras lógicas de control:

- Secuencia: Sucesión simple de las instrucciones, una debajo de la otra. Las instrucciones de un programa son ejecutadas en el mismo orden en que ellas aparecen en el programa.

- Selección: División condicional de las operaciones. Es la elección entre dos instrucciones tomando la decisión en base al resultado de evaluar una condición (falsa o verdadera).

- Interacción: Repetición de una operación mientras se cumple una condición

Estos tres tipos de estructuras lógicas de control pueden ser combinados para producir programas que manejen cualquier tarea de procesamiento de información. La estructura del programa mediante esta forma es clara puesto que las instrucciones están ligadas y relacionadas entre sí.

Una característica importante en un programa estructurado es que puede ser leído en secuencia, desde el comienzo hasta el final sin perder la continuidad de la tarea que cumple, contrario de lo que ocurre con otras formas de programación.

El principal inconveniente de este método de programación, es que se obtiene un único bloque de programa, que cuando el código fuente se hace demasiado grande puede resultar problemático su manejo. Para solucionar este problema se creó la programación modular.

3.3.2.2. Programación modular

Consiste en dividir al programa en módulos, para que sea más fácil de manejar y revisar. Cada modulo desempeña una acción especifica para el correcto funcionamiento del programa global. De las varias tareas que debe realizar un programa para cumplir con su objetivo, un módulo realizará una de dichas tareas.

Cuando el código fuente de un programa es demasiado grande, es conveniente crear módulos, para que el problema original sea divido en problemas más pequeños. Los módulos son creados dentro del mismo código fuente del programa y sólo pueden ser aprovechados por el código fuente de la misma aplicación.

Hoy en día los programas de aplicación son mucho más ambiciosos que las necesidades de programación existentes en los años 60, principalmente debido a las aplicaciones gráficas, por lo que las técnicas de programación estructurada y modular ya no son suficientes.

3.3.2.3. Programación Orientada a objetos (POO)

La programación orientada a objetos es una de las herramientas más poderosas con las que cuenta la programación. Básicamente consiste en crear partes de código que pueden ser reutilizables en cualquier parte del código fuente de un programa e incluso por diferentes aplicaciones; es parecida a la programación modular pero mucho más poderosa.

Durante años, los programadores se habían dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas, de manera que, cuando se creaba un programa el desarrollador tenía que volver a escribir líneas de código para problemas que probablemente ya se habían resuelto pero que no podía reutilizar. Para resolver esto y hacer que los avances de los programadores pudieran ser utilizados por otros desarrolladores se creó la POO.

De esta manera la programación avanzó a pasos agigantados, pues cada vez que un desarrollador creaba un programa podía implementar en su aplicación partes de código que alguien más ya se había tomado la molestia de desarrollar.

Un claro ejemplo de esto es la interfaz gráfica de usuario; los desarrolladores crearon (por medio de la POO), el código fuente para los controles, definiendo para éstos sus características y propiedades. Así, cada vez que un desarrollador crea una aplicación con interfaz gráfica, éste ya no se detiene para programar un control, sólo lo incorpora a su aplicación.

Su dominación fue consolidada gracias al auge de las Interfaces gráficas de usuario, para las cuales la programación orientada a objetos está particularmente bien adaptada. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

La programación orientada a objetos (POO) es un paradigma de programación que usa clases, objetos y sus interacciones para diseñar aplicaciones de computadora. Está basado en varias técnicas, incluyendo abstracción y herencia, entre otras.

- Clase

Es un modulo de código que contiene procedimientos y operaciones asociadas a un objetivo. Una clase tiene en un código fuente oculto que puede ser estructurado y modulado (maneras de programar que ya se han mencionado), y que puede ser pequeño o de gran tamaño, complejo o simple.

- Objeto

Es la manera de incorporar la clase a una aplicación. El objeto adopta las propiedades, atributos, y métodos de la clase cuando el desarrollador define al objeto, es decir se crea una instancia de esa clase.

- Abstracción

Es un concepto muy importante introducido por la POO y se puede definir como la capacidad de agregar una clase sin preocuparse de los detalles internos. En un programa es suficiente conocer que un procedimiento dado realiza una tarea específica. El cómo se realiza la tarea no es importante; mientras el procedimiento sea fiable se puede utilizar sin tener que conocer cómo funciona su interior.

- Herencia

Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Una clase puede pasar a otras sus propiedades y métodos, y éstas a otras y así sucesivamente. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. Esto hace que la POO sea una forma de programar muy poderosa.

Para entender un poco más estos conceptos, se muestra el siguiente ejemplo:

La clase *Respuesta1GL.dll*, es un conjunto de sentencias y procedimientos que calculan la respuesta dinámica de un oscilador de 1GL ante una excitación en su base.

Para incorporar la clase *Respuesta1GL.dll* a un programa desarrollado en Visual Basic.Net, se declara un objeto, en este caso *ObjCalculo* (código fuente 3.1); a través de este objeto se puede utilizar la clase mencionada dentro del programa, introduciendo los parámetros requeridos por la clase (periodo T, amortiguamiento ξ, condiciones iniciales Xo y Vo, señal de excitación, número de puntos de la señal y el intervalo de tiempo).

```
Imports CalculoRes1GL.Respuesta1GL

Dim ObjCalculo As New CalculoRes1GL.Respuesta1GL(T,ξ,Xo,Vo,SeñalExitación,Numpuntos,dt)

ObjCalculo.Resulatdos(RX1, RV1, RAr1, RAa1)
```

Código fuente 3.1 Uso de la clase en el programa

La abstracción de la clase queda de manifiesto, pues note que para usar la clase no es necesario conocer su contenido ni cómo calcula la respuesta dinámica; sólo es necesario saber que los resultados que arroja son correctos.

Ésta clase puede ser usada, si así se requiriera, para crear otras clases más sofisticada, por ejemplo una que calcule la respuesta de un oscilador de varios grados de libertad; a esta propiedad de relacionar las clases se le llama herencia.

Las clases son tan diversas, pueden ser desde procedimientos que arrojan resultados de un cálculo hasta el dibujo y comportamiento de un control gráfico.

3.3.2.4. Programación dirigida por eventos

Un evento es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). Un evento se puede definir como la reacción que desencadena un objeto, es decir la acción que genera. Por ejemplo, en el caso del control botón (que vimos que es un objeto de la clase *button*), la acción de hacer clic sobre él se denomina *evento click*.

Mencionamos anteriormente que en los programas que sólo contienen una programación estructurada y modular, son de forma secuenciales, es decir que se ejecutan línea por línea desde el principio hasta el final (de arriba abajo); en el proceso, el programa sólo se detiene cuando encuentra una línea de código que indica que el usuario debe introducir un valor, después el programa sigue hasta terminar.

La programación dirigida por eventos cambia la secuencia de ejecución del programa, es decir, el programa no realiza acción alguna hasta que se genera un evento (cualquiera que éste sea). Por ejemplo hacer clic en un botón, pasar el Mouse sobre una gráfica, escribir texto en un TextBox, etc. Hasta que se genere un evento el programa ejecuta la parte de código que tiene programada, no importa en que parte del código fuente se encuentre.

Aunque el proceso del programa sea dirigido por eventos, las partes de código contenido en los eventos siguen siendo secuenciales.

La programación por eventos no es posible sin la programación orientada a objetos.

3.3.3. Errores

Los errores son acciones inesperadas en un programa las cuales no se habían contemplado. El desarrollador debe programar todos los posibles casos para que el programa se ejecute adecuadamente. Se puede clasificar los errores en dos tipos: errores en ejecución y errores lógicos.

- Errores de ejecución

Éstos ocurren cuando la aplicación detecta acciones para las cuales no está programada. Por ejemplo, cuando en un control Textbox relacionado para introducir sólo datos numéricos recibe un texto, si el programa no contempla este caso, el programa generara un error pues normalmente el valor numérico que se esperaba es parte de una solución matemática, y en cambio si recibe una palabra el programa no puede ejecutar la ecuación pues la palabra no tiene un valor numérico.

No programar la solución a estos problemas deriva en la finalización inmediata del programa; en el peor de los casos, la aplicación se cierra perdiendo la información que se procese en ese momento.

- Errores lógicos

Son errores de programación que hacen que el código fuente del programa produzca resultados erróneos en la solución del problema. Estos errores son de mucha mayor importancia que los anteriores, pues éstos no son detectados a menos que se calibren los resultados del programa con una fuente confiable.