1. **¿Qué es la Abstracción?**

es un pilar fundamental de la POO que permite ocultar la complejidad de la implementación interna de una clase y exponer solo las características y funcionalidades esenciales al usuario.

1. **¿Qué promueve la Abstracción?**

promueve la modularidad, la reutilización de código y la facilidad de uso, ya que los usuarios no necesitan comprender los detalles internos de las clases para utilizarlas.

1. **¿Qué es Herencia en C#?**

es otro pilar fundamental de la POO que permite a una clase (clase derivada) heredar atributos y métodos de otra clase (clase base). Esto crea una jerarquía de clases donde las clases derivadas pueden especializarse y extender la funcionalidad de las clases base.

1. **¿Qué promueve la Herencia?**

promueve la reutilización de código, ya que las clases derivadas no necesitan Re implementar la funcionalidad heredada de las clases base, cómo el ejemplo que vimos de los carros y teléfono.

1. **¿Qué es Polimorfismo?**

es la capacidad de un objeto de tomar diferentes formas en tiempo de ejecución. Esto se logra mediante la sobrecarga de métodos, que permite que un mismo método tenga diferentes implementaciones en diferentes clases derivadas.

1. **¿Qué permite el Polimorfismo?**

El polimorfismo permite escribir código más flexible y adaptable, ya que el mismo método puede ser invocado en diferentes objetos y producir resultados distintos dependiendo del tipo de objeto.

1. **¿Qué es la encapsulación?**

es un pilar fundamental de la POO que oculta los detalles de implementación de una clase y solo expone propiedades y métodos públicos, protegiendo los datos internos y promoviendo la modularidad.

1. **¿Qué permiten los modificadores en la encapsulación?**

public, private, protected e internal. La encapsulación permite controlar el acceso a los datos de una clase y protegerlos de modificaciones no deseadas.

1. **¿Qué es interfaces?**

Las interfaces en C# son conjuntos de métodos abstractos que definen un comportamiento que las clases deben implementar.

1. **¿para que se utilizan las interfaces?**

para promover el polimorfismo y la separación de interfaces, permitiendo que diferentes clases implementen el mismo comportamiento de maneras distintas. Las interfaces no contienen implementaciones de métodos, solo definen las firmas de los métodos que las clases deben implementar.

1. **¿Qué son los delegados?**

Los delegados en C# son objetos que encapsulan referencias a métodos. Permiten encapsular métodos en variables o estructuras de datos, facilitando la transferencia de métodos como argumentos y la creación de eventos.

1. **¿Cómo se invocan los delegados?**

Los delegados se invocan utilizando la sintaxis de punto de exclamación (!) seguida del nombre del delegado y paréntesis que pueden contener argumentos para el método encapsulado.

1. **¿Qué es un evento en C#?**

son mecanismos que permiten notificar a los objetos interesados cuando ocurre un cambio en el estado de otro objeto. Se implementan utilizando delegados y permiten la comunicación asíncrona entre objetos.

1. **¿Qué es un indexador en C#?**

Los indexadores en C# permiten acceder a los elementos de una colección como si fuera un array, utilizando corchetes [] y un índice.

1. **¿Cómo se definen los indexadores en C#?**

como propiedades con una implementación personalizada que permite acceder a los elementos de la colección de forma indexada. Esto facilita el acceso a los elementos de una colección sin necesidad de utilizar bucles o iteradores

1. **¿Qué es una propiedad en C#?**

son atributos de una clase que encapsulan datos y proporcionan métodos de acceso y modificación controlados. Se definen utilizando la palabra clave property seguida del nombre de la propiedad, el tipo de dato y una llave que contiene el getter y/o el setter. Las propiedades permiten acceder y modificar los datos de una clase de forma controlada y segura.