

ComboBox	DropDownList → para que no se pueda escribir en el comboBox comboBox1.Items.Add(""); → agregar cosas a la lista del comboBox
----------	---

Microsoft quería que c# emule java. Se pueden crear aplicaciones, juegos, front end, back end, multiplataformas, ia.

Plataforma de desarrollo: da las herramientas para construir una app.

C# es un lenguaje de alto nivel: se escribe de una manera similar a como nos comunicamos. Visual basic: lenguaje que se asemeja a la manera de comunicarse: es mas claro y facil de entender
fuertemente tipado: no permite violaciones entre tipos de datos en int no puedo guardar char
se pueden hacer casteos
int numero = 10;
string numeroTexto = numero.ToString();

Orientado a objetos: se puede modelar eficientemente una situación del mundo real
Multiplataforma: se puede crear apps de .net para cualquier plataforma.

COMPONENTES DE .NET:

CLR COMMON LANGUAGE RUNTIME entorno de ejecución, es quien ejecuta la app.
Asigna y administra la memoria. Es una máquina virtual que genera y compila código, se compila a un lenguaje intermedio y el clr interpreta el código y lo pasa a binario.

BCL: conjunto de clases que exponen funcionalidades (se pueden usar para cualquier tipo de aplicación) y que pueden utilizarse en cualquier app. Librería de .net formada por muchos datos que permite acceder a datos de clr. Clases prefabricadas de donde se puede modificar y crear nuevas clases, creando más funcionalidades. Clase base donde están divididas. el BCL define los tipos de datos por ejemplo, BCL define que Int32 puede escribirse en C# como int o int16 como short

DLL: se generan a través de la compilación, CLR es quien interpreta los datos compilados.

CTS define los tipos por valor y tipos por referencia.

TIPOS DE DATOS

escalares son cuando un dato es atomico y unidimensional, tiene un unico valor, int por ejemplo

no escalares pueden tener mas de un valor como por ej array

namespace: agrupa clases

datos especiales en .net

object tipo de dato "padre" alias de system object, puede asignarse en esta variable cualquier valor

```
object objeto;  
objeto = 90;  
objeto = "lala";
```

Se pueden hacer cambios en el tipo de dato. no es una buena práctica
todos los tipos de datos heredan los métodos y atributos de object. de manera indirecta o indirecta (System.Object)

dynamic

inferencias de tipos

se usa var: tiene que asignarse un dato. e infiere el tipo
var numero = 22; asume que es un int
var numeroTexto = "22" asume que es un string

se determina el tipo al inferirse. Ejemplo: si se infiere que es un int, despues no se puede cambiar a string

Convenciones de microsoft para c#

<https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/coding-style/coding-conventions>

Conversiones

implícitas: no interviene el programador, no requiere casteo porque no hay perdida de datos
float numero = 10; se va almacenar como un int, no hay perdida

explicitas: puede haber pérdidas de datos, interviene el programador
float numero = 10.5F;
int numeroEntero = (int)numero; el programador asume la perdida de datos

TryParse: intenta convertir un dato, devuelve un booleano. Si puede convertirlo escribe true sino, false

ENTRY POINT: lo que se ejecuta primero

static void Main

static es un modificador que permite ejecutar sin realizar un object,

void: no necesita un return

No puede haber dos puntos de entrada porque no va a saber que ejecutar primero. Se puede designar cual ejecutar primero desde la consola.

CONSOLE

clase pública y estática, se puede acceder.

Propiedades

background cambia el color del fondo Console.BackgroundColor = ConsoleColor.Red

title: cambia el nombre de la app

Con los {} mostramos variables en la consola
Console.WriteLine()

CLASE 2

PRINCIPIO DRY

Don't repeat yourself. Funcionalidades deben ser únicas, no debe existir otra representación de la misma funcionalidad, tiene una única interpretación, se debe confiar en que es correcta. Cuando hay un cambio no se debería hacer modificaciones en todo el código

PRINCIPIO KISS

Keep it simple stupid

PARAMS

Sirve para cuando no sabemos cuantos parametros usamos.

ValidarTexto("escriba el nombre", nombreMascota, "pepe")

ValidarTexto("escriba el nombre", nombreMascota, "pepe", "mateo")

ValidarTexto("escriba el nombre", nombreMascota, "pepe", "mateo", "Juan")

```
void ValidarTexto(string mensaje, out string valor, params string valoresPermitidos)
{
    Console.WriteLine(mensaje);
    valor = Console.ReadLine();

    while(!valoresPermitidos.Contains(valor))
    {
        Console.WriteLine("Error ingrese");
    }
}
```

clase es como un molde donde se generan objetos y los objetos generan estaticas

internal: otras clases en el mismo proyecto pueden acceder

private: solo la clase donde esta escrita puede acceder

namespace: organiza el código

El string es inmodificable, se utiliza nuevo espacio de memoria para cada modificación

Se usa StringBuilder cuando se necesita modificar muchas veces el string para no utilizar tanta memoria

STATIC Es un modificador que permite ejecutar un método sin tener que instanciar una variable (sin crear un objeto)

Verdadero ✓

89%

Una de las funciones del Runtime es:

Compilar código escrito en lenguaje C#.

14%

Ejecutar las aplicaciones al interpretar el lenguaje intermedio al que se compilan los lenguajes de .NET y traducirlo a lenguaje nativo / máquina ✓

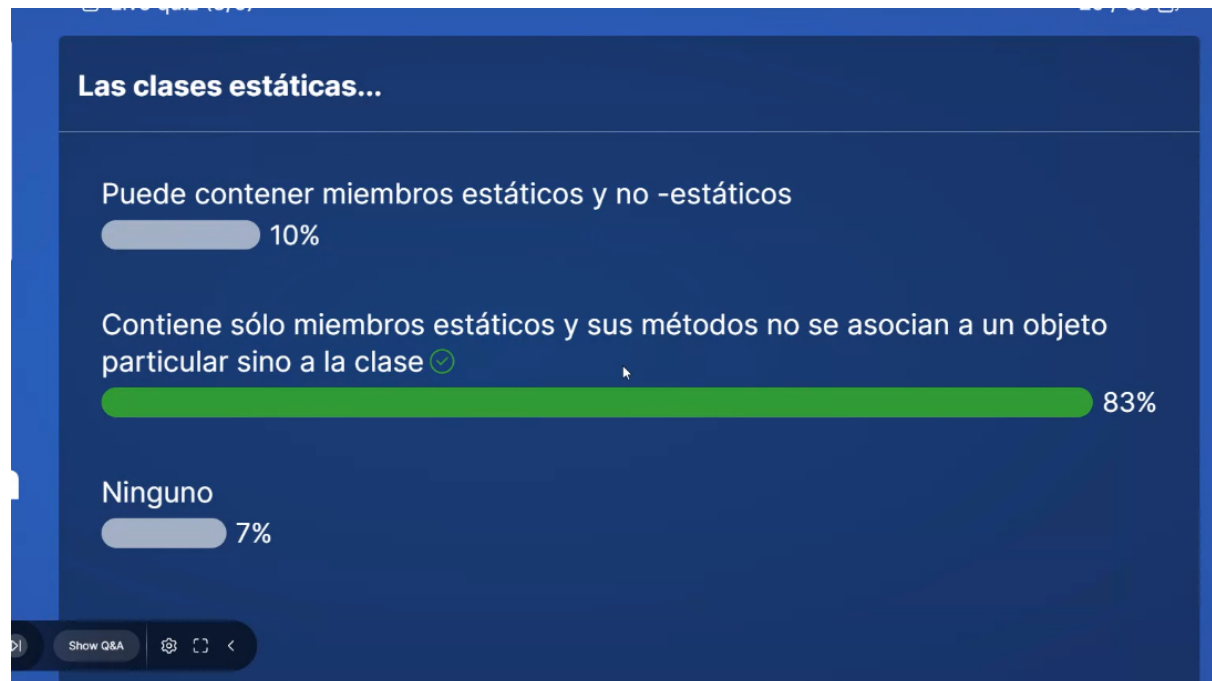
75%

Ninguna respuesta.

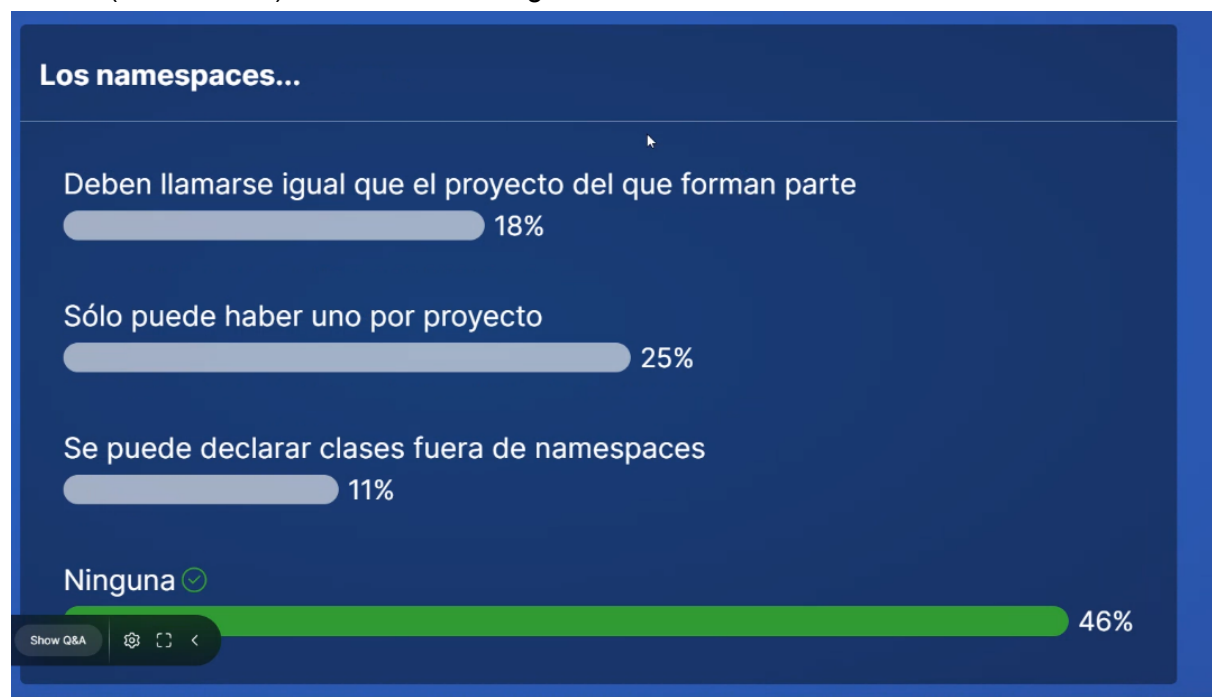
11%

VARIABLES Y ATRIBUTOS> sustantivos y lower camel case (Camel Case)
MÉTODOS: verbos y Upper Camel Case (Pascal Case)

Clases estáticas: conjunto de métodos. Por ejemplo, cuando hay una lista de métodos utilizada para el programa, validar número, validar string, validar una secuencia.



si no es estatico es de instancia. una clase de instancia si puede tener estático y no estatico(de instancia). Uno de instancia genera un vcalor cuando lo inicializo.



puede haber muchos namespaces, agrupacion de las clases, por lo tanto no se pueden declarar por fuera.

CLASE 3

Refactorizar: mejora y optimiza los procesos.

Paradigma de programación estilo de desarrollo de programa, modelo para resolver problemas computacionales. Los LenguajesProg se encuadran en varios paradigmas. Define la forma metodológica con el que se va a resolver un problema.

Lenguaje imperativo: agrupa instrucciones como parte del estado en el que operan

NEW: reserva memoria

Lenguaje declarativo: programador declara el resultado pero no comp
logico, matematico, reactivo

POO: manera de construir software que propone resolver problemas de la realidad identificando atributos comunes, unificando sus comportamientos y encontrando las relaciones entre ellos

PILARES:

- encapsulación: caracteriza en la que denota la capacidad del objeto de responder a peticiones o métodos sin exponer la forma en la que logra llegar a esos resultados
- abstracción: seleccionar datos de un conjunto más grande para mostrar solo los detalles relevantes del objeto. Ignorancia selectiva: enfocarse en lo que realmente importa para la construcción de la clase. "Habilidad de abordar un concepto mientras se ignoran algunos de sus detalles". Ejemplo: mapas políticos y geográficos
- herencia: se comparte el comportamiento de la clase sin tener que volver a programarlo. Se comparten atributos base
- polimorfismo: define la capacidad de que más de un objeto puedan crearse usando la misma clase base para lograr dos conceptos de objetos diferentes

CLASE

Declaraciones de objetos (abstracciones de objetos) (cuando definimos un objeto definimos una clase), una descripción de un conjunto de objetos que comparten los mismos atributos, métodos, relaciones y semántica en un determinado contexto

atributo estático: características que tienen TODOS los objetos de esa clase

Método de instancia: muestra información de la instancia

los objetos se pasan por tipos de referencia

this es de la clase, solo se puede usar en método de instancia, no en statics

Composición de una Clase

Atributos

Representan **características** que son compartidas por todos los objetos de una clase.

Definen el rango de valores que puede tomar cada una de las propiedades de un objeto.

Utilizar notación *lowerCamelCase* y sustantivos.

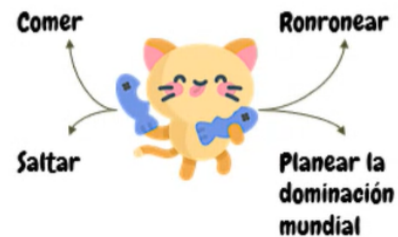


Métodos

Un método es la **implementación de una operación**.

Una operación es **una abstracción de algo que puede hacer un objeto** y que es compartido por todos los objetos de esa clase.

Utilizar notación *UpperCamelCase* y verbos.



objetos = instancia de una clase. viven en memoria se crean a partir de las clases, se pasan por referencia, se almacenan en el heap. Existen en tiempo de ejecución.

Tienen identidad, permite diferenciarlos de otros

los objetos de la misma clase tienen las mismas propiedades pero almacenan valores diferentes.

Se comunican, pueden enviarse mensajes entre otros objetos.

comportamiento: acciones que pueden realizar al recibir mensajes(métodos?) de otros

Dstrucción de un objeto

Variable local, deterministas: se crea cuando se declara y se destruye al final del ámbito(funciona solo entre los corchetes, no existe en otros lados)

Objetos: el tiempo de vida no está vinculado a su ámbito. Dstrucción no determinista. Se destruye a través de la recolección de basura ("Garbage collector" a través del CLR→ convierte de nuevo en memoria no utilizada)

CICLO DE VIDA

El operador new reserva memoria para el objeto sin inicializar. Constructor inicializa el estado del objeto

Estado de un objeto: valores que toma sus atributos en un determinado momento

constructores es un método especial cuya función es darle un valor inicial a los atributos de un objeto.

Constructor estatico se ejecuta una sola vez, los de instancia se ejecutan en cada objeto nuevo

Clase estatica: atributos de una
VS\
Clases de instancia

Un detalle, no se incluyen las validaciones en el constructor, sino que deberian validar de antemano y, si los valores que tienen son validos, recien ahi se llama al constructor

Lucas Ferrini

20:36

Por ponerte un ejemplo en el que podria tener un atributo estatico (que podrias llegar a usar aca en la materia) es un ultimold o ultimoLegajo en una clase que necesites instanciar para un ABM

Matias Mansilla

20:37

Gracias Lucas

Lucas Ferrini

20:38

Ese es un atributo que no esta asociada a una instancia en particular, sino a la clase. Me interesa saber cual fue el ultimo ID que use para crear, por ejemplo, un producto, asi se que no tengo que repetirlo. Eso de ahi es solo un ejemplo, hay mejores formas de hacerlo, pero capaz te ayuda a cerrar el concepto de que podrias llegar a hacer

Los Constructores:

Si no se declaró de forma explícita un constructor, existirá un constructor por defecto sin parámetros

☐ 6%

Los constructores de instancia, inicializan los atributos del objeto

☐ 6%

Estáticos no pueden ser invocados

☐ 0%

Estáticos son de la clase, no de una instancia específica

☐ 0%

Todas son correctas ✓

CLASE 4

Sobrecarga: técnica para mejorar la visibilidad de nuestro código. Genera miembros: métodos, constructores, operadores e indexadores. Métodos(por ej) con el mismo nombre pero que hagan tareas ligeramente distintas. por ejemplo Sumar(int int) o Sumar(int float). Mismo nombres, ambos suman algo ligeramente distinto

Los constructores estáticos no se pueden sobrecargar porque no reciben parámetros

SOBRECARGA DE MÉTODOS:

pueden cambiar la cantidad de parámetros, el tipo de datos de parámetros, o el orden de parámetros.

Las firmas de los métodos deben ser únicas dentro de una clase.

Que Forman la definición de la firma de un método?

Nombre del método.

Tipo de parámetros.

Cantidad de parámetros.

Modificador de parámetro (out o ref)

No afectan la firma de un método:

Nombres de parámetros.

Tipo de retorno del método.

Los métodos deben llamarse igual y hacer lo mismo.

No se puede sobrecargar cambiando solo el retorno. Hay que cambiar cantidad, orden o tipo de parámetros DE ENTRADA. No importa el de salida

SOBRECARGA DE CONSTRUCTORES.

Sobrecarga de constructores, cuando se usa?:

Se suele hacer cuando se quiere dar la posibilidad de instanciar objetos de formas diferentes.

Inicializar objetos en distintos estados.-

: this() se usa para llamar a otros constructores y en el nuevo constructor agregar solo lo que es realmente necesario. se resuelve por el número y orden de los tipos de datos, no por los nombres de los parámetros. Solo se pueden llamar a operadores de la clase. no se puede usar para otra clase.

constructor privado no va a permitir crear una instancia por fuera de la clase. se pueden sobrecargar aunque sean privados.

Los constructores estáticos no se pueden sobrecargar porque no reciben parámetros

SOBRECARGA DE OPERADORES

Sobrecargar un operador consiste en modificar su comportamiento cuando se utiliza con una determinada clase.

Nota: Los operadores de comparación, si son sobrecargados, se deben sobrecargar en pares; es decir, si se sobrecarga el operador ==, se deberá sobrecargar el operador !=

Por defecto el == compara por referencia. Cuando se realiza la sobrecarga se descarta el == básico y se utiliza el programado

Operadores de casteo (con sobrecarga?)

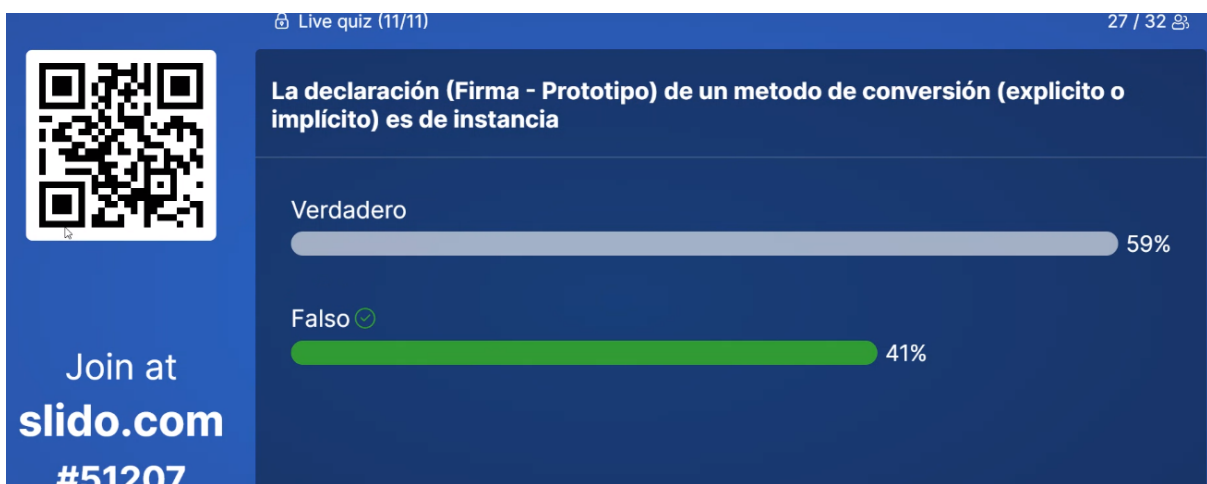
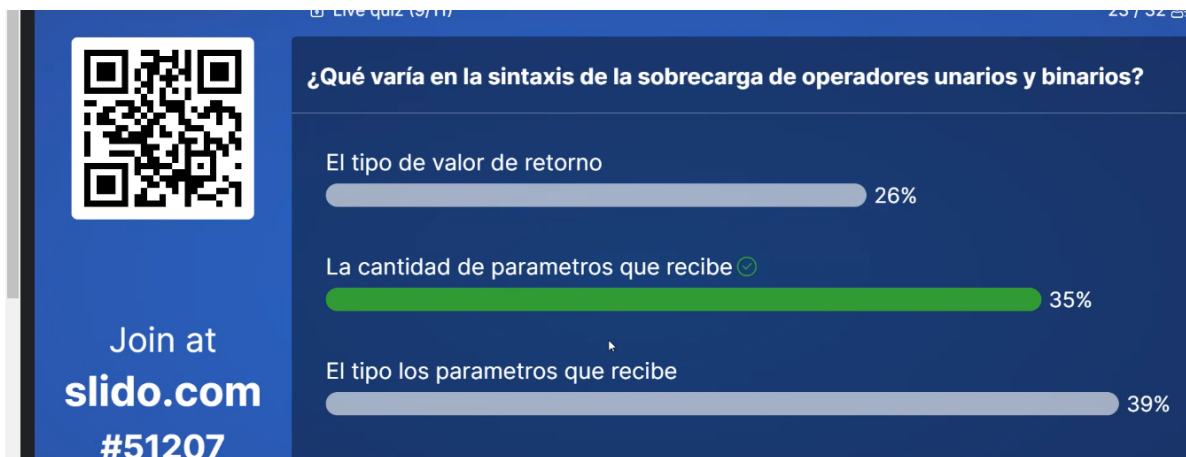
🔴 Ejemplo en sistema real del patrón de diseño Singleton | MVC .Net

```

8      {
9      15 referencias
10     public class Persona
11     {
12         private int dni;
13         private static Persona persona;
14         2 referencias
15         private Persona(int dni)
16         {
17             this.dni = dni;
18         }
19
20         2 referencias
21         public static Persona GetPersona(int dni)
22         {
23             if(Persona.persona is null)
24             {
25                 Persona.persona = new Persona(dni);
26             }
27
28             return Persona.persona;
29         }
30     }

```

Ejercicio cotizador: <https://www.onlinegdb.com/BTxLSn-W4>



Recuerden que los operadores, cuando los sobrecargamos, los hacemos estáticos

Lucas Bayon

18:54

Recuerden que es "Public STATIC [retorno] operator"

Lucas Ferrini

18:54

Es decir, afectan a la clase y no a la instancia

CLASE 5

INTERFAZ son los elementos gráficos lo que le permite a los usuarios interactuar con la computadora. Las interfaces gráficas (GUI) le permiten al usuario interactuar con los programas, ejemplo puntero del mouse o iconos. No se necesita utilizar la consola.

-Respuesta inmediatas

- iconos familiares

Windows form aplicaciones de desarrollo para windows usada en .net permite desarrollar aplicaciones complejas para windows. Actua como una interfaz de usuario local en windows. El primer Form se instancia en el main, no en un metodo

Son objetos se desprenden de la clase Form que exponen prop y métodos que van a definir su comportamiento, además se pueden encontrar eventos, que definen la interacción con los usuarios. tienen que ser instanciados

SE PUEDEN TRATAR COMO CUALQUIER OTRA CLASE

Contamos con el diseñador de visual studio. Permite:

- Organizar los controles mediante guías de alineación.
- Establecer los márgenes y rellenos de los controles.
- Establecer los valores de propiedad con la ventana propiedades.
- Generación de eventos.
- Copiar/Cortar/Pegar/Eliminar.

new: se crea

load: se carga

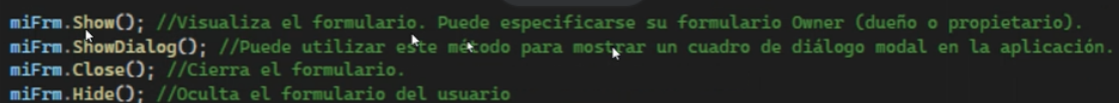
paint: se genera

activated

formclosing: chequear

Clase parcial: partial class, se introdujo en net2.0 permite separar el código de una clase en dos archivos fuentes(Form.cs y FormDisigner.cs) o mas. separa la lógica del diseño

El objeto form es el principal componente de una app windows.



```
miFrm.Show(); //Visualiza el formulario. Puede especificarse su formulario Owner (dueño o propietario).
miFrm.ShowDialog(); //Puede utilizar este método para mostrar un cuadro de diálogo modal en la aplicación.
miFrm.Close(); //Cierra el formulario.
miFrm.Hide(); //Oculta el formulario del usuario
```

Show dialog no interactua con otras ventanas

- ✓ **Name:** Indica el nombre utilizado en el código para identificar el objeto.
- ✓ **BackColor:** Indica el color de fondo del componente.
- ✓ **Cursor:** Indica el cursor que aparece al pasar el puntero por el control.
- ✓ **Enabled:** Indica si el control está habilitado.
- ✓ **Font:** Fuente utilizada para mostrar el texto en el control.
- ✓ **ForeColor:** Color utilizado para mostrar texto.
- ✓ **Locked** Determina si se puede mover o cambiar el tamaño del control.
- ✓ **Modifiers** Indica el nivel de visibilidad del objeto.
- ✓ **TabIndex** Determina el índice de tabulación que ocupará este control.
- ✓ **Text:** Texto asociado al control.

Los eventos a veces quedan guardados en el designer. Se debe ir al evento, se desasocia y si no se borra del método se borra manualmente del código lógico

MessageBox cuadro al usuario para interactuar

Control: componentes que se pueden reutilizar. encapsulan la funcionalidad de una interfaz grafica

CLASE 6

MATRICES estructuras de datos que permiten guardar varias variables del mismo tipo. Son utiles cuando se utilizan un numero determinado de objetos

The screenshot shows a Google Slides presentation with the title "CARACTERÍSTICAS DE LAS MATRICES". The slide contains four numbered points, each with a title and a description:

01.	02.	03.	04.
DIMENSIONALIDAD	TAMAÑO FIJO	INDEXACIÓN BASE-CERO	VALOR POR DEFECTO
Unidimensionales Multidimensionales	Se define cuando se instancia. No puede ser modificado.	Indice numerico. Comienza en el cero.	Todos lo elementos son inicializados con un valor por defecto.

Puede ser unidimensional, multidimensional, escalonada (jagged) o anidado

ARRAY: son una instancia de la clase System.Array y se inicializan en 0. Un array con n elementos irá de 0 a n-1 o sea,

```
int [] edades = new int[5] = 0, 1, 2, 3, 4
```

Se puede usar un for each para recorrer el array. Los [] se colocan detrás del tipo y no detrás de la variable.

REF Y OUT: ambos pasan parametros por referencia(dirección de memoria) pero ref necesita que la variable este inicializada.

MATRICES MULTIDIMENSIONALES

Más de una dimensión. Matrices de dos dimensiones:

INSTANCIAR

```
int[,] array = new int[3, 2];
```

INICIALIZAR

```
array = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```

ACCEDER A LOS ELEMENTOS POR ÍNDICE

```
int elemento = array [2,1];  
Console.WriteLine(elemento);  
//Output: 6
```

	Column 0	Column 1
Row 0	a[0][0]	a[0][1]
Row 1	a[1][0]	a[1][1]
Row 2	a[2][0]	a[2][1]

MATRICES ESCALONADA (Jagged Array)

Es una matriz cuyos elementos son matrices, posiblemente de diferentes tamaño
Array de arrays.

INSTANCIAR

```
int[][] jaggedArray = new int[3][];
```

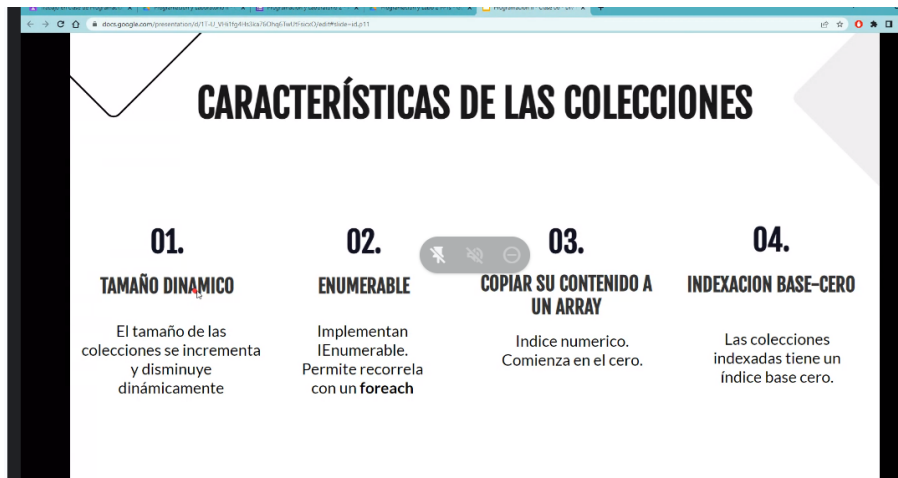
INICIALIZAR

```
jaggedArray[0] = new int[] { 1, 3, 5, 7, 9 };  
jaggedArray[1] = new int[] { 0, 2, 4, 6 };  
jaggedArray[2] = new int[] { 11, 22 };
```

ACCEDER A LOS ELEMENTOS POR ÍNDICE

```
Console.WriteLine(jaggedArray[0][3]);  
//Output: 7
```

Colecciones otra manera de agrupar objetos. ES UNA CLASE. Se puede aumentar o reducir dinámicamente sin intervención del programador a medida que el programa lo necesita.



GENÉRICA

Tiene el mismo tipo de dato. Cuando se recupera el dato no necesitamos averiguar cuál es o convertirlo porque ya sabemos el tipo

using

`System.Collection.Generic;`

Es útil cuando todos los elementos tienen el mismo tipo de dato. ejemplo la clase Persona

`List<tipoDeDato> variableLista;` → se debe instanciar

`variableLista = new List<tipoDeDato>();`

`variableLista.Add(loQuequiero agregar)`

`variable.Add(1)` → se agrega el número 1 a la posición 0

`variable.Add(56)` → se agrega el número 56 a la posición 1

`foreach (int variableBuscar in variableLista)`

{

`Console.WriteLine(variableBuscar)`

}

`variableLista.Clear()` → limpia todos los elementos de la lista.

No genéricas: agrega datos de tipo object, o sea cualquier tipo de datos.

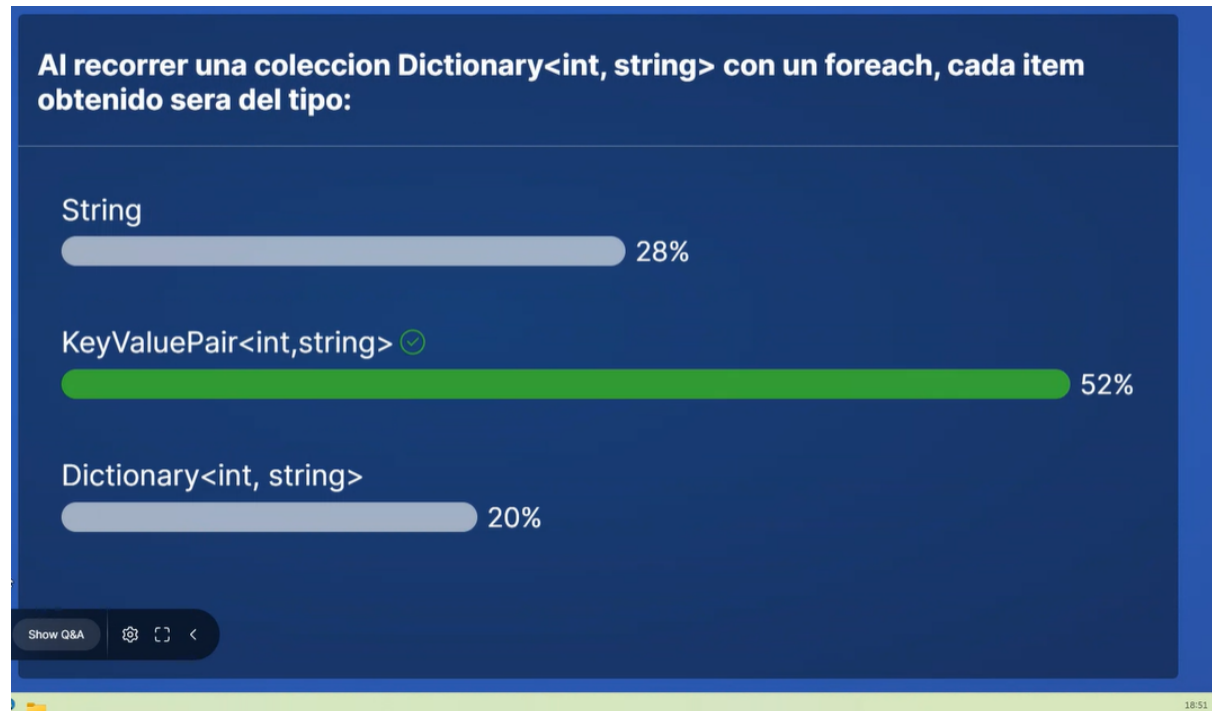
Queue no puede cambiar la dimensión es una cola y agarra al primero de la fila, primero en entrar, primero en salir. El `foreach` no puede modificar el tamaño de la fila. No se puede ordenar

Stack, pila. El último en entrar es el primero en salir. Como una pila de platos que a pilas para lavar.

https://www.youtube.com/watch?v=xXd-44d_xgc&t=2s

CLASE 7

Concepto que permite agrupar los datos del objeto. Permite ocultar los detalles de la implementación y proteger el estado del objeto. Oculta los procesos internos. El programador accede solo a lo que necesita.



SortedList indexado por el numero correspondiente al elemento y por una clave/ key

sustitucion de liskov; si tengo una clase vehiculo → le puedo guardar un taxi en la clase base se puede guardar una clase derivada.

Se puede hacer:

```
Vehiculo v = new Taxi();
```

Pero no se puede hacer:

```
Taxi t = new Vehiculo();
```

: base inicializa los elementos de la clase padre

El significado del modificador de acceso protected depende de la relación entre la clase que tiene el modificador y la clase que intenta acceder a los miembros que usan el modificador.

Para una clase derivada, la palabra reservada protected es equivalente a la palabra public.

Entre dos clases que no tengan una relación base-derivada, por el contrario, los miembros protegidos de una clase se comportan como miembros privados para la otra clase.

Cuando una clase derivada hereda un miembro pr

Cuando una clase derivada hereda un miembro protected, ese miembro también es implícitamente un miembro protegido de la clase derivada.

Esto significa que todas clases que deriven directa o indirectamente de la clase base pueden acceder a los miembros protegidos.

Los métodos de una clase derivada sólo tienen acceso a sus propios miembros heredados con protección. No pueden acceder a los miembros protegidos de la clase base a través de referencias a ésta

<https://youtu.be/45m7yJelX2c>

```
Entidades.Competencia
MostrarDatos()

{
    a.EnCompetencia = true;
    a.VuestrasRestante = c.cantidadVuestras;
    a.CantidadCombustible = (short)rnd.Next(15, 100);
    c.competidores.Add(a);
    return true;
}

return false;
}

0 referencias
public static bool operator -(Competencia c, AutoF1 a)
{
    if(c.competidores.Count > 0 && c == a/*lo retiro si esta*/)
    {
        a.EnCompetencia = false;
        c.competidores.Remove(a);
        return true;
    }
    return false;
}

2 referencias
public static bool operator ==(Competencia c, AutoF1 a)
{
    foreach (AutoF1 item in c.competidores)
    {
        if(item == a)
        {
            return true;
        }
    }
    return false;
}

1 error: 1 problema. | 66 líneas | 125 caracteres
```

CLASE 9

HERENCIA: Si el constructor de la clase base no tiene parámetros no es necesario llamarlo

Tratar de forma diferente a los mismos objetos. Habilidad de los objetos de responder al mismo mensaje de distinta forma

En tiempo de ejecución cada uno responde como responde según el tipo de objeto. Mostrar datos responde como una moto cuando en una moto y como un auto cuando es un auto

animal emitirSonido() → perro y gato pueden emitir sonido pero gato va a emitir miau y perro guag

protected	estrellita
private	candado
letra <i>italic</i>	abstracto

poli: se puede reutilizar la implementación de la clase base → se resuelve en tiempo de ejecución
no polimórfica: se anula el método de la clase base y se utiliza una nueva implementación

ampliar: se agrega mas info a la mostrada. utilizando el mismo método

```
public override string MostrarDatos()
```

```
return "El color es es {this.color} y mis datos son base.MostrarDatos()"
```

Redefinir : cambiar el resultado

```
public override string MostrarDatos()
```

```
return "El color es es {this.color}"
```

Chequear info de las 20:33

override no puede cambiar la accesibilidad

Se puede hacer un override a otro override.

clases abstractas: no se pueden instanciar, proporcionan una definición que modelan una jerarquía de herencia. no se puede usar el operador new en una clase abstracta. No se puede sellar, porque el propósito es modelar una jerarquía de herencia.

En un árbol, la raíz es una clase abstracta y de ahí salen el tronco, las hojas ya su vez estas últimas es una clase sellada, porque no hay nada más después de las hojas.

abstract: modificador, una clase incompleta que se utiliza como una clase base. La clase no abstracta debe incluir instancias reales de todos los miembros de la clase abstracta. La clase que hereda de una clase abstracta necesita implementar las propiedad abstracta.

No se puede declarar un método abstracto en una clase común.

incluir una documentación

CLR llama a los atributos estaticos