**Common Typo System (CTS)**

* Define un conjunto común de tipo 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 REFERENCIA.

**Tipos de Datos**

* 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 vector.

**Conversiones**

* Implícita: No interviene el programador.
* Explícita: interviene el programador, ya que puede haber pérdida de datos.

**P.O.O**

* Es una manera de construir software basada en un nuevo paradigma.
* Propone resolver problemas de realidad a través de identificar objetos y relaciones de colaboración entre ellos.
* El Objeto y el Mensaje son sus elementos fundamentales.

**ABSTRACCION**

* El modificador abstract se utiliza para indicar que una está incompleta y que sólo se va a utilizar como una clase base.
* Las clases abstractas se situan en la cima de la jerarquía de clases.
* Establecen la estructura y significado del código.

**ENCAPSULAMIENTO**

* Se denomina encapsulamiento al ocultamiento de los datos de un objeto de manera que solo se pueda cambiar mediante las operaciones definidas para ese objeto.
* 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 brindar estos resultados.
* El exterior de la clase lo ve como una caja negra.

**HERENCIA**

* La herencia permite crear nuevas clases más especializadas a partir de otras ya existentes más generales.
* Las clases derivadas son versiones especializadas de las clases base. (Son del tipo de la clase base).
* Se hereda TODO menos los constructores y finalizadores.
* Una clase derivada no puede ser más accesible que su clase base.

**POLIMORFISMO**

* 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.
* Es la capacidad de tratar objetos diferentes de la misma forma.
* Implica la definición de métodos en una clase base y sobrescribirlos con nuevas implementaciones en clases derivadas.

**Clase**

* Clase es una clasificación.
* Clasificamos en base a comportamiento y atributos comunes.
* A partir de la clasificación se crea un vocabulario.
* Es una abstracción de un objeto.

**NameSpace**

* Agrupación lógica de clases y otros elementos.
* Proporcionan un marco de trabajo jerárquico sobre el cual se construye y organiza todo el código.
* Su función principal es la organización del código para reducir los conflictos entre nombres.
* Esto hace posible utilizar en un mismo programa componentes de distinta procedencia.

**Objetos**

* Son clases instanciadas.
* Se crean en tiempo de ejecución.
* Poseen comportamientos (métodos) y estados (atributos).
* El tiempo de vida de un objeto dinámico no está vinculado en su ámbito. (tiempo de vida mas largo. No determinista)

**Las memorias y tipos de datos**

* El CLR administra dos segmentos de memoria: Stack (Pila) y Heap (Montón)
* El Stack es liberado automáticamente, y el Heap es administrado por el Garbage Collector.
* Los tipos VALOR se almacenan en Stack.
* Los tipos REFERENCIA se almacenan en Heap.

**Variables**

* Creación y destrucción determinista.
* Tiempos de vida muy cortos por lo general.

**Constantes**

* Una constante es otro tipo de campo.
* Contiene un valor que se asigna cuando se compila el programa y nunca cambia.

**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.
* Constructor de instancia: Inicializan objetos.
* Constructor estático: Inicializan las clases (sólo atributos estáticos, sin modificador de acceso ni parámetros).

**Sobrecarga de métodos**

* Los métodos se sobrecargan cambiando el numero, 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.

**Operadores de conversión**

* Las conversiones definidas permiten hacer compatibles tipos que antes no lo eran.
* Los operadores de conversión pueden ser explícitos o implícitos.

**Windows Forms**

* Un formulario Windows Forms actúa como interfaz del usuario local de Windows.
* Los formularios son objetos que exponen propiedades, métodos que definen su comportamiento y eventos que definen la interacción con el usuario.
* El concepto Partial Class permite separar el código de una clase en dos archivos fuentes diferentes. Se utiliza para mantener separada la parte lógica del diseño.

**Arrays**

* Un array puede ser unidimensional, multidimensional o anidado.
* Los Arrays son de reference type, heredan de la clase abstracta System.Array.
* Implementan la interfaz IEnumerable, por lo tanto pueden iterar usando un foreach.

**Colecciones**

* Las colecciones proporcionan un método flexible para trabajar con grupos de objetos.
* A diferencia de las matrices, el grupo de objetos con el que trabaja puedo aumentar y reducirse dinámicamente a medida que cambian las necesidades de la aplicación.
* Una colección es una clase.

**Colección Genérica**

* Son las incluidas en el NameSpace System.Collection.Generic.
* Una colección genérica es útil cuando todos los elementos de la colección tienen el mismo tipo de datos.
* Una colección genérica cumple la seguridad de tipos para que ningún otro tipo de datos pueda agregarse a ella.
* Dictionary: Representan 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.
* SortedList: Representan una colección de pares de clave y valor que se organizan por claves según la implementación de la interfaz IComparer<T> asociada.
* Stack: Representa una colección de objetos con el orden: ultimo en entrar, primero en salir.

**Colección NO Genérica**

* Son las incluidas en el NameSpace System.Collection.
* Estas no almacenan los elementos como un tipo especifico, sino como objetos del tipo Object.
* Arraylist, Hashtable, Queue, Stack.