1. ¿Cuántos objetos alumno puedo tener en mi sistema?

   Todos los que quiera

2.¿Cuántas clases alumno tengo en mi sistema?

Una sola, no puede haber dos clases con el mismo nombre en el mismo namespace

3.¿Qué diferencia hay en la sintaxis de la llamada a un método estatico y uno de instancia?

Un método static lleva la palabra static después del modificador de acceso indicando que es una metodo de clase que no se necesita de una instancia del objeto para invocar el método, mientras que la de instancia no utiliza ninguna palabra y necesita anteponer un objeto instanciado para ser invocada.

4.¿Para que uso el this?

Tiene varios usos: Para hacer referencia al objeto que estamos utilizando, para crear un constructor a partir de otro que tenga distintos parámetros y para utilizar objetos distintos a la vez.

5.¿Puedo usar el this en un método estatico?

No, en un método statico no es necesario tener instanciado el objeto.

6.¿Puedo usar el this en un método de instancia?,¿Qué pasa si no lo uso?

Puede usarse o no, si no se usa debo aclarar objeto.atributo al cual me estoy refiriendo.

Guía de Estudio: Constructores

Constructores

1. ¿Cuál es la diferencia entre el operador “new” y un constructor?
   El operador new genera una instancia del objeto utilzando un constructor.
   Un constructor instancia los atributos de una clase
2. ¿A qué se le llama constructor por defecto?
   Al que inicializa los atributos de una clase a su valor por defecto (int = 0, string = “”, bool = false, referencia = null)
3. ¿Qué pasa si no declaro ningún constructor en la clase?
   EL compilador utiliza el constructor por defecto
4. ¿Qué se ejecuta primero un constructor de instancia o uno estático?
   Uno estatico.

Guía de Estudio: Sobrecarga

Sobrecarga de Métodos

1. ¿La sobrecarga se resuelve en tiempo de ejecución o en tiempo de compilación?
   En tiempo de compilacion
2. ¿Los métodos estáticos pueden ser sobrecargados?
   Si pueden ser sobrecargados
3. ¿Se puede sobrecargar un método basados en un tipo de retorno diferente?

   No, no se puede

4. ¿Se tiene en cuenta el nombre de los parámetros para la sobrecarga?
   No, no se tienen en cuenta
5. ¿Qué técnica es una alternativa a la sobrecarga de métodos en los lenguajes que no la soportan?
6. ¿Qué debe cambiar para que la sobrecarga de un método sea válida?
   Tipo de dato, orden y/o numero de parámetros de entrada Si tenemos diferentes versiones de un mismo método, ¿cómo evitamos repetir código?
   Haciendo llamamiento
7. ¿Los métodos pueden tener el mismo nombre que otros elementos de una misma clase? (atributos, propiedades, etc).
   Si de un mismo método(siempre y cuando sea sobrecarga valida) pero no de otro tipo de elemento
8. Repasar los ejercicios en la PPT “04-Sobrecarga.pptx”.
9. Para después de las clases de polimorfismo: ¿Cuál es la diferencia entre sobrecargar (overload) y sobrescribir (override)?
   Overload es la sobrecarga de un método declarando un método con el mismo nombre pero distinto retorno o distintos tipos de parámetros.
   Override es la sobreescritura de un método declarado como virtual o abstract en una clase padre que después va a ser heredada y la clase hija va a hacer un override de ese método para utilizarlo.

Sobrecarga de Constructores

10. ¿Se pueden sobrecargar los constructores estáticos?
    Si se puede
11. ¿Se puede llamar a un constructor estático con el operador “this”?
    No
12. ¿Se puede llamar a constructores de otras clases con el operador “this”?
    No, no se puede
13. ¿Se puede sobrecargar a un constructor privado?
    Si se puede, utilizando la palabra this
14. ¿Existe alguna alternativa al operador “this” a la hora de evitar repetir código o tareas en común entre los constructores? ¿Cuál?
    Si, tenes un método que haga la misma función que un constructor
15. Indique tres ejemplos distintos de sobrecargas para el constructor de la clase Persona sin repetir el criterio utilizado (ej. Cambiar el número de parámetros sólo se puede usar una vez):

```
public Persona (string nombre, string apellido, int edad) {}

public Persona (string nombre, string apellido, int edad) {}
public Persona (string nombre, string apellido, int edad, int dni)
:this(nombre,apellido,edad) {}
public Persona (string apodo, string nombre, int edad) {}
```

Guía de Estudio: Sobrecarga de Operadores

1. ¿Qué es un operador unario y que es un operador binario? De un ejemplo de cada uno.
   Un operador unario trabaja con un solo operando (++, !) y el binario trabaja con dos operandos (+ -, /,*)
2. ¿Qué varía en la sintaxis de la sobrecarga de operadores unarios y binarios?
   La cantidad de objetos recibidos por parametros
3. Los operadores de casteo "(T)x" no se pueden sobrecargar. ¿Cuál es la alternativa?
   Se pueden definer nuevos operadores de conversion
4. ¿Cuál es la diferencia entre una conversión implícita y una explícita?
   -Explicita: El programador tiene que escribir el resultado que quiere obtener, para no perder datos o cuando la conversión por alguna razón podría fallar.
   -Implicitas: no hace falta poner sintaxis, se entiende que no hay perdidas de datos

5. ¿Cuál es la diferencia entre castear (casting) y parsear (parsing)?
   -Casting, es una forma de decir al compilador que un objeto es algo más, de int a double, pueden ser implícitos o explicitos.
   -Parsing, es decirle al programa que trate de interpretar un string y trate de convertirlo en algo mas. Texto=10/10/2010 ->DateTime Parse

Guía de Estudio: Windows Forms

1. ¿Los formularios son objetos?
   Si, son objetos del tipo form
2. ¿De qué clase heredan todos los formularios?
   Heredan de la clase Form
3. ¿Qué es una partial class o clase parcial?
   Es un concepto que permite separar el codigo de una clase en dos partes.
4. ¿Puedo agregar parámetros de entrada a la clase del formulario? ¿Y sobrecargar el constructor? ¿Y declarar nuevos campos/propiedades?
5.
6. ¿Cuál es la diferencia entre Show() y ShowDialog()?
   el Show() abre un form pero no bloquea al resto, puede abrir varios formulario y acceder a uno u otro indistintamente.
   El ShowDialog muestra un form modal, o sea muestra el form y hasta que no lo cierras no podras acceder a los inferiors.
7. ¿Qué es un formulario MDI? ¿Con qué propiedad indico que un formulario es un contenedor MDI? ¿Con qué propiedad del formulario hijo indico cuál es el formulario MDI padre?
   Un formulario MDI es el que se utiliza en un proyecto para acceder a múltiples formularios.
   Se indica con la propiedad IsMdiContainer
   Se indica con la propiedad MdiParent

7. Detalle los pasos del ciclo de vida de los formularios.

New -> Load -> Paint -> Activated -> Closing -> Close -> Disposed.

Guía de Estudio: Colecciones

1. ¿Cuál es la diferencia entre las colecciones genéricas y las no genéricas?
   Una colección genérica cumple la seguridad de tipos para que ningún otro tipo de datos se pueda agregar a ella.
   En una colección no genérica se puede guardar desde un Integer hasta un Boolean
2. ¿Cuál es la diferencia entre las colecciones y las matrices?
   Las colecciones proporcionan un método más flexible para trabajar con grupos de objetos, ademas el grupo de objetos con el que trabaja puede aumentar y reducirse dinámicamente a medida que cambian las necesidades de la aplicación.
   Las matrices son muy útiles para crear y trabajar con un número fijo de objetos fuertemente tipados.
3. Describa los siguientes tipos de colecciones genéricas: Dictionary, List, SortedList.
   -Dictionary representa una colección de pares de clave y valor que se organizan por claves.
   - Representa una lista de objetos que pueden ser obtenidos mediante un índice.
   -SortedList representa una colección de pares de clave y valor que se ordenan por claves según la implementación de la interfaz IComparer<T> asociada.
4. ¿Cuál es la diferencia entre una cola (queue) y una pila (stack)?

   En la cola lo primero en entrar es lo primero en salir y en una pila lo primero en entrar es lo ultimo en salir.

5. ¿Se pueden ordenar las colas y las pilas?
   No, ellas mismas establecen un orden y es inalterable.
6. ¿Se pueden serializar (convertir a un formato en el que pueda ser transferido o leído por otro sistema) las colas y las pilas?
   idn
7. ¿Cuál es la diferencia entre las colas y pilas genéricas y las colas y pilas no genéricas?

   Las no genericas permiten guardar datos de cualquier tipo, incluso datos de distinto tipo en una misma estructura.

   Las genericas obliga a generar una coleccion de un tipo de dato especifico.

Guía de Estudio: Propiedades, Indexadores y Enumerados

1. ¿Cuál es el primer valor numérico de un enumerado por defecto? ¿Se pueden sobrescribir los valores por defecto?
   El primer valor numeric es 0 y si se pueden sobreescribir valores por defecto
2. Indique los valores asociados a cada constante: enum Day {Sat, Sun, Mon=15, Tue, Wed, Thu=2, Fri};
   Sat=0;Sun=1;Mon=15,Tue=16,Wed=17;Thu=2;Fri=3;
3. ¿Los indexadores solo se pueden indexar por valores numéricos?
   No es necesario indexarlos solo con numeros.
4. ¿Cuál es la diferencia a la hora de declarar un indexador y una propiedad?
   Fundamentalmente una propiedad NO recibe parametros y ademas se declara utilizando UpperCamelCase.
   Un indexador recibe parametros.
5. ¿Un indexador puede recibir más de un parámetro?
   Si, puede.

Guía de Estudio: Encapsulamiento

1. Defina abstracción.

   Es el concepto de definir que es importante y que no.

2. Explique las dos definiciones de encapsulación (encapsulación y ocultamiento de la información).

   Se denomina encapsulación al ocultamiento de los datos miembro de un objeto obligando a cambiarlos únicamente a través de las operaciones definidas para ese objeto.

   El ocultamiento se encarga de proteger a los datos asociados de un objeto contra su modificación por quien no tenga derecho a acceder a ellos.

3. Defina cada nivel de ocultamiento de la programación orientada a objetos.

   -Público: todos pueden acceder a los datos o métodos de una clase que se definen con este nivel, este es el nivel más bajo, esto es lo que tu quieres que la parte externa vea.

   -Protegido: podemos decir que estás no son de acceso público, solamente son accesibles dentro de su clase y por subclases.

   -Privado: en este nivel se puede declarar miembros accesibles sólo para la propia clase.

4. ¿Qué es la encapsulación con respecto a la abstracción?

   La encapsulación logra que los atributos y métodos de nuestra clase no estén visibles o accesibles para modificarlos. Si quisiéramos hacerlo tendríamos que crear métodos públicos para poder hacer las modificaciones internas, pero no tendríamos acceso directamente a los miembros de nuestra clase

Guía de Estudio: Herencia

1. ¿Qué nombre recibe la clase que hereda y qué nombre recibe la clase que es heredada?
   Hereda: Clase base o padre
   Heredada: Clase derivadas, subclases o hija.
2. ¿Qué significa que la herencia es transitiva?
   Que si A hereda de B y B hereda de C entonces A es derivada de C.
3. ¿Se heredan los constructores?
   No
4. ¿Se heredan los miembros private de la clase base?
   Si, se heredan pero no se puede acceder a ellos.
5. ¿Qué es herencia múltiple? ¿Es posible en C#?
   Se define herencia multiple cuando una clase derivada puede heredar de una o más clases base. En C# no, no lo soporta.
6. ¿Que nos permite la herencia? ¿Cuál es su propósito?
   La herencia nos permite crear clases nuevas a partir de otras ya existentes, su proposito es organizar major las clases que componen una misma realidad, agrupandolas en base a sus atributos y a su comportamiento.
7. ¿Una clase pública puede heredar de una clase privada?
   Una subclase no puede ser mas accessible que su clase base.
8. ¿Qué es una clase sellada (sealed)?
   Un tipo de clase que no puede ser heredada por otra clase.
9. ¿Una clase sellada puede heredar de otras clases? (Ser clase derivada)
   Si puede.

10. ¿Cómo actúa el modificador protected en los miembros de la clase base para una clase derivada y cómo para una clase no-derivada?
    Para una clase derivada el protected equivale al public y para una NO derivada equivale al private
11. ¿Qué pasa si la clase derivada no hace una llamada explícita a un constructor de la clase base? En esta situación, ¿qué pasa si la clase base declaró explícitamente un constructor con parámetros de entrada?
    Si la clase derivada no tiene un constructor declarado el compilador de C# proporciona automáticamente una llamada al constructor sin parámetros o predeterminado de la clase base.
    Si la clase base tiene declarado un constructor con parametros la clase derivada esta obligada a declarar un constructor que utilize los mismos parametros.

Guía de Estudio: Abstract y Virtual

1. ¿Cuáles son las diferencias entre sobrecargar (overload) y sobrescribir (override) un método?

Criterio	Sobrecargar / Overload	Sobrescribir / Override
Firma (Diferencias o no diferencias en las firmas)		
Ubicación (Misma clase / Clases diferentes)		
Tiempo de resolución (Compilación / Ejecución)		
Tipo objeto / Tipo Referencia (¿Qué determina cuál implementación se utilizará?)		

2. Si quiero declarar un método que **pueda** ser sobrescrito en las clases derivadas, ¿qué modificador debo usar?
   Virtual

3. Si quiero declarar un método que **deba** ser sobrescrito en las clases derivadas, ¿qué modificador debo usar?

Abstract

4. Las clases no-abstractas que derivan de una clase abstracta, ¿**deben** implementar **todos** sus métodos abstractos?
   Si

5. Las clases abstractas que derivan de una clase abstracta, ¿**deben** implementar **todos** sus métodos abstractos?
   Si

6. ¿Se pueden declarar miembros abstractos en clases no-abstractas?
   No

7. ¿Para sobrescribir un método se debe heredar de una clase abstracta?
   Si

8. Llenar los campos de la siguiente tabla con SÍ o NO según corresponda.

Tipo de Clase	**Puede heredar de otras clases**	**Puede heredarse de ella**	**Puede ser instanciada**
normal (sin modificadores)	Y	Y	Y
abstract	Y	y	N
sealed	Y	n	Y
static	N	N	N

Guía de Estudio: Polimorfismo

1. ¿Qué es polimorfismo en POO?

Es la capacidad de tratar objetos diferentes de la misma forma

2. ¿Cómo implementamos el polimorfismo en el código de C#?

Con la palabra reservada virtual

3. ¿Todos los objetos en C# son polimórficos?

Si (no estoy seguro)

4. La clase Alumno hereda de Persona. ¿Es una instancia de Alumno también de tipo Persona? ¿Se puede hacer Persona persona = new Alumno()? ¿Y Alumno alumno = new Persona()? ¿Esto es polimorfismo?

Si, en parte.

Si se puede hacer.

No, Persona no es un Alumno, pero Alumno si es una Persona

Si, es polimorfismo

5. Según el siguiente código:

```
class Persona
 {
        public virtual string teach()
        {
                return “Una persona puede enseñar”;
        }
}
class Profesor : Persona
{
        public override string teach()
         {
                return “Un profesor puede enseñar en un colegio”;
        }
}
```

```
class Program
{
        public static void main(string args[])
        {
                //Caso 1
                Persona persona = new Persona();
                Console.WriteLine(persona.teach());
                //Caso 2
                Persona otraPersona = new Profesor();
                Console.WriteLine(otraPersona.teach());
                //Caso 3
                Profesor profesor = new Profesor();
                Console.WriteLine(profesor.teach());
        }
}
```

a. Complete la siguiente tabla indicando en los campos de qué tipo es la referencia y de qué tipo es el objeto (Persona o Profesor):

Nombre Variable	Referencia	Objeto
persona	Persona	Persona
otraPersona	Persona	Profesor
profesor	Profesor	Profesor

b. Indique la salida por consola para cada caso:

Caso	Salida
1	Una persona puede enseñar
2	Un profesor puede enseñar en un colegio
3	Un profesor puede enseñar en un colegio