*Programación Orientada a Objetos.*

*Carpeta Teórica.*

# Unidad 1

1. Los aspectos que hacen a un sistema complejo son:
   1. La complejidad del dominio del problema= la dificultad presente al realizar la toma de requisitos y la interpretación de las necesidades de los usuarios.
   2. La dificultad del proceso de desarrollo= al volverse tan grandes los sistemas, se requiere un número mayor de gente para desarrollarlos, lo que vuelve más difícil la comunicación y coordinación.
   3. La flexibilidad que se puede alcanzar con el software= A diferencia de otras industrias, en el software no hay estándares, volviendo al desarrollo más flexible pero también más laborioso.
   4. Los problemas que caracterizan el comportamiento de sistemas discretos= los sistemas que se implementan en ambientes digitales tienen estados discretos, de una cantidad finita, que se agranda de manera exponencial cuanto más grande él es, aumentando la posibilidad de que haya sorpresas ocultas. Por eso se busca que las partes de un sistema tengan el menor grado de acoplamiento posible.
2. Los 5 atributos de un sistema complejo son:
   1. Jerarquía.
   2. Componentes primitivos.
   3. Acoplamiento y cohesión.
   4. Todo sistema complejo nace de un sistema simple.
   5. Los sistemas jerárquicos están compuestos de subsistemas.
3. Tipos de jerarquía:
   1. Jerarquía estructural (parte de) -> Objetos.
   2. Jerarquía de tipos (es un) -> Clases.
4. La forma de enfrentar la complejidad es a través de la descomposición.
5. Las 2 formas de descomposición más conocidas son la algorítmica y la orientada a objetos.
6. La descomposición logarítmica se enfoca en el análisis de los procesos que ocurren en el sistema mientras que la descomposición orientada a objetos se basa en el análisis de las entidades y como se relacionan entre ellas.
7. La abstracción nos permite remover o descartar los detalles que no sean necesarios, de los objetos con los que tratamos.
8. La jerarquía nos sirve para, mediante una clasificación de abstracciones relacionadas (entidades), llegar a una distinción explicita de las propiedades comunes y distintas entre diferentes objetos.
9. Desde mi punto de vista el diseño orientado a objetos es evolutivo. Debido al progreso de la tecnología y al surgimiento de nuevas tecnologías se pasó a un diseño que explica como diversas entidades se relacionan entre si permitiendo una mayor abstracción, cohesión y adaptabilidad comparado con el diseño algorítmico.
10. Hay 4 modelos básico en orientado a objetos. Modelo físico, lógico, estático y dinámico.
11. La programación orientada a objetos es un método de implementación en el que los programas se organizan como colecciones de objetos, que representan la instancia de una clase, las cuales forman una jerarquía en base a herencia.
12. El diseño orientado a objetos abarca el proceso de descomposición orienta a objetos y una notación para describir los modelos lógico y físico, así como los modelos estático y dinámico.
13. El análisis orientado a objetos se basa en analizar los requisitos desde el punto de vista de las clases y objetos que se encuentran en el dominio del problema.
14. Jerarquía, encapsulamiento, abstracción y modularidad.
15. Tipos, concurrencia y persistencia.
16. Ignorar lo no esencial de un objeto.
17. El encapsulamiento es ocultar información de un objeto usando métodos.
18. La modularidad es dividir el programa en partes con una alta cohesión y fronteras bien definidas, para producir a arquitectura del sistema.
19. La jerarquía es el ordenamiento y la clasificación de abstracciones (estructura de clases y objetos) que están relacionados por herencia.
20. La tipificación es la definición de estructuras con características específicas que son determinadas por las clases.
21. Concurrencia es básicamente que procesos están utilizando recursos e hilos y que procesos están inactivos.
22. Persistencia es la propiedad de un objeto para perdurar en tiempo y espacio, por ejemplo, al terminar el programa, por ser guardado en un archivo.
23. Las características esenciales de un objeto que lo distinguen de los demás son su comportamiento, su estado y su identidad.
24. Un objeto cliente es aquel que puede operar sobre otros objetos pero nunca se opera sobre el por parte de otros objetos.
25. Un objeto servidor nunca opera sobre otros objetos, solo otros objetos operan sobre él.
26. Meyer denomina modelo contractual de programación a las responsabilidades que tiene un objeto y a la relación que tiene con los demás objetos.
27. Un contrato entre objetos establece un protocolo. Es decir, el conjunto de operaciones que puede realizar un objeto cliente sobre un objeto servidor, junto con las formas de invocación y las ordenes que admite.
28. Se denomina protocolo.
29. Se denomina protocolo.
30. El concepto de la invariancia es clave en el modelo contractual porque establece una condición booleana que siempre debe cumplirse para que el contrato pueda mantenerse. Habiendo así, pre y post condiciones.
31. Para cualquier operación asociada con un objeto se debe definir las precondiciones y las postcondiciones.
32. Una precondición es un conjunto de invariantes a cumplir para poder iniciar la operación.
33. Las postcondiciones es un conjunto de invariantes que se deben satisfacer al terminar la operación.
34. Una excepción se produce cuando alguno de los 2 tipos de invariantes no a sido cumplido.
35. Se denomina mensaje a lo que los objetos se mandan entre sí, ya sea parámetros, u ordenes, como la invocación de un método.
36. La abstracción y el encapsulamiento son complementarios porque mientas que la abstracción se dedica al comportamiento observable de un objeto, el encapsulamiento se encarga de la implementación (métodos) que da lugar a ese comportamiento. Es decir, oculta la información a través de métodos para abstraer al objeto.
37. La interfaz de un objeto captura su vista externa.
38. La implementación de una clase captura su vista interna y los mecanismos que determinan su comportamiento.
39. Ocultar los detalles de implementación se refiere a evitar el uso inapropiado de los detalles de la implementación, porque , se busca agregar una capa mas de abstracción, para limitar el acceso a los mecanismos internos de la clase. Esto no afecta al usuario de la clase debido a que el comportamiento de la misma no se ve afectado, pero se protege su funcionamiento interno.
40. Los aspectos que son esenciales en la modularidad son la creación de la arquitectura de un programa y la posibilidad de analizar cada módulo por separado, permitiendo modificaciones y reutilización de código.
41. La jerarquía se utiliza para ordenar lógicamente las abstracciones de un sistema.
42. Se denomina herencia.
43. No, los tipos no implementan clases.
44. Si las clases implementan tipos.
45. Lenguajes de fuerte tipado.
46. Lenguajes de comprobación de tipos débil.
47. La ligadura estática o temprana se refiere a que se define el tipo de una variable en tiempo de compilación.
48. La ligadura tardía o dinámica se refiere a ligación de tipos en tiempo de ejecución.
49. No, la ligación dinámica y la comprobación estricta de tipos no es lo mismo. Uno se refiere al momento en el cual se define y el otro se refiere a la congruencia de la asignación.
50. La concurrencia es la característica que permite a diferentes objetos actuar al mismo tiempo.
51. La concurrencia pesada es un proceso manejado de forma independiente por el sistema operativo de destino, que abarca su propio espacio de direcciones.
52. La concurrencia liviana es cuando un proceso existe dentro de otro proceso del sistema operativo en compañía de otros procesos ligeros, que comparten el mismo espacio de direcciones.
53. Si, la concurrencia diferencia a los objetos que están activos de los que no.
54. la característica que permite conservar los objetos en tiempo y espacio es la persistencia.
55. Se persiste: resultados de expresiones, variables locales, variables globales, datos de ejecución, datos de varias versiones de un programa y datos que sobreviven al programa.
56. Desde la perspectiva humana un objeto es una cosa tangible, que puede comprenderse intelectualmente y algo hacia lo que se dirige un pensamiento o acción.
57. Los objetos pueden ser reales o abstractos, pueden representar algo físico o algo abstracto que surja de la solución al dominio del problema.
58. Los objetos, para poder serlo, deben tener fronteras bien definidas, por eso las emociones no son objetos, aunque sean abstractos.
59. Las 3 cosas que debe tener un objeto son estado, comportamiento e identidad.
60. Sinónimo de objeto: Tipo.
61. Sinónimo de instancia: implementación.
62. El estado de un objeto está definido por los valores de sus atributos en un tiempo determinado.
63. La propiedad de un objeto es una característica que ayuda en su definición.
64. Lo que distingue a un objeto de los valores simples es su comportamiento.
65. El comportamiento de un objeto es como actúa este objeto al interactuar con otros objetos y al cambiar de estado.
66. El comportamiento de un objeto define su estado.
67. Si, hay comportamientos que alteran el estado del objeto.
68. Si, el estado de un objeto es el resultado acumulado de sus comportamientos.
69. Una operación es un procedimiento que una clase ofrece a los objetos del mismo tipo.
70. Las 3 operaciones más comunes son modificar, seleccionar e iterar.
71. Las 2 operaciones que se utilizan para crear instancias son constructor y destructor.
72. El modificador se dedica a alterar el estado de un objeto.
73. El selector se dedica a acceder al estado de un objeto, sin modificarlo.
74. El iterador permite recorrer todas las partes de un objeto, sin alterarlo.
75. El constructor crea un objeto, le inicializa su estado y le asigna un espacio de memoria.
76. El destructor libera el estado de un objeto, rompe la referencia a memoria y lo destruye.
77. Los subprogramas libres son operaciones fuera de las clases.
78. Todos los métodos son operaciones.
79. Todas las operaciones no son métodos, por ejemplo, los subprogramas libres de C++.
80. En base al protocolo de un objeto, hay veces que es conveniente dividir sus métodos en pequeños grupos lógicos, para ver más claramente los roles y responsabilidades que tiene ese objeto.
81. Estos grupos lógicos se denominan papeles.
82. Las responsabilidades más grandes de un objeto son su estado y su comportamiento, ya que definen su papel en el sistema.
83. Es relevante el orden en el cual se invocan las operaciones de un objeto, ya que cada una al terminar va a modificar el estado del mismo, y cambiando como la próxima operación va a modificar el objeto.
84. Los objetos se pueden considerar como maquinas en base a si están activos, ejecutando un hilo de control, o pasivos.
85. La identidad de un objeto es aquello que lo distingue de los demás objetos, más precisamente el espacio de memoria que le fue asignado.
86. Significa que ambas apuntan a la misma dirección de memoria.
87. Asignarle y a x significa que y ahora va a tener la misma referencia a memoria que x.
88. La clonación es duplicar un objeto para que en memoria existan 2 objetos iguales en un mismo espacio de tiempo.
89. Una clonación superficial es solo una clonación de estado.
90. Una clonación profunda es una clonación de estado y del objeto es si (referencia a memoria).
91. El ciclo de vida de un objeto es el rango de tiempo que pasa desde que el objeto es asignado a una posición de memoria hasta que esa posición es liberada con la destrucción del objeto.
92. El espacio ocupado por un objeto se recupera destruyendo la referencia en memoria o llamando a un destructor.
93. Los tipos de relaciones que existen entre objetos son enlaces y agregación.
94. Un enlace es la relación en la cual un objeto usa los servicios del otro.
95. Los mensajes en un enlace entre objetos pueden ser unidireccionales o bidireccionales.
96. Un mensaje unidireccional es cuando un objeto a invoca un servicio de un objeto b sin que b invoque nada de a.
97. Un mensaje bidireccional es cuando un objeto cliente utiliza un servicio de un objeto servidor y a su vez recibe una respuesta del servidor.
98. El objeto que actúa como cliente o actor es el que inicia el paso de un mensaje en una relación bidireccional.
99. Los roles en una relación de enlace son actor, servidor y agente.
100. Un objeto actor es que puede operar en otros objetos, pero ningún objeto puede operar en él.
101. Un objeto servidor es el que no puede operar en otros objetos, pero los demás objetos pueden operar en él.
102. Un objeto agente puede operar en otros objetos y además pueden los demás objetos operar en él.
103. Visible.
104. Las 4 formas de visibilidad que puede tener un objeto servidor hacia un objeto cliente son: global, por parámetro, ser parte del cliente o local.
105. Sincronizados.
106. Las formas de sincronización son secuencial, vigilado y sincrónico.
107. Que dos objetos estén sincronizados de manera secuencial significa que la semántica del objeto pasivo está garantizada solo en presencia de un objeto un único activo simultáneamente.
108. La sincronización vigilada se refiere a que el objeto pasivo está garantizado por múltiples hilos de control, pero los clientes activos deben colaborar para lograr la exclusión mutua.
109. La sincronización sincrónica se refiere a que la semántica del objeto pasivo está garantizada por la presencia de múltiples hilos de control, y el servidor garantiza la exclusión mutua.
110. El enlace es una relación de igual a igual.
111. La agregación es una relación basada en jerarquía.
112. La agregación denota una jerarquía de todo/parte, permitiendo ir de la parte al todo y del todo hacia sus partes.
113. Al todo también se lo denomina el agregado.
114. Las partes forman parte del estado del todo.
115. Las agregaciones pueden ser físicas o conceptuales.
116. La agregación con contención física permite la navegación entre objetos.
117. Una clase es un conjunto de objetos que comparten una estructura y un comportamiento en común.
118. No, la interfaz de una clase proporciona una visión externa de la misma y profundiza la abstracción.
119. No, la implementación de una clase muestra su visión interna, que contiene los secretos de sus comportamientos.
120. Una interfaz en base a la visibilidad y accesibilidad se puede dividir en pública, privada o protegida.
121. Las relaciones básicas entre clases son de generalización (es un), de todo/parte (parte de) y de asociación que denota una dependencia semántica.
122. Las relaciones que se desprenden de las relaciones básicas entre clases que pueden existir son: asociación, herencia, agregación, uso, instanciación y metaclase.
123. La asociación de denota una dependencia semántica pero no denota una dirección en la asociación.
124. La cardinalidad en una asociación define a cuantas instancias de a le corresponden tantas instancias de b.
125. La cardinalidad puede ser uno a uno, uno a muchos y muchos a muchos.
126. La herencia es una relación entre clases en la cual una clase padre comparte ciertos atributos o métodos con sus clases hijos.
127. Hay 2 tipos de herencia, la simple y la múltiple.
128. La herencia simple es cuando una clase hereda características y métodos de otra clase.
129. La herencia múltiple es cuando una clase hereda características y métodos de varias clases.
130. Se denomina a esa clase abstracta o superclase.
131. Se denomina a esa clase subclase.
132. Al método de una clase abstracta que no posee implementación y va a ser sobrescrito por las subclases se lo denomina virtual.
133. La clase más generalizada de una estructura de clases se denomina clase base.
134. El polimorfismo es la sobre escritura de los métodos heredados de una clase padre en una clase hija.
135. Sobrecarga.
136. Con el polimorfismo correctamente no son necesarias las sentencias case.
137. La agregación en la relación entre clases es que una clase es parte de otra clase, siendo una el todo y la otra la parte.
138. En la agregación entre clases hay 2 tipos de contención física. La contención física por valor en la cual el objeto A no existe de manera independiente de la instancia de B que lo encapsula. La contención física por referencia se define mediante la utilización de un puntero, y de esta manera A es independiente de B, porque en el caso de destruirse B, se destruye el puntero de A y no A en sí.
139. La contención física por valor en la cual el objeto A no existe de manera independiente de la instancia de B que lo encapsula.
140. La contención física por referencia se define mediante la utilización de un puntero, y de esta manera A es independiente de B, porque en el caso de destruirse B, se destruye el puntero de A y no A en sí.
141. Una relación de uso es un refinamiento de la asociación, donde se define que abstracción es el cliente y que abstracción es el servidor y que servicios provee.
142. La instanciación es la creación de un objeto a través de una clase, al cual se le asigna un espacio de memoria.
143. Todo objeto es una instancia de una clase.
144. Una metaclase es un tipo de clase cuyas instancias son la clase en sí.
145. Las métricas para medir la calidad de una abstracción son el acoplamiento, la cohesión, la suficiencia, la compleción o plenitud y el ser primitivo.
146. El acoplamiento es grado de dependencia que hay en las relaciones entre módulos.
147. La cohesión es el grado de conectividad entre los elementos internos de un módulo, buscando siempre una relación lógica entre los mismos.
148. La suficiencia se refiere a que un módulo tiene suficientes características de la abstracción para permitir una interacción significativa y eficiente.
149. La compleción o plenitud es cuando un módulo es lo suficientemente completa para ser usada por cualquier cliente.
150. Ser primitivo se refiere a la presencia de operaciones que solo pueden ser realizadas en el ámbito de la clase.
151. A la hora de aplicar un método a una abstracción se debe tener en cuenta la reutilización de ese método, que grado de complejidad trae el mismo, que tan aplicable es y que conocimiento se tiene de la implementación.
152. Un mensaje entre entidades se puede pasar de forma sincrónica, de abandono inmediato, de intervalo o asincrónica.
153. Un mensaje sincrónico es en el cual el emisor inicia la acción y el receptor está listo para recibir el mensaje, y ambos esperan hasta que la otra parte esté lista para poder continuar.
154. Un mensaje de abandono inmediato es igual que el sincrónico pero el emisor abandona la operación si el receptor no está preparado de manera inmediata.
155. Un mensaje de intervalo es de la misma manera que el sincrónico pero el emisor espera al receptor por un periodo especificado.
156. Un mensaje asincrónico es cuando el emisor inicia la operación sin importar si el receptor está listo o no.
157. Que una abstracción sea visible para otra significa que una debe poder verla y acceder a los métodos públicos de la otra.
158. La ley de Deméter dice que los métodos de una clase no deberían depender de ninguna manera de la estructura de ninguna clase, salvo de la estructura inmediata (padre) de su propia clase. Y cada método debe enviar mensajes solo a objetos pertenecientes a un reducido grupo de clases.
159. La aplicación de esta ley genera clases con un bajo nivel de acoplamiento, altamente encapsuladas.
160. Las 4 formas en las cuales un objeto X puede hacerse visible a un objeto Y son: X es global a Y, X es un parámetro de una operación de Y, X es parte de Y o X es declarado localmente.
161. Clasificar objetos sirve para identificar jerarquías de generalización, especialización y agregación entre clases. Reconocer las similitudes de la comunicación entre objetos. Tomar decisiones de diseño.
162. La dificultad de la clasificación reside en determinar la frontera de cada objeto, que no es tarea sencilla.
163. La tarea del observador es descubrir el orden que rige a las cosas.
164. Las aproximaciones generales de la clasificación son la categorización clásica, el agrupamiento conceptual y la teoría de prototipos.
165. La categorización clásica es la división de los objetos en conjuntos dependiendo de la presencia o ausencia de una propiedad particular.
166. La agrupación conceptual se basa en la descripción de las entidades y su clasificación en base a esas descripciones.
167. La teoría de prototipos se basa en representar un clase de objetos a través de un prototipo, y se considera un objeto como un miembro de esta clase si y solo si se parece a este prototipo de forma significativa.
168. Una abstracción clave es una clase u objeto que forma parte del vocabulario del dominio del problema.
169. Un mecanismo es una decisión de diseño sobre como cooperan colecciones de objetos. Los mecanismos representan así patrones de comportamiento.

# Unidad 2

1. Un framework es una abstracción que provee ciertas funcionalidades generales que asisten en el desarrollo de aplicaciones. Provee una forma estandarizada para el desarrollo de aplicaciones. Es un ambiente de software universal y reutilizable, que contiene diversos componentes para asistir en el desarrollo de proyectos.
2. Los Frozen spots es un framework son los elementos que definen su arquitectura general y que permanecen inalterados durante cualquier instanciación del software.
3. Los Hot Spots en un Framework son los ámbitos en los cuales los programadores ingresan su código para darle la funcionalidad pretendida al proyecto.
4. Un framework en base a su extensibilidad se puede clasificar en extensible o no, si deja o no a los usuarios ampliar sus capacidades.
5. Un framework de caja blanca requiere que los usuarios tengan conocimiento de la estructura y del código interno del mismo, vienen con el código fuente y su comportamiento se extiende por medio del uso de subclases y herencia.
6. Un framework de caja negra no requiere conocer el funcionamiento interno del mismo y su comportamiento se extiende composición y delegación de comportamiento entre objetos.
7. Las ventajas que ofrece un framework son que facilita y agiliza el desarrollo de aplicaciones, permite reutilizar código existente, promueve buenas prácticas de desarrollo y permiten generar un código mas claro.
8. El framework .net ofrece capacidades Full-OOP, incluyendo herencia, Windows Forms, soporte nativo de XML, gestión de errores estructurada un modelo de objetos para acceso a datos más potente con ADO.NET; posibilidad de crear aplicaciones de consola (ventana MS-DOS); programación para Internet mediante Web Forms; un entorno de desarrollo común a todas las herramientas de .NET, etc.
9. El COMMON LANGUAGE RUNTIME es el encargado de ejecutar el código de las aplicaciones, administrar el código en tiempo de ejecución, realiza gestión a bajo nivel, proporciona un sistema común de tipos, gestiona la seguridad del código y ofrece un soporte multilenguaje.
10. El MIL es un código intermedio que se genera a la hora de compilar el código fuente y consiste en un conjunto de instrucciones independientes del sistema operativo que ocupa de la manipulación de objetos, accesos a memoria, manejo de excepciones, etc.
11. El CTS es el Common Type System que define el modo en cual van a ser creados y manipulados los tipos dentro del framework.
12. El CLS o Common Language Specification es un conjunto de características comunes que deben tener los lenguajes en la plataforma para poder integrarse entre sí.

# Unidad 3

1. Los campos son las variables internas de una clase que determinan su estructura. Son características cualitativas y cuantitativas.
2. Los métodos de una clase son las funciones o procedimientos públicos que figuran en la interfaz.
3. Se llama sobrecarga a cuando se crean métodos en una clase que tienen el mismo nombre, pero se diferencian por los parámetros que reciben.
4. Las propiedades son los campos que determinan las características de una clase. Sin embargo, en el libro se toma a las propiedades como métodos públicos por los cuales se accede a los campos de la clase. Getters y Setters.
5. Los 5 tipos de propiedades son: escritura y lectura, lectura, escritura, con parámetros y por defecto.
6. Los campos, métodos y propiedades de las clases pueden tener ámbito público, privado, protegido o amigo.
7. El ámbito público determina que todo lo que sea público puede ser visto en todo el ámbito del proyecto. Todo lo que sea protegido solo puede ser accedido solo dentro del ámbito de la clase o por una subclase de la misma. Amigo o Friend permite que el elemento sea visto por otro elemento al que se lo declara amigo. En el caso de que sean dos clases amigas o Friends ambas clases podrán ver los elementos privados y protegidos del otro. El ámbito privado solo puede ser accedido por dentro de la clase.
8. Los constructores se utilizan para instanciar una clase y crear un objeto, asignándole un espacio de memoria.
9. El tiempo de vida de un objeto es tiempo que transcurre desde que el objeto es instanciado hasta que es des referenciado o destruido.
10. No, no se usa un contador de referencias, en .net se usa un garbage collector.
11. El garbage collector es el objeto encargado de liberar memoria.
12. La zona donde se guardan todas las instancias de los objetos es el HEAP.
13. Los 2 métodos son dispose y finalize.
14. El método finalize se hereda de System Object.
15. La firma de Finalize es : Protected Overridable Sub Dispose(disposing as boolean).
16. El método dispose se usa para eliminar objetos no utilizados.
17. Se utiliza disposing.
18. Cuando se ejecuta el método collect del Garbage Collector se ejecuta el método Destructor en los objetos afectados.
19. Una clase bien diseñada que no tenga destructor debería implementar el método Dispose.
20. Para usar dispose se tiene que implementar la clase IDisposable
21. Dentro del método dispose se genera un algoritmo para indicarle al garbage collector que el espacio de memoria ocupado por el objeto puede ser liberado.
22. El uso de dispose y finalize se puede combinar utilizando el finalize como garantía de que se liberen los recursos en caso de que el dispose falle.
23. Se denomina “resurrección de objetos” a lo que pasa cuando un objeto que está en cola para ser eliminado por el GC se referencia a sí mismo a través de otro objeto que esta fuera de la cola, evitando así ser eliminado.
24. Generación se denomina al tiempo de vida que tiene un objeto.
25. Los valores que puede tener la generación de un objeto van de 0 a 2, para determinar cuánto tiempo de vida tiene un objeto.
26. Para saber cuántas veces se hizo la recolección para una generación especifica basta con mirar el numero de la generación, porque cada vez que pasa el GC aumenta en 1 la generación de los objetos sobrevivientes.
27. El número de generación de un objeto se obtiene con el método getGeneration(objeto)
28. La cantidad de bytes asignados se obtienen con GC. GetTotalMemory(boolean).
29. Se utiliza el método GC. KeepAlive(Object).
30. Se utiliza GC.SuppressFinalize(Object).
31. Se utiliza GC.ReRegisterForFinalize (Object)
32. Para obtener el máximo número de generaciones se utiliza la propiedad GC. MaxGenerations.
33. Los sucesos o eventos son las reacciones de los objetos a estímulos externos o del sistema.
34. Para declarar un evento se debe declararlo en la clase, definir que tiene que pasarle a un objeto para que se desencadene y asociar ese evento a un procedimiento.
35. Para que ocurra un evento tiene que ocurrir el suceso que se definió cuando se lo declaro.
36. Los eventos se atrapan con el event handler.
37. El método addhandler sirve para conectar un evento con un handler.
38. En una clase se pueden compartir métodos, propiedades y eventos, utilizando la palabra Shared, que indica que para todas las instancias de una clase el elemento compartido va a ser el mismo y se va crear uno solo al cual van a acceder todas las instancias.
39. Los campos compartidos serán comunes para todas las instancias de la clase, y tendrán todo el mismo valor. Y si se cambia ese valor se altera en todas las instancia de la clase.
40. Los métodos compartidos no están asociados con una instancia especifica de una clase. Pueden ser llamados desde la clase o de una instancia.
41. Los eventos compartidos pueden ser invocados por métodos compartidos o no. Solo pueden usar addhandler para agregar un controlador.
42. Para atrapar eventos de una matriz se tendría que realizar una iteración por la misma agregando handlers que agarren los eventos que puedan tirar los objetos contenidos.
43. Los miembros compartidos son las funciones, propiedades y campos de una clase que son compartidos por todas las instancias de la misma, pero que no son específicos de que ninguna.
44. Las clases anidadas son clases contenidas dentro de otra clase. Sirven para hacer los paquetes de clases más compactos, aumentar la encapsulación, mejorar la legibilidad del código.
45. Los ámbitos que existen son public, private, friend y protected.
46. El ámbito público es todo aquello que sean visible en todo el proyecto.
47. El ámbito privado es todo aquello que es visible solo desde la clase.
48. El ámbito friend determina que 2 clases que son amigas pueden ver sus elementos privados y protegidos entre sí.
49. El ámbito protegido determina que los elementos con este modificador solo podrán ser vistos dentro de la clase o por subclases de la misma.
50. El ámbito protected friend dice que los elementos podrán ser accedidos por subclases o por clases amigas.
51. Se pueden heredar las cosas con ámbito público, protegido y protected friend.
52. El polimorfismo se utiliza para evitar escribir código innecesario, pudiendo determinar métodos que van a ser heredados y luego implementados en las clases que lo heredan, reutilizando el mismo.
53. la clase derivada se utiliza para heredar características de una clase base, que tienen una relación lógica con la clase derivada. Para realizar esto se debe usa la palabra inherits.
54. La clase Me hace referencia a la instancia especifica de la clase donde se ejecuta el código.
55. MyBase representa a la clase base.
56. MyBase se usa para acceder a los métodos de una clase base, o para llamar al constructor de la clase base desde el constructor de la clase derivada.
57. My class hace referencia a la instancia actual de una clase X y como se implementó originalmente. Todas las llamadas a métodos que usen MyClass deben llamar métodos NotOverridable.
58. La diferencia entre MyBase y MyClass es que MyBase hace referencia a la clase base de la instancia actual de una clase, mientras que MyClass hace referencia a la instancia actual de la clase.
59. Se puede usar una clase abstracta con el fin de que todas las clases que hereden de la misma tengan propiedades y métodos en común, e, implementen dichos métodos.
60. Una clase que se declara sellada es aquella clase que no se la puede heredar.
61. La sobre escritura es la acción de sobre escribir los métodos heredados por una clase hija de una clase padre, para su implementación.
62. Los elementos que se pueden sobre escribir son los que tengan el modificador Overridable, mustOverride.
63. El sombreado de métodos se denomina cuando 2 métodos tienen el mismo nombre como pasa en la herencia, y una esconde al otro, quedando el escondido no accesible. Se usa el que el esconde y no el escondido. Sirve para proteger las definiciones de la clase, por si se producen cambios en la clase base.
64. La sobre escritura a diferencia del shadowing sobre escribe el método de la clase base, mientras que el shadowing solo lo des referencia.

# Unidad 4

1. Una excepción es un error en tiempo de ejecución.
2. En el bloque catch se coloca todo el código necesario para controlar cualquier excepción que pueda surgir en el try.
3. Para construir una excepción se la debe nombrar con el prefijo Exception, debe heredar de System.Exception y debe ser serializable implementando ISerializable.
4. Si no se trata la excepción con el correspondiente catch la misma ira escalando la jerarquía de llamadas hasta detener el programa de manera abrupta.
5. El objeto de más jerarquía para el manejo de excepciones es System.Exception.
6. La clase Exception está en el namespace System.
7. Las clases genéricas más importantes aparte de Exception son System.Object y System.Collections
8. La instrucción Catch es la usada para interceptar las instrucciones.
9. Se coloca en el Finally.
10. Se usa la excepción   System.DivideByZeroException
11. Se usa ApplicationException.
12. Colocaría una sentencia If.
13. Se debe colocar en el Catch.
14. Para provocar un error se utiliza la instrucción throw.
15. Throw New System.ArgumentException().
16. Lo construiría haciendo que herede de la clase Exception, le crearía los respectivos constructores y haría que implemente ISerializable.
17. Catch Ex As MyException

Console.WriteLine("Custom Exception thrown " & Ex.Message)

# Unidad 5

1. Las interfaces se utilizan para que las clases que las implementan puedan hacer uso de sus métodos y propiedades, siendo una clara expresión del polimorfismo. Este conjunto de funcionalidades será dado estableciendo un contrato entre la interfaz y la clase que implementa sus elementos.
2. Una interfaz se implementa usando la palabra Implements.
3. No, las interfaces no se heredan.
4. Si, va a suceder que muchas clases van a implementar a una interfaz y cada una va a aplicar polimorfismo de la manera que más le convenga sobre los métodos de la interfaz.
5. La interfaz iComparable se utiliza para realizar ordenamientos en arrays.
6. La interfaz IComparer se utiliza para el ordenamiento de arrays recibiendo 2 objetos como parámetros, teniendo en cuenta varios criterios de ordenamiento. A diferencia de IComparable que no permite más de un criterio de ordenamiento.
7. La interfaz IClonable se utiliza para copiar un objeto a otra posición de memoria.
8. La interfaz IEnumerable se utiliza para hacer una iteración simple en una colección no genérica.
9. La interfaz IEnumerator es la interfaz que se encarga de darle el numerador a IEnumerable.
10. Un delegado es un objeto al cual los demás objetos ceden la habilidad de ejecutar su código.
11. Se puede delegar a las variables declaradas como Delegates.
12. Se pueden delegar métodos.
13. Construir un delegado Public Delegate Sub X()
14. Declararía un delegado que maneje la ejecución de una rutina Devolver Llamada que solo la active si se encuentra una llamada perdida.
15. La multidifusión de delegados sirve para crear un delegado que a su vez invoque a otros delegados que estos invocan a su vez a funciones. Pudiendo con un solo delegado invocar mucha cantidad de funciones.
16. Los atributos son palabras claves que permiten definir información sobre las entidades del programa, ya sean métodos entidades, etc.
17. Sintaxis de un atributo: [ modificador del atributo] nombre atributo [ ( argumentos| inicializador) ]
18. El struct layout se utiliza para controlar la distribución física de los datos de una clase o estructura en memoria.
19. El FieldOffset indica la posición en memoria de los campos sin manejar afectados por el struct layout.
20. El ConditionalAttribute le dice al compilador que un método o atributo debe ser ignorado hasta que se defina un símbolo de compilación condicional.
21. El atributo Obsolete marca lo elementos de una clase que no están en uso.
22. El DebuggerStepThroughAttribute se usa para decir al debugger que pase a través del código en vez de por adentro del código.

# Unidad 6

1. El esquema cliente servidor se basa en 2 entidades. El servidor que está dedicado a proveer recursos y servicios, y el cliente, que se dedica a solicitar los recursos o servicios que provee el servidor, el cual le responde. El cliente es quien tiene el papel activo, porque hace las peticiones, es orientado a usuarios, puede conectarse a múltiples servidores y recibe las respuestas del servidor. El servidor siempre espera las solicitudes de los clientes, las cuales procesan, y en base a ellas elaboran la respuesta. El servidor puede servir a múltiples clientes.
2. Pasar información batch es pasar la información en lote, de manera rutinaria sin interacción humana.
3. Pasar la información online es un pase de información en el cual hay intervención humana de por medio.
4. Un protocolo es un sistema de reglas que 2 entidades deben cumplir para poder comunicarse y transmitir información. Determinan el estándar de semántica, sintaxis y sincronización de la comunicación.
5. Una red de área local utiliza el protocolo TCP.
6. Internet utiliza el protocolo TCP/IP.
7. El protocolo IP es una dirección numérica que identifica de forma lógica y jerárquica a una computadora dentro de una red.
8. La distribución de procesos se basa en que varias computadoras se conectan entre sí por una red y comparten capacidad de procesamiento. Las ventajas que tiene la distribución de procesos son un menor costo, más confiable, más flexible y un menor tiempo de procesamiento.
9. Distribuir almacenamiento ofrece las ventajas de un menor costo al no tener que concentrar todos los datos en un solo sistema, una mayor seguridad ya que los datos no están concentrados en un solo lugar, además de poder distribuir los backups, y un menor costo al utilizar varios sistemas chicos trabajando en conjunto antes que un solo sistema grande que se encargue de todo.
10. Un socket es un conjunto de elementos (protocolo TCP/IP y puertos locales y remoto) que constituyen el mecanismo de entrega de paquetes.
11. Un socket sincrónico tiene la característica de que cuando se invoca un método o servicio, el cliente queda esperando hasta que se devuelva el resultado.
12. En un socket asincrónico se realiza la invocación y se continua con la ejecución del programa, pero se crea un callback al momento de invocación, que recibirá la respuesta y tomara una acción en base a ella.
13. Se puede utilizar el objeto System.Windows. Forms.WebBrowser para crear un navegador dentro de la aplicación.
14. Los puertos de una Pc son interfaces a través de las cuales se envían y reciben datos.
15. La PC posee puertos paralelos, USB y en serie.
16. Los puertos paralelos actúan como interfaz entre el pc y un periférico. Los bits de datos viajan en paquetes de un byte.
17. El puerto en serie actúa también como interfaz, con la diferencia que los bits pasan uno atrás del otro en serie.
18. Los puertos USB funcionan en base a pipes o tubos que son la conexión entre el controlador del USB y el dispositivo conectado. Hay 2 tipos de pipes, stream y message. Stream se usa para transferencias bidireccionales de control y las message para comandos cortos y simples al dispositivo. Hay 3 tipos de transferencias isócrona, por interrupción y por lotes. La isócrona se realiza a un ritmo alto, pero con posibilidad de pérdida de datos. La transferencia por interrupción se realiza para dispositivos que necesitan respuestas rápidas. Y la transmisión por lote son transferencias esporádicas que utilizan todo el ancho de banda.
19. La domótica es la practica basada en la utilización de sistemas informáticos para la automatización de edificios o casas.