f**echa de creación/publicación/aparición y creadores de LDP**

C# (originalmente llamado Cool) fue creado por Microsoft entre los años 1999 y 2000. El líder del equipo fue Anders Hejlsberg.

\* **principales objetivos y usos para los que el LDP fue creado**

-Ser un LDP moderno, simple, multi-propósito y orientado a objetos.

-El lenguaje y sus implementaciones apoyan principios de la ingeniería de software tales como: “strong type checking” (sensibilidad en el formato de escritura, es decir, diferenciar entre mayúsculas, minúsculas y caracteres especiales), “array bounds checking” (revisión de los límites de un array), “detección de variables no inicializadas” y “recolección automática de basura”.

-Ser un LDP robusto, duradero y que aporte a la productividad del programador.

-Ser un LDP para uso en el desarrollo de componentes de software adecuadas para los entornos de lanzamiento y distribución.

-Ser un lenguaje “portable”, especialmente para ‘programas fuente” y programadores familiarizados con C y C++.

-Ser un LDP con soporte para la internacionalización.

-Ser un LDP adecuado para escribir aplicaciones tanto para sistemas locales, como para sistemas embebidos. Siendo usado para escribir desde pequeños programas, con funciones específicas, hasta grandes y sofisticados sistemas operativos.

**\* herramientas de desarrollo o IDEs disponibles**

-Visual Studio

-Project Rider

-Visual Studio Code

-Mono Develop

-entre otras.

\* **plataformas soportadas**

Windows, Linux y Mac

**\* ¿Cómo se implementa el encapsulamiento de atributos y métodos?**

Se antepone “public”, “protected” o “private” a la declaración del atributo o método, según el grado de encapsulamiento que se desee.

\*¿**Cómo se crea una clase?**

**\*¿Cómo se implementa el constructor de una clase?**

namespace NombreDelProyecto{

public class NombreDeLaClase{

public string NombreVariable1;

protected string NombreVariable2;

private int NombreVariable3;

public NombreDeLaClase( Variables de entrada para el constructor ) {

la implementación del Constructor va aquí

}

}

}

\***¿Cómo se implementa la herencia?**

namespace NombreDelProyecto{

public class NombreDeLaClaseHija **: NombreDeLaClasePadre**{

public string NombreVariable1;

protected string NombreVariable2;

private int NombreVariable3;

public NombreDeLaClaseHija( Variables de entrada para el constructor de la clase hija) **: base(Variables de entrada de la clase Padre)**{

la implementación del Constructor va aquí

}

}

}

**En C# una clase solo puede tener una clase Padre, pero una clase Padre puede tener muchas clases Hijas y cada clase Hija puede a su vez ser la clase Padre de una n-ésima clase.**

**Las secciones rojas hacen referencia a la clase Padre y debiesen ir en la misma línea que las declaraciones de la clase hija. La primera sección roja indica que la clase Hija está heredando de la clase Padre. La segunda sección Roja especifica que la clase hija está heredando el constructor de la clase Padre. Todo lo escrito en la sección “implementación del constructor va aqui” se considera como una extensión del constructor de la clase Padre. La clase Hija no está obligada a heredar el constructor de la clase Padre y puede, en cambio, tener su propio constructor. Si la clase Hija hereda el constructor de la clase Padre, entonces, las variables de entrada de su constructor deben contener *al menos* las variables de entrada del constructor de la clase Padre.**

\* **¿Cómo se instancia un objeto?**

NombreClase nombreObjeto = new NombreClase( variables de entrada (solo si son necesarias) );