**Common Type System (CTS)**Reference types**:** Clase, Interfaz, Array, String, Delegado, OtrosValue types: Byte, Char, Int, Decimal, Boolean, UInt, Single, Double, Enum, otros.

**Pilares de la programación orientada a objetos:**

* **Abstracción:** Ignorancia selectiva, decide que es lo importante y que no, utiliza la encapsulación para reforzar la abstracción.
* **Encapsulamiento:** Esta característica denota la capacidad del objeto para responder peticiones a través de sus métodos o propiedades sin la necesidad de exponer los medios utilizados para llegar a brindar estos resultados (Caja negra desde el exterior)
* **Herencia:** La herencia es uno de los conceptos más cruciales en la POO. La herencia básicamente consiste en que una clase puede heredar sus variables y métodos a varias subclases (la clase que hereda es llamada superclase o clase padre). 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.
* **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, en este caso podemos citar el típico ejemplo de los teléfonos, los cuales se basan en un teléfono base, con la capacidad de hacer ring y tener un auricular, para luego obtener un teléfono digital, inalámbrico, con botonera de marcado y también, tomando la misma base, construir un teléfono analógico y con disco de marcado.

**Definición de “clase”:**Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase.  
Es un conjunto de objetos que tienen características comunes. Por ejemplo, Ford Focus como un Honda Civic o un Toyota Corolla son objetos concretos que pertenecen a la clase "coche".

**Modificadores de las clases:**  
- abstract: Indica que la clase no podrá instanciarse.  
- internal: Accesible en todo el proyecto.  
- public: Accesible desde cualquier proyecto.  
- private: Acceso por defecto.  
- sealed: Indica que la clase no podrá heredar.

**Modificadores de los atributos:**- private: Los miembros de la misma clase.- protected: Los miembros de la misma clase y derivadas o hijas.  
- internal: Los miembros del mismo proyecto.  
- internal protected: Los miembros del mismo proyecto o clases derivadas.  
- public: Cualquier miembro, accesibilidad abierta.

**Modificadores de los métodos:**- abstract: Sólo 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 método declarado como virtual en una clase padre.  
- public: Accesible desde cualquier proyecto.  
- private: Sólo accesible desde una clase.  
- protected: Sólo accesible desde la clase o derivadas.  
- static: Indica que es un método de clase.  
- virtual: Permite definir métodos, con su implementación, que podrán ser sobrescritos en clases derivadas.

**Conversión implícita:** No interviene el programador  
float flotante = 15;

**Conversión explícita:** Interviene el programador ya que puede haber perdida de datos  
int entero = (int)15.2;

**Static:** Es un modificar que permite ejecutar un método sin tener que instanciar a una variable (sin crear un objeto). Por ejemplo, el método Main(), que es estático. No se puede utilizar el this.

**Constructores de instancia:** Que inicializan objetos (atributos no estáticos). Se puede usar el this.

**Constructores estáticos:** Que inicializan clases (atributos estáticos). No debe llevar modificadores de acceso. Utilizan la palabra reservada static. No pueden recibir parámetros.

**Console:** Es una clase pública y estática. Miembro del NameSpace System.

**Parse vs TryParse:** Parse lanza una excepción si no pudo “parsear” el valor. En cambio, TryParse retorna un booleano indicando lo que sucedió.  
bool success = Int32.TryParse(value, out number);

**Sobrecarga de operador:**  
Sobrecargar un operador consiste en modificar su comportamiento cuando este se utiliza con una determinada clase. Operadores sobrecargables: Unarios, binarios y de comparación.  
[acceso] static TipoRetorno operator nombreOperador (Tipo a[, Tipo b]){ //… }