1. Haz un repaso histórico del nacimiento y la evolución del lenguaje de programación C# y de la plataforma .NET.

En el año 2000, Microsoft presenta su plataforma .NET junto con un nuevo lenguaje, C# (diseñado por Anders Hejlsberg), que servirá de lenguaje principal de la plataforma. C# es un híbrido de C++ y Java que fusiona, principalmente, la capacidad de combinar operadores propia del primero (no incorpora la herencia múltiple) con la plena orientación a objetos del segundo. La orientación a objetos es tal que el propio programa está encapsulado en una clase.Actualmente C# se encuentra entre los 10 lenguajes más utilizados. A pesar de su corta historia, ha recibido la aprobación del estándar de dos organizaciones: en el 2001 se aprueba el ECMA y en el 2003 el ISO.

2. Explica en que consiste la POO y cuales son sus pilares más importantes.

La abstracción:

La abstracción es la capacidad de obtener y aislar toda la información y cualidades de un objeto que no nos parezcan relevantes, para poder encapsularlos. Para ello separamos "mentalmente" los objetos y nos centramos en su comportamiento fundamental.

La encapsulación:

La encapsulación es la capacidad de ocultar los datos abstraídos, aislarlos o protegerlos de quién no desees que tenga acceso a ellos; otro objeto o función por ejemplo.

Herencia y reutilización:

La Herencia lo que nos dice es que puede crearse un objeto a partir de otro objeto ya existente. El nuevo objeto hereda todas las cualidades del objeto del que deriva y además puede añadir nuevas funcionalidades o modificar las ya existentes.

Polimorfismo:

El polimorfismo es la capacidad para que varias clases u objetos derivados de otros, reaccionen de manera diferente ante los mismos métodos. El polimorfismo se puede aplicar tanto a objetos como a funciones, por lo que podemos hablar de objetos polimórficos y de funciones polimórficas.

3. Haz un programa en C# que contenga al menos una clase y que muestre por pantalla un mensaje de saludo. Comenta cada parte del código.

```
namespace consola_prueba
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hola c#");
                 Console.ReadLine();
        }
    }
}
```

namespace consola_prueba:

- Ámbito que contiene un conjunto de objetos relacionados. (Espacio de nombre)

class Program:

- Es la clase que se crea nuestro programa o función.

static void Main(string[] args)

- Es el método estático publico, Main es el punto de inicio del programa. Como no devuelve nada se le añade void puesto que los métodos no pueden devolver nada.

Console.WriteLine("Hola c#");

- Console es un método de la consola, que pertenece al espacio de nombres System.
- WriteLine es una función que escribe un string añadiendo un salto de líneas al acabar de escribir.

Console.ReadLine();

- Console es un método de la consola, que pertenece al espacio de nombres System.
- ReadLine lee un string por la entrada.

4. Mira la ayuda de C# y haz una relación de los métodos de la clase "Console" que permiten limpiar la pantalla, leer información del teclado y mandar información para que se muestre en la pantalla.

Limpiar pantalla:

Clear(); Borra la información que se muestra en el búfer de pantalla y en la correspondiente ventana de la consola.

Leer pantalla:

- Write: Es un método sobrecargado que almacena otras variantes en este caso en la misma línea.
- WriteLine: Es igual a Write pero con un salto de línea.

Leer información

- Read: Es un método sobrecargado que almacena otras variantes en este caso en la misma línea.
- ReadLine: Igual que Read pero con un salto de línea.

5. Como ya sabemos en las clases hay métodos, algunos de ellos son especiales; constructores y destructores, habla de cada uno de ellos y de su utilidad.

Toda clase tiene un **constructor** público (se usa desde fuera) que se llama igual que la clase, tanto si se definen como si no. Si no se define, C# proporciona automáticamente un constructor por defecto, que inicializa las variables a cero (variables por valor) ó a null (variables por referencia). Si se define un constructor, deja de usarse el de por defecto.

Todas las clases tienen un **destructor** de la clase que no es llamado; se usa automáticamente al finalizar la clase, se puede utilizar para afianzar ciertas operaciones que queremos que se realicen antes de destruir un objeto. Por ejemplo, se podría usar el destructor para asegurarnos que se cierra una conexión a una base de datos, o que se cierra un archivo abierto del objeto en cuestión.

6. Los componentes principales de la plataforma .NET son: CLR, CLS, BCL y JIT. Habla de las características más importantes de cada uno de ellos.

Common Language Runtime(CLR): Entorno de ejecución de la plataforma.

Para poder interactuar completamente con otros objetos, sea cual sea el lenguaje en que se hayan implementado, los objetos deben exponer a los llamadores sólo aquellas características que sean comunes para todos los lenguajes con los que deben interoperar. Por este motivo, se ha definido Common Language Specification (CLS)

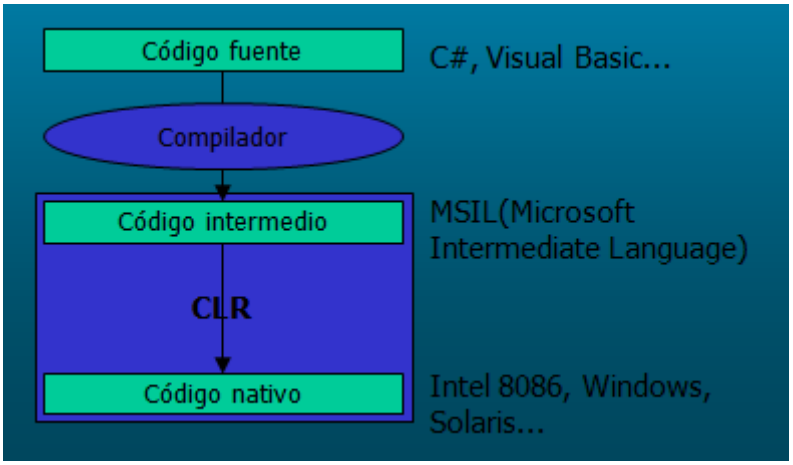
La Biblioteca de Clase de Base alberga componentes de funcionalidad mas general como las colecciones de datos, manejo de hileras, operaciones matemàticas y salida entre otros. Abarca la mayoria de namespaces (paquetes organizados de manera jerarquica).

JIT *(Just-In-Time)* es el que genera el código máquina real que se ejecuta en la plataforma del cliente. De esta forma se consigue con .NET independencia de la plataforma de hardware.

7. Que son y para qué sirven los namespace (espacio de nombres) en C#

La palabra clave namespace se utiliza para declarar un ámbito que contiene un conjunto de objetos relacionados. Puede utilizar un espacio de nombres para organizar elementos de código y crear tipos globales únicos.

8. Explica y haz un gráfico del proceso de compilación y ejecución de un programa de consola en la plataforma .NET



El proceso de compilación produce un fichero ejecutable en Windows al que se denomina portable executable (PE)
CLR sólo ve IL, por lo que se puede reemplazar el compilador JIT para usar un nuevo lenguaje

9. En qué consiste el MSIL, que inconvenientes y ventajas tiene

Microsoft Intermediate Language o MSIL es el lenguaje de programación legible por humanos de más bajo nivel en el Common Language Infrastructure (CIL) y en el .NET Framework. Los lenguajes del .NET Framework compilan a CIL. CIL es un lenguaje ensamblador orientado a objetos, y está basado en pilas. Es ejecutado por una máquina virtual.

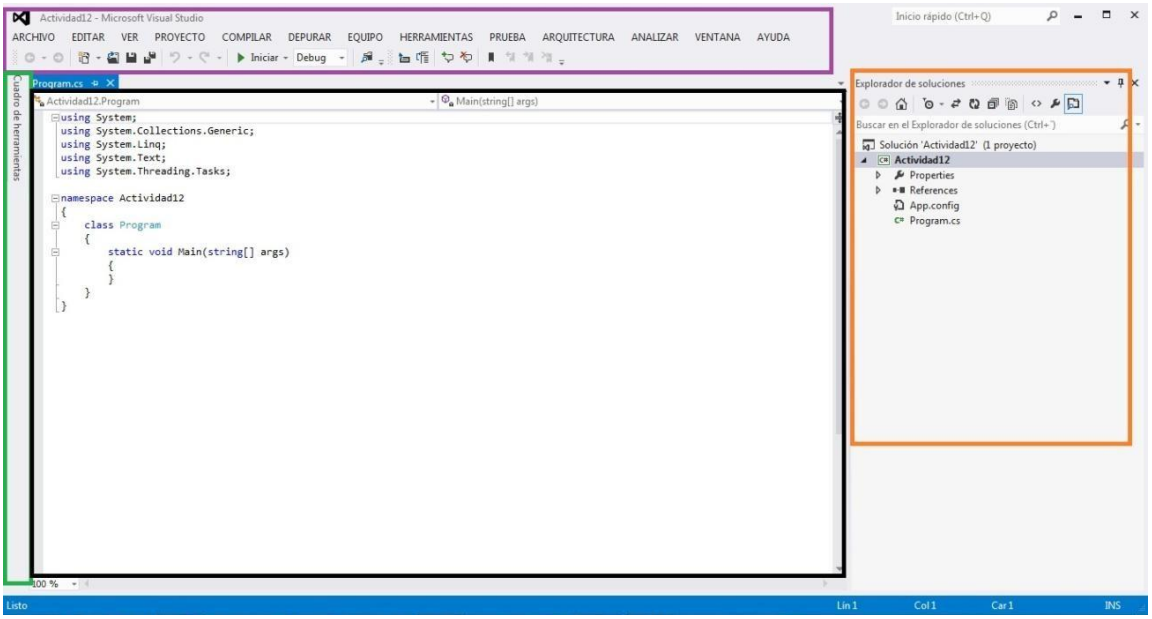
Ventajas:

1. Es emulado por lo tanto es seguro.
2. Altísimo grado de interoperabilidad.

Desventajas:

1. Se necesita la plataforma .NET.
2. La dependencia del código fuente de una aplicación.
3. Es propietario.

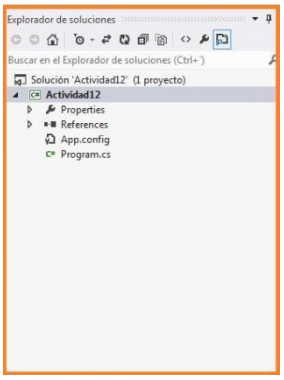
10. Haz un dibujo con las partes o ventanas que aparecen por defecto en el compilador de C# "MicrosoftVisual 2010".

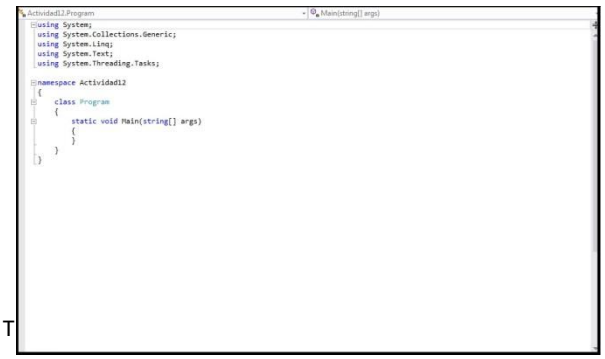


Distintos menús que visual estudio ofrece para la creación de software, tales como depurar compilar o abrir otras herramientas en pantalla.

Cuadro para las herramientas adicionales que necesitemos, se pueden poner arriba, abajo ya a ambos lados.

A la derecha un ejemplo de una herramienta abierta (en este caso el explorador de soluciones para ver los distintos proyectos que conforman la solución.





Lugar donde se escribe el código de la solución.