Common Type System

Define un conjunto común de “tipos” de datos orientados a objetos.
Todo lenguaje de programación .NET debe implementar los tipos definidos por el CTS.
Todo tipo hereda directa o indirectamente del tipo System.Object.
El CTS define tipos de VALOR y de REFERENCIA.
Las variables escalares son constantes o variable que contiene un dato atómico y unidimensional.
Las variables no escalares son array (vector), lista y objeto, que pueden tener almacenado en su estructura más de un valor.

Enteros: 0 (cero)
Punto flotante: 0 (cero)
Lógicos: False
Referencias: Null

Conversiones

Implícitas: No interviene el programador, la conversión es automática.
Explícitas: interviene el programador, ya que puede haber pérdida de datos.

Operadores lógicos

En C# todas las evaluaciones se realizan por “cortocircuito”, eso implica que siempre se evaluara la primera condición, puede evaluarse la segunda condición si la primera es verdadera o puede evaluase la segunda condición si la primera es falsa.

Entry point

El punto de entrada para los programas en C# es la función Main

static: Es un modificador que permite ejecutar un método sin tener que instanciar a una variable (sin crear un objeto). El método Main() debe ser estático.
void: Indica el tipo de valor de retorno del método Main(). No necesariamente tiene que ser void.
string [] args: Es un Array de tipo string que puede recibir el método Main() como parámetro. Este parámetro es opcional.

P.O.O.

Es una manera de construir Software basada en un nuevo paradigma. Este propone resolver problemas de la realidad a través de identificar objetos y relaciones de colaboración entre ellos.
Utiliza modelos abstractos (clases) e instancias (objetos).
Tiene 4 pilares: herencia, polimorfismo, encapsulamiento y abstracción.

Herencia

La herencia básicamente consiste en que una clase puede heredar sus variables y métodos a varias subclases. Esto significa que una subclase, aparte de los atributos y métodos propios, tiene incorporados los atributos y métodos heredados de la superclase. De esta manera se crea una jerarquía de herencia.
La relación entre clases es del tipo “es un tipo de”. Va de la generalización a la especialización.

Herencia Simple: Una clase derivada puede heredar sólo de una clase base (los lenguajes .NET soportan este tipo de herencia)

Herencia Múltiple: Una clase derivada puede heredar de una o más clases base (C++ es un ejemplo de lenguaje que soporta este tipo de herencia).

Polimorfismo

El término de polimorfismo también define la capacidad de que más de un objeto puedan crearse usando la misma clase de base para lograr dos conceptos de objetos diferentes.

- La definición del método reside en la clase base o padre.
- La implementación del método reside en la clase derivada o hija.
- La invocación es resuelta al momento de la ejecución.

Encapsulamiento

Esta característica es la que denota la capacidad del objeto de responder a peticiones a través de sus ***métodos*** o ***propiedades*** sin la necesidad de exponer los medios utilizados para llegar a brindar estos resultados.
El exterior de la clase lo ve como una caja negra.

Abstracción

Una declaración de método abstracto introduce un nuevo método virtual, pero no proporciona una implementación del método. Por esto es necesario que las clases derivadas no abstractas proporcionen su propia implementación mediante el reemplazo del método. Debido a que un método abstracto no proporciona una implementación real, el cuerpo-del-método de un método abstracto consiste simplemente en un punto y coma. Son métodos y propiedades que se declaran sin implementación.

- Ignorancia selectiva.
- Decide qué es importante y qué no lo es.
- Se enfoca en lo que es importante.
- Ignora lo que no es importante.
- Utiliza la encapsulación para reforzar la abstracción.

Clases

- Una clase es una Clasificación.
- Clasificamos en base a comportamientos y atributos comunes.
- A partir de la clasificación se crea un vocabulario.
- Es una abstracción de un objeto.

Abstract: Indica que la clase no podrá instanciarse.
Internal: Accesible en todo el proyecto (Assembly).
Public: Accesible desde cualquier proyecto.
Private: Accesor por defecto.
Sealed: Indica que la clase no podrá heredar.

Atributos

Puede ser accedido por:

private: Los miembros de la misma clase.
protected: Los miembros de la misma clase y clases derivadas.
internal: Los miembros del mismo proyecto.
internal protected: Los miembros del mismo proyecto o clases derivadas.
public: Cualquier miembro. Accesibilidad abierta.

Métodos

abstact: Solo la firma del método, sin implementar.
extern: Firma del método (para métodos externos)
internal: Accesible desde el mismo proyecto.

override: Reemplaza la implementación del mismo declarado como virtual una clase padre.
public: Accesible desde cualquier proyecto.
private: Solo accesible desde la clase.
protected: Solo accesible desde la clase o derivadas.
static: Indica que es un método de la clase.
virtual: Permite definir métodos, con su implementación, que podrán ser sobrescritos en clases derivadas.

Objetos

- Los objetos son ***clases instanciadas*.**
- Se crean en ***tiempo de ejecución***.
- Poseen Comportamiento (métodos) y Estado (atributos).
- Para acceder a los métodos o atributos se utiliza el . (punto).
- Para crear un objeto se necesita la palabra reservada NEW.
- Una vez inicializado el objeto se puede utilizar para manipular sus atributos y llamar a sus métodos.

Ciclo de vida de un objeto

Creación del objeto

Se usa new para asignar memoria.
Se usa un constructor para inicializar un objeto en esa memoria.

Utilización del objeto

Llamadas a métodos y atributos.

Destrucción del objeto

Se pierde la referencia en memoria, ya sea por finalización del programa, cambio o eliminación de la variable, etc.
El Garbage Collector liberará memoria cuando lo crea necesario.

Constructores

- Los constructores son métodos especiales que se utilizan para inicializar objetos al momento de su creación.
- En C#, la única forma de crear un objeto es mediante el uso de la palabra reservada **new** para adquirir y asignar memoria.
- Aunque no se escriba ningún constructor, existe uno por defecto que se usa cuando se crea un objeto a partir de un tipo referencia.
- Los constructores llevan el mismo nombre de la clase.

Hay dos tipos de constructores:

Constructores de instancia: que inicializan objetos (atributos NO estáticos).
Constructores estáticos: que son los que inicializan clases (atributos estáticos).

Constructores estáticos

- Son los encargados de inicializar clases.
- Sólo inicializará los atributos estáticos.
- No debe llevar modificadores de acceso.
- Utilizan la palabra reservada **static**.
- No pueden recibir parámetros.

Sobrecarga de métodos

Los métodos no pueden tener el mismo nombre que otros elementos en una misma clase (atributos, propiedades, etc.). Sin embargo, dos o más métodos en una clase sí pueden compartir

el mismo nombre. A esto se le da el nombre de **sobrecarga**.
Los métodos se sobrecargan cambiando el número, el tipo y el orden de los parámetros (se cambia la firma del método). El compilador de C# distingue métodos sobrecargados comparando las listas de parámetros.

Formularios

- Windows Forms es la plataforma de desarrollo de aplicaciones para Windows basadas en el marco .NET. Este marco de trabajo proporciona un conjunto de clases que permite desarrollar complejas aplicaciones para Windows.
- Las clases se exponen en el NameSpace System.Windows.Forms y NameSpaces asociados. Es posible crear clases propias que hereden de algunas de las anteriores.
- Un formulario Windows Forms actúa como interfaz del usuario local de Windows.
- Los formularios pueden ser ventanas estándar, interfaces de múltiples documentos (MDI), cuadros de diálogo, etc.
- Los formularios son objetos que exponen propiedades, métodos y eventos.
El concepto de Partial Class, que se incorpora en .NET 2.0, permite separar el código de una clase en dos archivos fuentes diferentes.
- Su ciclo de vida consta de 7 etapas: New, Load, Activated, FormClosing, FormaClosed y Dispose.

Colecciones

Existen dos formas de agrupar objetos: mediante la creación de matrices de objetos y mediante la creación de colecciones de objetos. Las matrices son muy útiles para crear y trabajar con un número fijo de objetos fuertemente tipados. Las colecciones proporcionan un método más flexible para trabajar con grupos de objetos.
A diferencia de las matrices, el grupo de objetos con el que trabaja puede aumentar y reducirse dinámicamente a medida que cambian las necesidades de la aplicación.
Una colección es una clase, de modo que antes de poder agregar elementos a una nueva colección, debe declararla.
Una colección genérica cumple la seguridad de tipos para que ningún otro tipo de datos se pueda agregar a ella. Cuando se recupera un elemento de una colección genérica, no tiene que determinar su tipo de datos ni convertirlo.

Colecciones Genericas

Se puede crear una colección genérica utilizando una de las clases en el espacio de nombres System.Collections.Generic. Una colección genérica es útil cuando todos los elementos de la colección tienen el mismo tipo de datos.

- Dictionary: Representa una colección de pares de clave y valor que se organizan por claves.
- List: Representa una lista de objetos que pueden ser obtenidos mediante un índice. Proporciona métodos para buscar, ordenar y modificar listas.
Queue: Representa una colección de objetos con el orden primero en entrar, primero en salir (FIFO).
SortedList: Representa una colección de pares de clave y valor que se ordenan por claves según la implementación de la interfaz IComparer<T> asociada.
Stack: Representa una colección de objetos con el orden último en entrar, primero en salir (LIFO).

Colecciones no Genericas

Son las incluidas en el espacio de nombres System.Collections. Estas no almacenan los elementos como objetos de un tipo específico, sino como objetos de tipo Object. Siempre que sea posible, se deberían utilizar las colecciones genéricas de otros tipos en lugar de estas.

- ArrayList: Representa una matriz de objetos cuyo tamaño aumenta dinámicamente según sea necesario.
- Hashtable: Representa una colección de pares de clave y valor que se organizan por código hash de la clave.

- Queue: Representa una colección de objetos con el orden primero en entrar, primero en salir (FIFO).
- Stack: Representa una colección de objetos con el orden último en entrar, primero en salir (LIFO).

Clase Sellada

- La mayor parte de las clases son autónomas y no están diseñadas para que otras clases deriven de ellas.
- Para que el programador pueda comunicar mejor sus intenciones al compilador y a otros programadores, C# permite declarar una clase como ***sealed*** (sellada).
- La derivación de una clase sellada no está permitida (no se puede heredar de ella).

Redefinición de métodos

- El modificador override es necesario para ampliar o modificar la implementación abstracta o virtual de un método, propiedad, indixador o evento heredado.
- Un método override proporciona una nueva implementación de un miembro que se hereda de una clase base. El método invalidado por una declaración override se conoce como método base invalidado.
- El método base reemplazado debe tener la misma firma que el método override.
- No se puede reemplazar un método estático o no virtual. El método base reemplazado debe ser virtual, abstract u override.
- Una declaración override no puede cambiar la accesibilidad del método virtual. El método override y el método virtual deben tener el mismo modificador de nivel de acceso.
- No se pueden usar los modificadores new, static o virtual para modificar un método override.
- Una declaración de propiedad de invalidación debe especificar exactamente el mismo modificador de acceso, tipo y nombre que la propiedad heredada, y la propiedad invalidada debe ser virtual, abstract u override.