Utilizando la bibliografía indicada en clase conteste las siguientes preguntas orientadoras

1. ¿Qué es un objeto?  
   Objeto: Se trata de un ente abstracto usado en programación que permite separar los diferentes componentes de un programa, simplificando así su elaboración, depuración y posteriores mejoras.
   1. Describa sus características:

Se compone principalmente de: métodos, eventos y atributos

Posee: identidad, comportamiento y estado.

* 1. ¿Qué puede ser un objeto?
     1. Algo físico
     2. Algo abstracto
     3. Algo del dominio del problema
     4. Algo del dominio de la solución

1. Explique los conceptos de:
   1. Mensaje: se llama mensaje a una porción de información que un proceso emisor envía a un destinatario. Un texto que se le emite a un destinatario que suele ser el usuario que usa la aplicación.
   2. Colaboración: Interacción de clases u objetos externos a la clase u objeto principal
   3. Responsabilidad: La principal actividad o lo que tiene que hacer. Básicamente es lo que el objeto sabe que tiene que hacer.
2. ¿Qué es una clase?

Es una matriz o plantilla para “crear” objetos del mismo tipo.

* 1. ¿Cómo se compone?
     1. Atributos
     2. Métodos
     3. Eventos
     4. Relaciones

1. Una relación en UML refleja un vínculo existente entre elementos.
   1. Asociación 🡪 Simple
   2. Agregación/composición
   3. Generalización/herencia
   4. Dependencia
   5. Realización
2. Explique los pilares del POO
   1. Abstracción: es omitir detalles que no son necesarios para nosotros, y solamente mostrar lo que sí es relevante.
   2. Herencia: relación entre clases, “Es un tipo de”
      1. ¿Qué tipos existen?

Herencia simple: una clase hereda únicamente de una clase.

Herencia múltiple: una entidad deriva de dos o más clases, sin importar los niveles de herencia.

* 1. Encapsulamiento: Mecanismo que permite ocultar los detalles de implementación de un objeto.
     1. ¿Cómo se vincula con la abstracción?
  2. Polimorfismo: capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros (diferentes implementaciones) utilizados durante su invocación.
     1. ¿Cómo se logra?

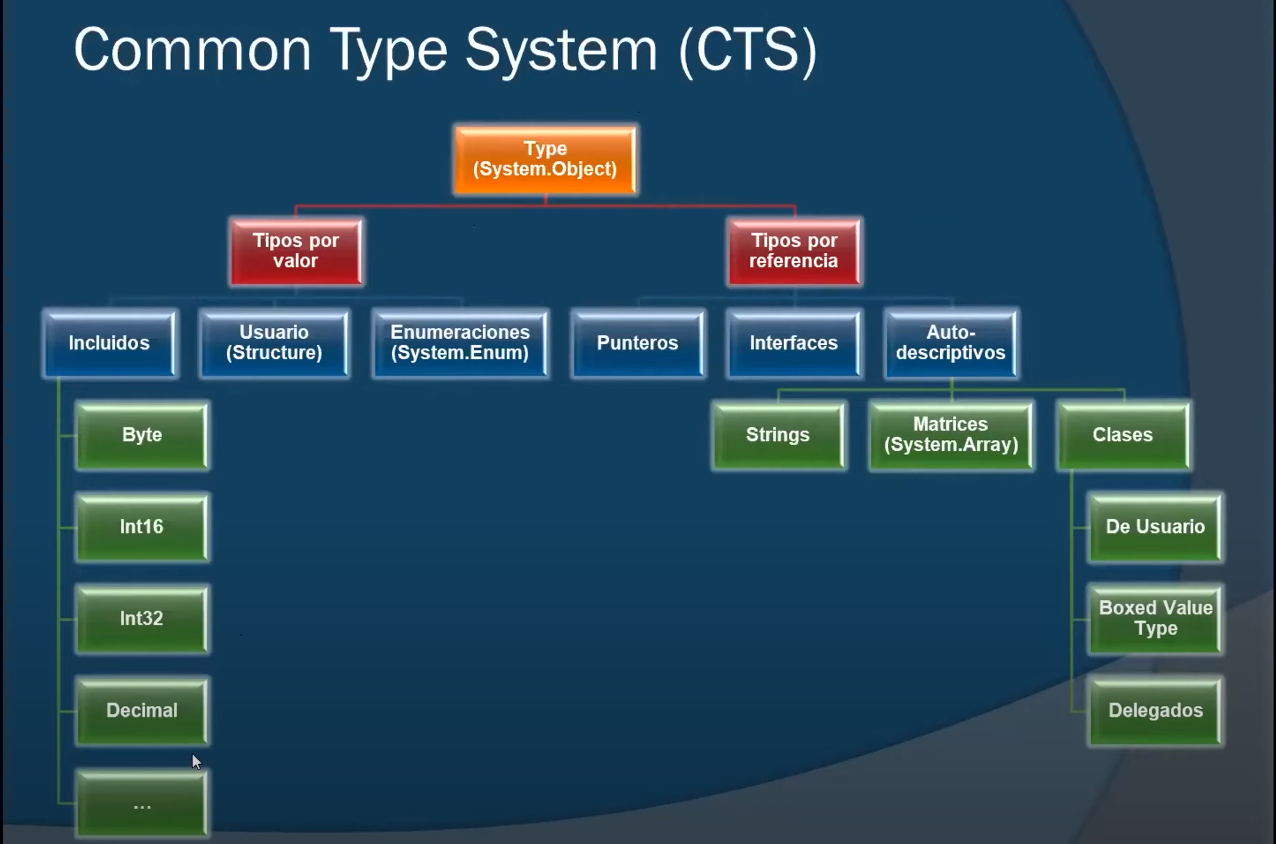
Una forma de conseguir objetos polimórficos es mediante el uso de punteros a la superclase.

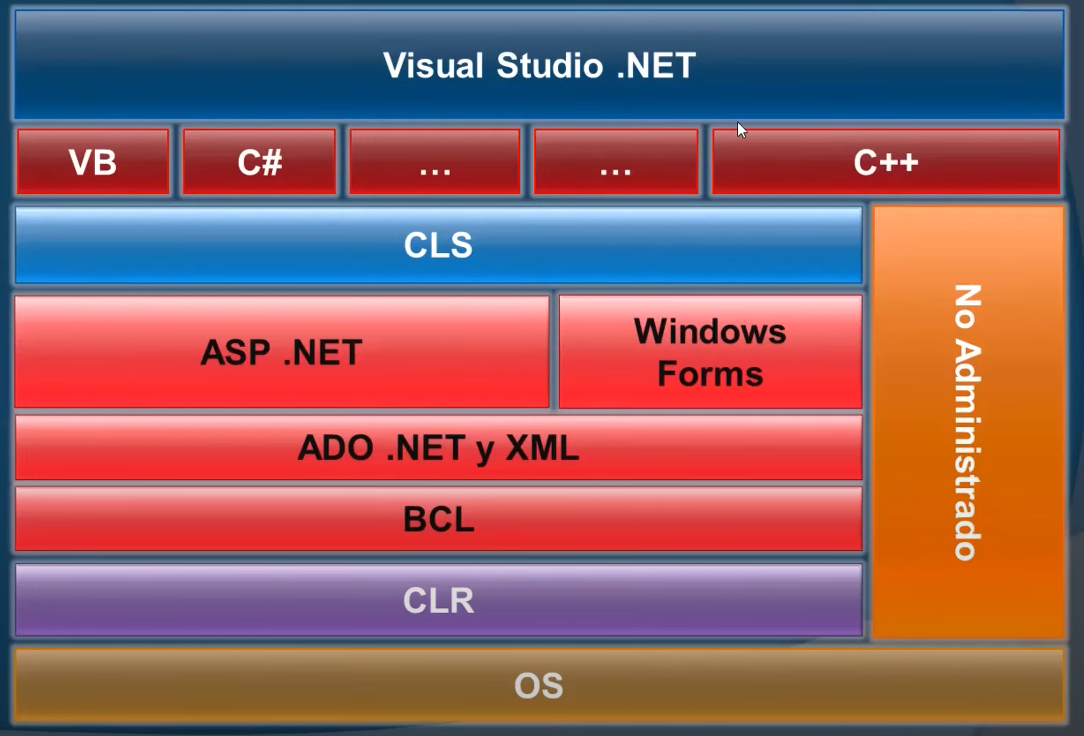
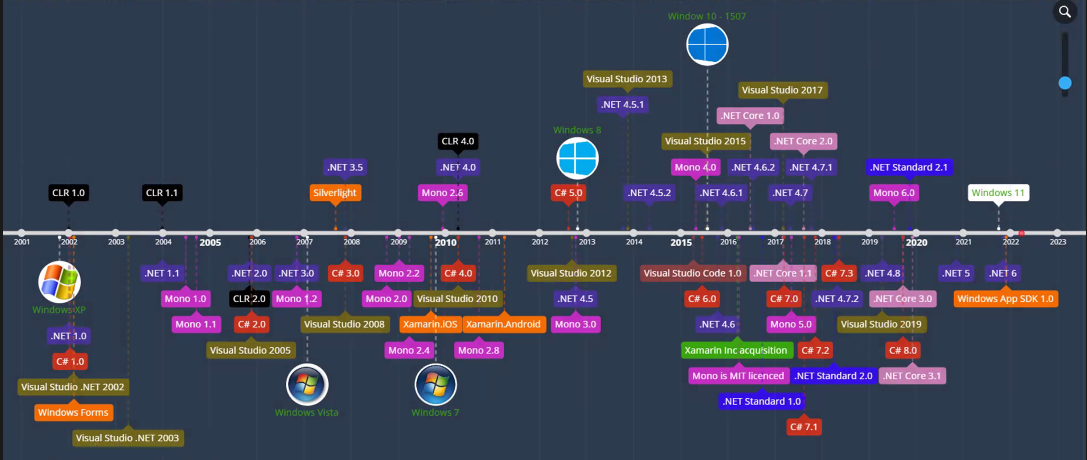
1. Explique brevemente de que se trata cada una de las tecnologías que precedieron a .NET. Ventajas y Desventajas.
   1. C/Windows API: Desarrollo tradicional: Lenguaje de bajo nivel. Paradigma estructural
   2. C++/MFC: Orientado a objetos. La MFC funciona como un envoltorio para la Windows API.
   3. Visual Basic 6.0: RAD mínimo esfuerzo. Tiene un nivel de abstracción mayor. Permitía desarrollar app para Windows con un nivel menor de complejidad. RAD: desarrollo rápido de aplicaciones, sirve para prototipar algo para ofrecer al cliente.
   4. COM: (component object model) primer intento de Microsoft para llegar a unificar las tecnologías dentro del entorno de programación. Permitía crear objetos que puedan ser reutilizables por diferentes apps.
2. ¿Qué diferencia existe entre una plataforma y un framework?
   1. Una plataforma involucra:
      1. HW
      2. SW: SO, Entorno de desarrollo, etc.
   2. Un framework es un conjunto de librerías para determinado trabajo es decir un marco de trabajo.
3. En particular, ¿qué es el .NET Framework y qué tipo de aplicaciones se pueden desarrollar?

Objetivos de .NET Framework: Consistente, minimice problemas, segura, apunta al performance, integrable.

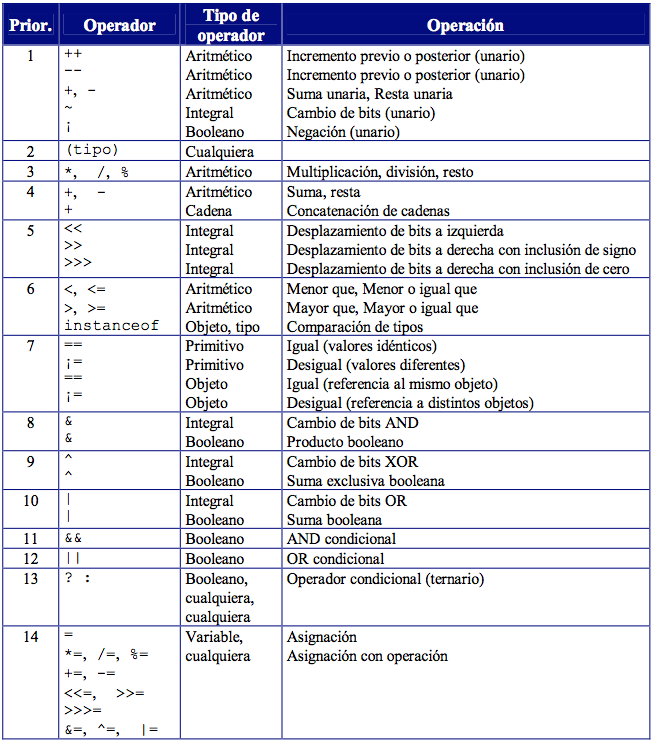


1. Responda las siguientes preguntas acerca de .NET Framework/.NET Core/.NET
   1. ¿Cuáles son sus dos partes principales?
      1. Common Lenguage Runtime (CLR): motor de ejecución (runtime) del .NET Framework. Localiza, carga y administra objetos .NET.   
         Proporciona la administración de la memoria y otros servicios del sistema, y una biblioteca de clases completa, que permite a los programadores aprovechar el código estable y fiable de todas las áreas principales del desarrollo de aplicaciones.
      2. Base Class Library (BCL): Es una colección de tipos reusables. Permiten dividir funcionalidades útiles en módulos que pueden usar varias aplicaciones. También se pueden usar para cargar la funcionalidad no necesaria o no conocida al inicio de la aplicación.
   2. ¿Qué es el CTS y cuál es su importancia?

CTS (Common Type System): define un conjunto común de “tipos” orientados a objetos. Conjunto de tipos primitivos que todos los lenguajes compatibles con .NET entienden, permitiendo así, una interoperabilidad entre ellos.  


* + 1. ¿Cómo se vincula con la CLS?  
       CLS (Common Language Specification): es una especificación justamente a la que se pueden adherir los lenguajes compatibles con .NET. El CLS es un subconjunto del CTS y su implementación es opcional.  
       En otras palabras: Define un conjunto mínimo de tipos comunes y construcciones de programación.
    2. 
  1. ¿Qué versiones del framework fueron liberadas en el tiempo? 

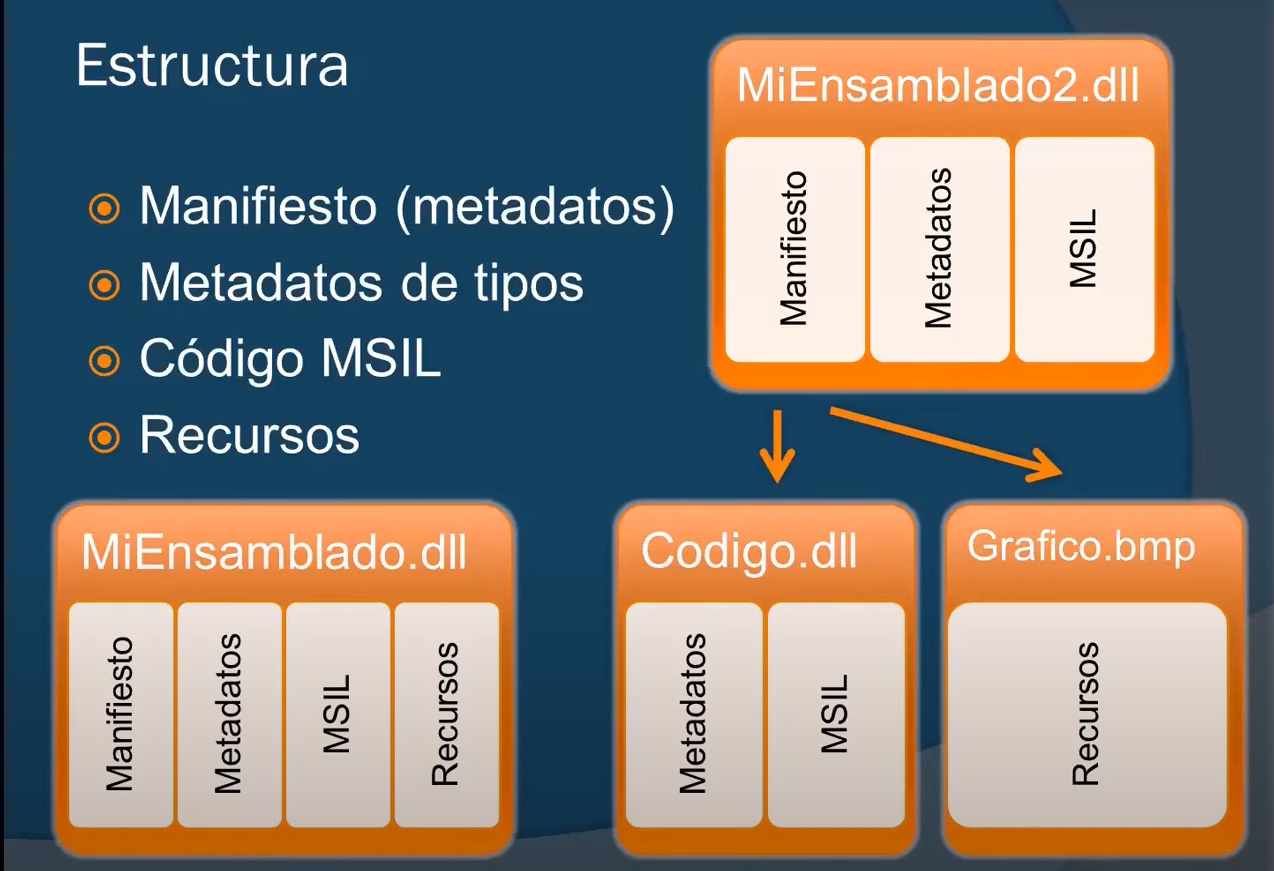
1. Responda las siguientes preguntas acerca de .NET Framework/.NET Core/.NET
   1. Caracterice a C# y sus versiones
      1. 1.0: basado en C++ y VB6. Propiedades y arreglo de parámetros, sobrecarga, enumeraciones, callbaks (via delegates).
      2. 2.0: Generics, métodos anónimos (funciones inline).
      3. 3.0: incializacion de objetos inline, tipos anónimos, métodos de extensión, lambda, LINQ.
2. IDE:
   1. ¿Qué es Visual Studio?  
      IDE de Visual Studio es una plataforma de lanzamiento creativa que puede utilizar para editar, depurar y compilar código y, finalmente, publicar una aplicación.
3. Operadores en C#:



1. Ensamblados:
   1. ¿Qué es un ensamblado?

Es un bloque de construcción básico de .NET Framework. es principalmente una biblioteca de código compilado para ser utilizado en instalaciones, versionamiento y seguridad.

* 1. ¿Qué funciones realiza?
     1. Contenedor de código: MSIL 🡪 PE 🡪 Punto de entrada
     2. Limite de: seguridad, tipos, ámbito de referencia, versión
     3. Unidad de implementación
     4. Ejecución simultanea
  2. Describa su estructura
     1. Manifiesto: contiene todos los metadatos necesarios para especificar los requisitos de versión y la identidad de seguridad del ensamblado, y todos los metadatos necesarios para definir el ámbito del ensamblado y resolver las referencias a los recursos y las clases.
     2. Metadatos de tipos:   
        Tipos🡪 clases, estructuras, enumeración, etc.  
        Miembros 🡪 propiedades, métodos, eventos, etc.  
        Usado por 🡪 compilador, intellisense.
     3. MSIL: Cuando se compila a código administrado, el compilador convierte el código fuente en Lengua intermedio de Microsoft (MSIL), que es un conjunto de instrucciones independiente de la CPU que se pueden convertir de forma eficaz en código nativo.

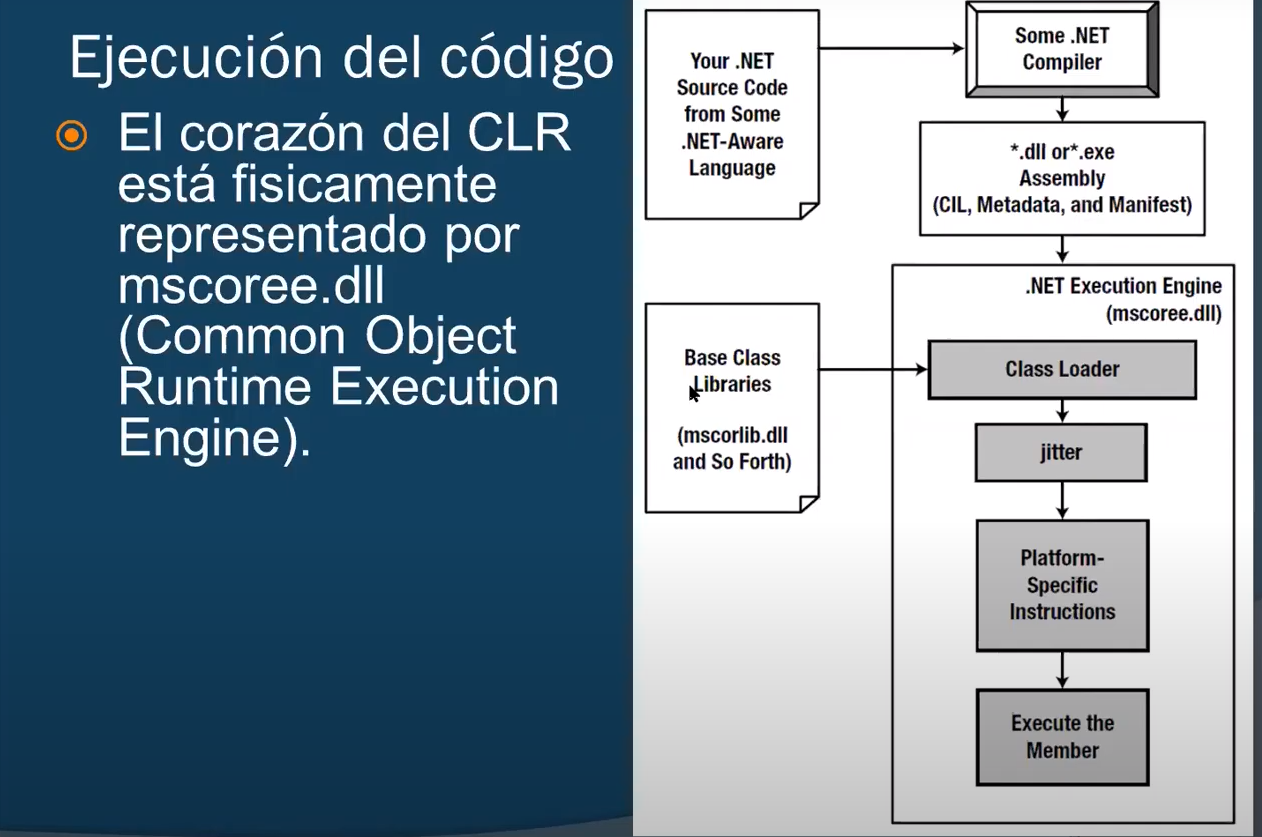


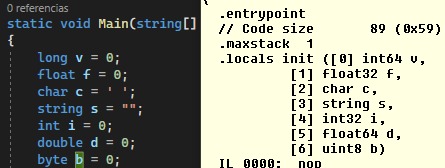
* 1. ¿Según su disponibilidad entre aplicaciones como se clasifican los ensamblados?
     1. Privados
     2. Compartidos
  2. ¿Qué es la GAC?   
     GAC (Global Assambly Cache): es el almacenamiento de ensamblados firmados ubicados en la máquina del usuario.
  3. ¿Cuál es el proceso de ejecución de del código administrado?  
     

Compilación MSIL: al compilar convertimos el código fuente en MSIL y METADATOS

Compilación a código nativo: antes de poder usar MSIL debe convertirse a código de máquina, actúa el compilador JIT adecuado.

Compilador JIT: (Just-In-Time) es un componente del entorno de ejecución que mejora el rendimiento de aplicaciones Java™ compilando códigos de bytes en código de máquina nativo en tiempo de ejecución.

Ejecución del código:   


* 1. ¿Cuáles son los principales tipos de ensamblados compilados?  
     

1. Función Split: El método String.Split crea una matriz de subcadenas mediante la división de la cadena de entrada en función de uno o varios delimitadores.
2. Función Replace: Strings.Replace(String, String, String, Int32, Int32, CompareMethod) Devuelve una cadena en la que la subcadena especificada se reemplaza determinado número de veces por otra subcadena.