Platzi

Curso Fundamentos de C# con NetCore

Juan Carlos Ruiz

@JuanKRuiz

Senior Software Engineer

Microsoft

Módulo 1

Introducción

Instalación .Net Core SDK

Instalación en

Windows

Linux

OSx

Mi primer programa C#

Demo

Hola mundo en .net Core

```
dotnet new console --output sample1
dotnet run --project sample1
dotnet build
```

```
dotnet build -c Release -r win10-x64
```

```
win7-x64
win10-x64
osx.10.11-x64
ubuntu.16.04-x64
```

Instalación de visual studio code

Instalación en

Windows

Linux

OSx

Instalar los complementos necesarios en vs-code

Extensión para C#

Aceptar sugerencias del editor

Revisar extensiones disponibles

Demo

Hola mundo en .net Core

Desde Visual Studio Code

RESUMEN

- Instalar .net Core
- Instalar vs-code
- Crear una aplicación por comandos
- `dotnet run/build/new`
- Estructura de una aplicación básica
- Generar ejecutables específicos para plataformas
- Crear aplicaciones desde vs-code

Historia de C#

Creado por Anders Hejilsberg

Y su equipo.

También creador de Delphi y protagonista de otra larga lista de innovaciones entre ellas TypeScript

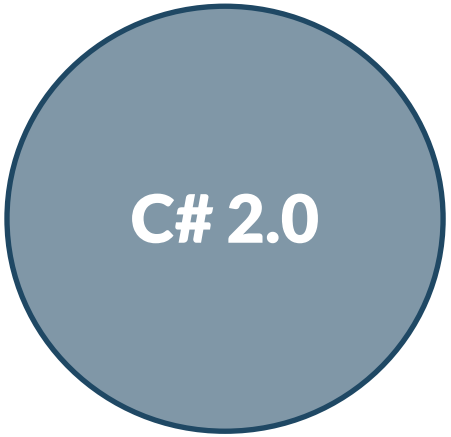
Lenguaje Multiparadigma

- strong typing
- imperative,
- Declarative
- Functional
- Generic
- object-oriented (class-based)
- component-oriented

Principios de C#

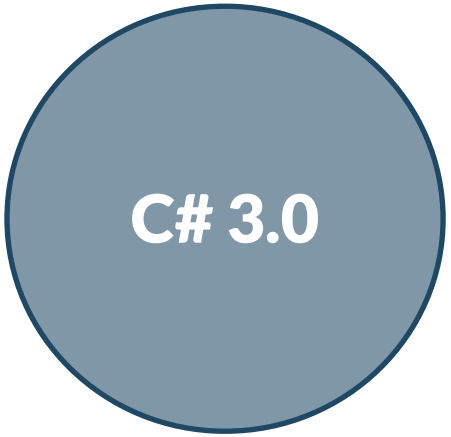
- simple, moderno, de propósito general , orientado a objetos.
- robusto, durable, productividad para el programador
- Ambientes distribuidos
- Portabilidad
- Soporte para internacionalización
- Uso en servidores y entornos embebidos
- Bajo uso de procesador y memoria

Versiones



- Generics, Partial types
- Anonymous methods
- Iterators
- Nullable types
- Getter/setter separate accessibility
- Method group conversions Co- and Contra-variance
- Static classes
- Delegate inference

Versiones

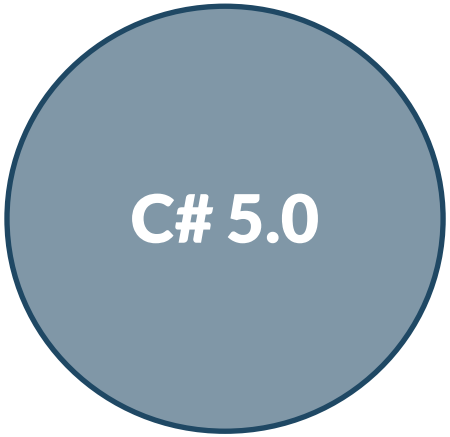


- Implicitly typed local variables
- Object and collection initializers
- Auto-Implemented properties
- Anonymous types
- Extension methods
- Query expressions
- Lambda expression
- Expression trees
- Partial methods

Versiones

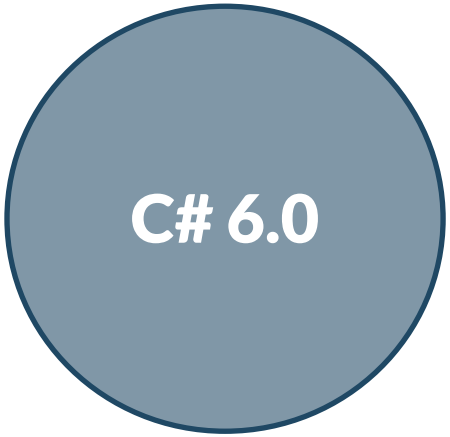
- Dynamic binding
- Named and optional argument
- Generic co- and contravariance
- Embedded interop types ("NoPIA")

Versiones



- Asynchronous methods
- Caller info attributes

Versiones



- Compiler-as-a-service (Roslyn)
- Import of static type members into namespace
- Exception filters
- Await in catch/finally blocks
- Auto property initializers
- Default values for getter-only properties
- Expression-bodied members
- Null propagator (null-conditional operator, succinct null checking)
- String interpolation
- nameof operator
- Dictionary initializer

Versiones

- Out variables
- Pattern matching
- Tuples
- Deconstruction
- Local functions
- Digit separators
- Binary literals
- Ref returns and locals
- Generalized async return types
- Expression bodied constructors and finalizers
- Expression bodied getters and setters
- Throw as expression

Versiones

- Async main
- Default literal expressions
- Inferred tuple element names

Versiones

- Reference semantics with value types
- Non-trailing named arguments
- Leading underscores in numeric literals
- private protected access modifier

Versiones

C# 7.3

- Accessing fixed fields without pinning
- Reassigning ref local variables
- Using initializers on stackalloc arrays
- Using fixed statements with any type that supports a pattern
- Using additional generic constraints

¿Por qué se llama C#?

C

C++

(C++)++

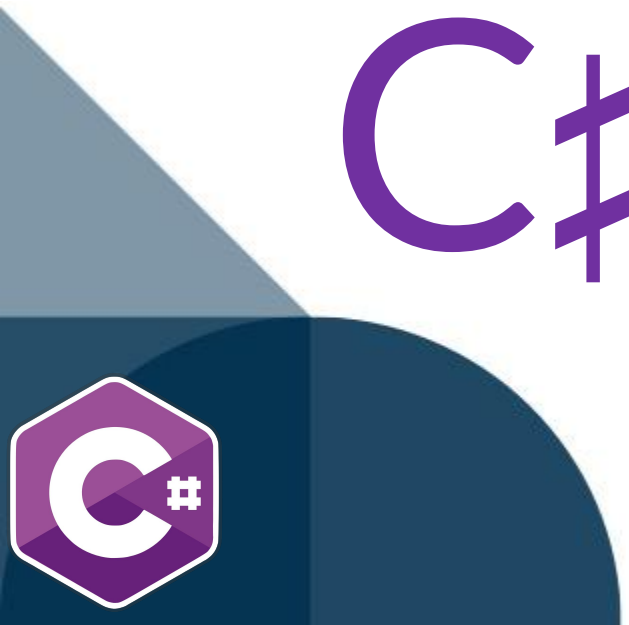
C++

C#

C#

!=

C# → C#



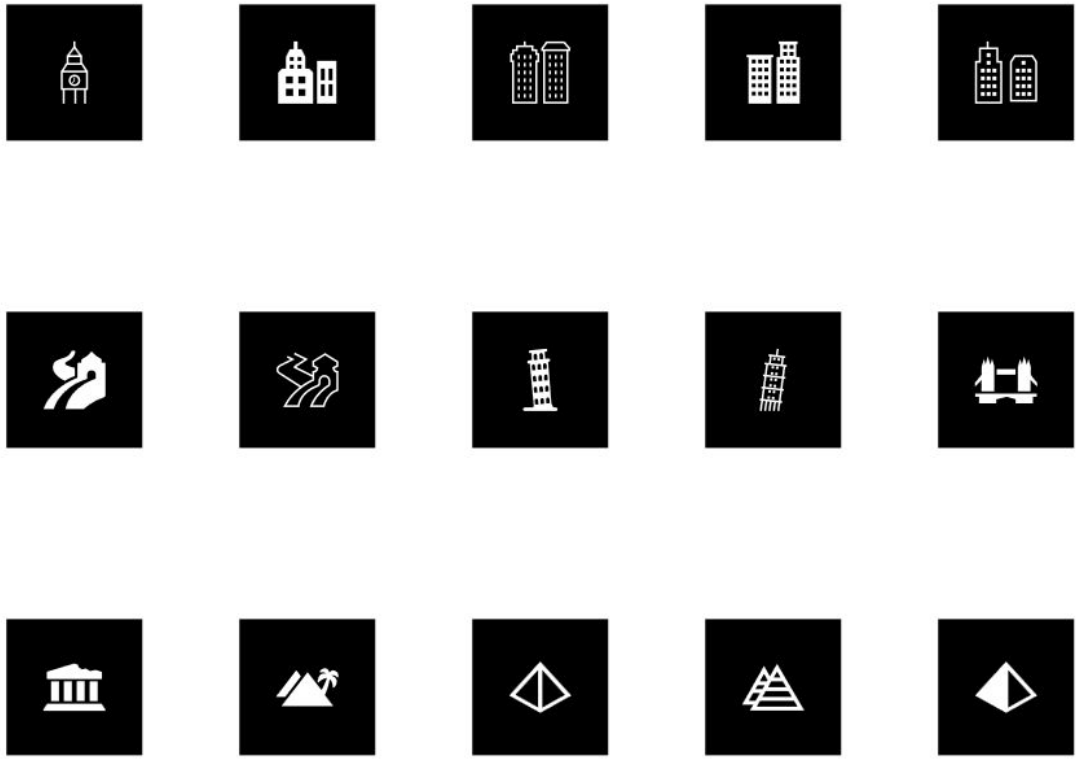
RESUMEN

- Breve Historia de C#
- Un recorrido de sus versiones
- Principios con los que se creó el lenguaje y su contexto histórico
- Porque se llama C#

Módulo 2

Clases y Objetos

¿Qué puede ser descrito como un objeto?



Escuela ← Objeto

Atributos →

- Nombre
- Ciudad
- Tipo:
  - Preescolar
  - Primaria
  - Secundaria

- Iniciar Jornada Académica
- Terminar Jornada Académica
- Timbrar
- Iniciar Emergencia

← Métodos

Objeto

- Es descrito por sus atributos
- Sus acciones se representan con métodos (funciones)

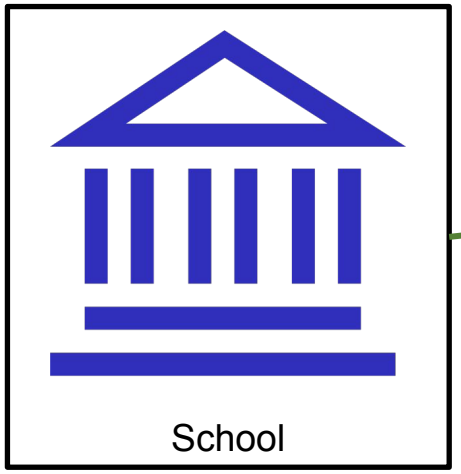
Entonces un objeto es una clase?

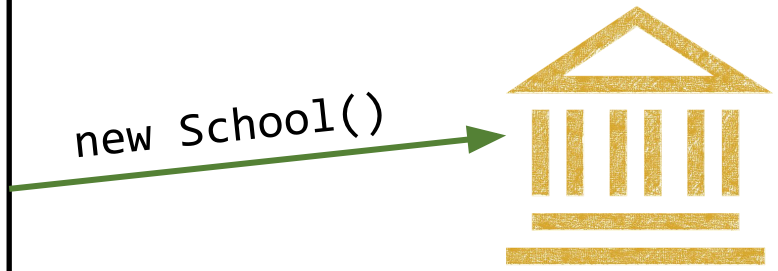
NO

La clase es la descripción de un objeto.

Pero no es el objeto, es una plantilla para crear objetos.

Clase School





Demo

Clase Escuela, sus atributos , propiedades y métodos

RESUMEN

- Qué son las clases, los objetos y sus diferencias
- Estructura básica de una clase
- Modificadores de acceso principales: `private`, `protected`, `public`
- Uso de la clase `Console` para escribir texto y hacer Beep
- Fundamentos de Atributos y Campos
- Creación básica de objetos
- Tipos de datos básicos: `int`, `string`
- Cómo usar atributos y métodos de un objeto

Aprendamos con un proyecto

Proyecto: CorEscuela

- Administrar una escuela pequeña
- Manejar los alumnos de cada grado
- Controlar sus asignaturas
- Controlar sus evaluaciones
- Elaborar Informes.

Etapa 1 – Creando la escuela

Tipos de Dato basicos

Tipo	operadores
numérico	int, float, long, double, decimal, byte
texto	string, char
personalizados	class, struct, enum, interface

Hay muchos más y se pueden agrupar de diferentes maneras

Tipos Básicos	https://docs.microsoft.com/es-es/dotnet/csharp/basic-types
Tabla de referencia de todos los tipos	https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/keywords/reference-tables-for-types

Resumen

- Inicialización de objetos (object)
- Propiedades
- Constructor de una clase
- Uso de Tuplas
- Tipos de dato: string, int
- Uso de espacios de nombres
- Enumeraciones
- La ‘,’como separador de listas
- El ‘;’ como fin de instrucción
- Concatenación de cadenas con +, $
- Parámetros opcionales

Etapa 2– Creando otros objetos para la escuela

Comparaciones y el operador if

Operadores deC#

Tipo	operadores
Aritméticos	+,-,*,/,=,++,--
Lógicos	&&, ||, ==, !=, >, <, >=, <=
Binarios	&, |, ^, >>, <<, ~
De palabras clave	new, typeof, sizeof, nameof, checked, uncheked, default

Hay muchos más y se pueden agrupar de diferentes maneras

Hay más de 40 operadores, la mayoría de ellos son combinaciones de tipos más básicos

docs.microsoft.com/es-es/dotnet/csharp/language-reference/operators/

Resumen

- Uso de using static
- Comentarios con //
- ¿Qué son los Arreglos?
- Diferentes formas de inicializarlos
- Adicionar arreglos como atributos
- ¿Cómo recorrer arreglos por medio de while, do while y for?
- Operador +=, ++, > y <
- La palabra clave null
- Uso de la sentencia if
- Operador == , !=
- ¿Cómo validar tipos nulos?
- ¿Cómo validar nulos usando el operador?

Etapa 3 – Implementado colecciones para el manejo de cursos

Arreglos y Colecciones

Arreglos	Colecciones
Son mucho más rápidos en memoria	Más fáciles de manipular
Consumen menos memoria	Múltiples variantes y, especializaciones para cada tarea
Si utilizas tipos nativos son de lejos mucho más eficientes	Tamaños flexibles
Ideal para trabajar APIS de bajo nivel	Extensibles, personalizables

Tipos de colecciones

Tipo	Descripción	Clases
Simples	Manipulan todo como tipos object	ArrayList, BitArray, Queue, Stack, Sorted
Specialized	Implementaciones comunes	StringCollection, BitVector, ListDictionary, NameValueCollection
Genéricas	Usan generics para crear implementaciones optimizadas de colección según el tipo de dato configurado	Dictionary<T,K>, List<T>, Queue<T>, Stack<T>, HashSet<T>, SortedSet<T>
Concurrent	Preparadas para acceso concurrente	ConcurrentBag<T>, ConcurrentStack<T>, ConcurrentQueue<T>

docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections

Resumen

- Limitaciones de los array y beneficios de las colecciones
- Como crear una colección y adicionar elementos
- Cómo borrar elementos de una colección
- /**/
- El concepto de delegado
- Formas de representar un delegado
- Tipo de dato bool
- Retornar valores de una función

Etapa 4 – Refactorizando y cargando datos de prueba

Resumen

- Concepto de refactoring
- Tipo double
- Clases estáticas
- Crear entidades para Alumno, Asignatura, Escuela, Evaluación
- La clase Guid para generar identificadores únicos
- Generación de datos de prueba con mecanismos aleatorios
- Crear datos de prueba haciendo producto cartesiano con Linq
- La clase Random, generar números aleatorios

RETO

- Implementar la carga aleatoria de evaluaciones
- 5 evaluaciones x asignatura
- Por cada alumno de cada curso
- Notas al azar entre 0.0 y 5.0

Etapa 1 – Creando la escuela

Etapa 2– Creando otros objetos para la escuela

Etapa 3 – Implementado colecciones para el manejo de cursos

Etapa 4 – Refactorizando y cargando datos de prueba

Platzi

Curso Fundamentos de C# con NetCore

Juan Carlos Ruiz

@JuanKRuiz

Senior Software Engineer

Microsoft

Platzi

Curso de C# con NetCore

Juan Carlos Ruiz

@JuanKRuiz

Senior Software Engineer

Microsoft

Etapa 5 – reutilizar tanto código como sea posible

Resumen

- Herencia
- Sobreescritura de métodos
- Debugger y el método ToString
- Clases abstractas y selladas
- Polimorfismo
- GetType
- Casting
- Varias referencias a un mismo objeto
- Excepciones en tiempo de ejecución
- Operadores : is, as

Etapa 6 – Ajustes y Funcionalidad

Resumen

- Interfaces
- Implementación explícita e implícita
- Palabra clave this
- Parámetros opcionales
- Uso del depurador
- Inspección de variables
- Inspección automática
- Uso de #region

Etapa 7 – Más funcionalidades para nuestro proyecto

Resumen

- Dictionary<K,T>
- más Refactoring
- Ienumerable<T>
- Cast<t>()
- Uso del operador switch
- Detectar la salida de un programa de Consola
- Event
- Variables de salida out

Etapa 8 – Consultas

Resumen

- DictionaryTryGet
- Linq Query
- Distinct
- from, select, where
- Orderby, groupby

RETO

- Crear un reporte que muestre solo los mejores X promedios por asignatura
- X debe ser pasado como parámetro
- El reporte contiene la asignatura y una lista de los Top X alumnos con su promedio

Lecturas recomendadas

Lectura	Link
Introducción a las consultas LINQ (C#)	https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries
Operaciones básicas de consulta LINQ (C#)	https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/linq/basic-linq-query-operations

Etapa 9 – Creando una UI de Consola

Resumen

- Como capturar información desde la consola
- Excepciones
- Try/catch, finally
- Cascadeo de excepciones
- Buenas prácticas con las excepciones

RETO

- Crear métodos para mostrar por consola cada uno de los reportes.
- Mantener un Formato visualmente agradable
- Privilegiar la experiencia del usuario al leer el reporte sobre la cantidad de información mostrada.
- No hay una única solución, cada cual puede hacerlo a su manera.