**Programación orientada a objetos:**

1. ¿Qué propone el **paradigma orientado a objetos** ? ¿Qué es un

paradigma?

Un paradigma es una forma de entender y representar la realidad: un conjunto de teorías, estándares y métodos que, juntos, representan un modo de organizar el pensamiento, es decir, un modo de ver el mundo.

El paradigma orientado a objetos (OO) define los programas en términos de comunidades de objetos surge para solventar los problemas con el objeto de elaborar programas y módulos más fáciles de escribir, mantener y reutilizar.

2. Nombre a los **pilares** del paradigma orientado a objetos.

Abstracción, Encapsulamiento, Herencia y Polimorfismo.

3. ¿Qué es una **clase** ?

Es una clasificación entorno a comportamientos y atributos comunes, es una abstracción de un objeto(Ignora lo que no es importante y utiliza encapsulación para reforzar la abstracción)

4. ¿Qué es un **objeto** ? ¿En qué se relacionan con las clases?

Los objetos con características comunes se agrupan en clases, son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas.

5. ¿Qué significa **instanciar** un objeto?

**Instanciar** objetos es el proceso de generar un ejemplar de una clase, es decir, la clase es como una declaración de una forma y el **objeto** es un caso o elemento concreto que responde a esa forma.

6. Explique qué es la **abstracción** en el contexto de programación

orientada a objetos y cuál es su relación con las clases.

Es la ignorancia selectiva, decidir que es importante y que no enfocándose en lo que SI lo es.

7. ¿Qué son los **atributos** o estado de un objeto?

Los **atributos** son las características individuales que diferencian un **objeto** de otro y determinan su apariencia, estado u otras cualidades.

8. ¿Qué son los **métodos** de un objeto?

Un **método** es una subrutina cuyo código es definido en una clase y puede pertenecer tanto a una clase, como es el caso de los **métodos** de clase o estáticos, como a un **objeto**, como es el caso de los **métodos** de instancia. Consiste generalmente de una serie de sentencias para llevar a cabo una acción, un juego de parámetros de entrada que regularán dicha acción o, posiblemente, un valor de salida (o valor de retorno) de algún tipo.

**Miembros estáticos y de instancia (no estáticos):**

9. ¿Qué es un **método estático** ? ¿En qué se diferencia de los métodos de

instancia (no estáticos)?

Un método de instancia es el que se invoca **siempre** sobre una instancia (objeto) de una clase. Por ejemplo p1.getNombre(); siendo p1 un objeto de tipo Persona es un método de instancia: para invocarlo necesitamos una instancia de persona. **Un método de clase es aquel que puede ser invocado sin existir una instancia.**

**Este último tiene las limitaciones de no poder acceder a un campo de instancia y tampoco pueden invocar a un método que no sea estático.**

10. ¿Qué es una **clase estática** ? ¿En qué se diferencia de las clases no

estáticas?

Una [static class](https://msdn.microsoft.com/en-us/library/79b3xss3.aspx) es aquella clase que se usa sin necesidad de realizar una instanciación de la misma son adecuadas cuando no tienen que almacenar información, sino sólo realizar cálculos o algún proceso que no cambie.

Una clase no estática (o dinámica) puede ser instanciada varias veces creándose los objetos que sean necesarios.

11. ¿Puedo tener miembros estáticos en clases no-estáticas? ¿Puedo

tener miembros no-estáticos en clases estáticas?

Una clase no estática **puede** contener métodos, campos, propiedades o eventos estáticos. El miembro estático es invocable en una clase, incluso si no se ha creado ninguna instancia de la clase.  Solo existe una copia de un miembro estático, independientemente del número de instancias de la clase que se creen.

A diferencia de una clase no estática, la estática NO solo contiene métodos estáticos, no puede contener constructores de instancias.

12. ¿Necesito instanciar un objeto de la clase para llamar a sus métodos

estáticos?

Si es esperado como argumento de ese método estático si, ya que se lo tiene que pasar por parámetro.

13. ¿Cuántos valores distintos pueden existir para un campo estático y

cuántos de uno no-estático cuando se crearon 5 instancias de la clase?

Para un campo estático 1 solo valor ya que pertenece a la clase y se comparte entre todas las instancias de la misma, los cambios en una instancia se verán reflejados automáticamente en las otras 4. Por el contrario, el no estático se crean 5 objetos distintos los cuales pueden modificar cada uno su campo sin alterar el de otro.

14. ¿Se puede usar el **operador “this”** dentro de un método estático? ¿Se puede acceder a miembros no-estáticos desde un método estático de la misma clase? ¿Se puede acceder a miembros estáticos desde un

método de instancia?

Los métodos y las propiedades estáticos **no** pueden tener acceso a campos y eventos no estáticos en su tipo contenedor, y tampoco pueden tener acceso a una variable de instancia de un objeto a menos que se pase explícitamente en un parámetro de método.

Por otro lado, los miembros no-estáticos (de instancia) pueden acceder a un miembro estático usando el nombre de la clase o una variable de referencia.

15. ¿Se pueden declarar variables estáticas dentro de un método

(locales)?

No, una variable estática es creada y se mantiene (aunque se cambie su valor) hasta la finalización de la ejecución del programa.

16. De dos ejemplos de métodos estáticos que pertenezcan a las clases de

.NET Framework.

Math.Abs() y Math.Pow()

17. De un ejemplo de un método de instancia que pertenezca a las clases

de .NET Framework.

Existen los llamados métodos de extensión, los cuales se definen como estáticos pero se les llama usando la sintaxis de método de instancia (Librería LINQ) que permiten “agregar” métodos a los tipos existentes sin crear un nuevo tipo derivado o modificar el tipo original.

Ejemplos: GroupBy(), OrderBy(), Average()...

**Namespaces y directivas:**

18. ¿Qué es un **namespace** y cuál es su función principal?

Se usan para organizar tantas clases como se necesite. Declarar namespaces propios nos ayudan a controlar el scope de clases y métodos en programas grandes.

19. ¿Puedo tener distintos namespaces dentro de un mismo proyecto o

ensamblado de .NET?

Si, se pueden tener namespaces adentro de otro namespaces.

20. ¿Para qué se usa la **directiva using** ?

Permite la especificación de una llamada a un método sin el uso obligatorio de un nombre completamente cualificado.

21. ¿Para qué se usa la **directiva alias** ?

Permite utilizar un nombre distinto para un Namespace.

Generalmente se utiliza para abreviar nombres largos.

22. ¿Puedo declarar dos clases independientes/distintas con el mismo

identificador dentro del mismo namespace ? ¿Y en namespaces

distintos?

Se puede, pero solamente en namespaces distintos.