**Cuestionario - Polimorfismo y clases abstractas**

1. ¿Dónde reside la definición del método a sobrescribir? ¿Qué palabra clave se usa para definirlo?

Reside en la clase padre, se utiliza virtual para definirlo

1. ¿Dónde reside la implementación del método sobrescrito? ¿Qué palabra clave se usa para implementarlo?

Reside en la clase deriva/hija, se utiliza override

1. ¿La invocación de los métodos sobrescritos (override) se resuelve en tiempo de compilación o ejecución? ¿Y la de los métodos sobrecargados (overload)?

Tiempo de ejecución, creo que en tiempo de compilacion

1. ¿Cambia la firma de los métodos cuando los sobrescribimos (override)? ¿Y cuando los sobrecargamos (overload)?

No. Si debido a que debe recibir distintos parámetros para ser una sobrecarga

1. ¿Los métodos sobrescritos (override) se encuentran en la misma clase? ¿Y los métodos sobrecargados (overload)? ¿Qué es el polimorfismo en el contexto de la programación orientada a objetos?

Creo que sí, Creo que sí. El polimorfismo es la habilidad de vincular objetos de diferentes tipos a un solo identificador en tiempo de ejecución.

1. ¿Todos los objetos en C# son polimórficos? ¿Por qué?

En teoría si, debido a que cada objeto se lo puede vincular a otro objeto dependiendo de las características del mismo

1. Según el siguiente código, complete la tabla indicando:
   1. El tipo de la referencia (Persona o Profesor).
   2. El tipo del objeto en tiempo de ejecución (Persona o Profesor).
   3. La salida por consola.

public class Persona

{

public virtual string Teach()

{

return "Una persona puede enseñar.";

}

}

public class Profesor : Persona

{

public override string Teach()

{

return "Un profesor puede enseñar en un colegio.";

}

}

public 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(persona.Teach());

// Caso 3

Profesor profesor = new Profesor();

Console.WriteLine(persona.Teach());

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| Objeto | Tipo de referencia | Tipo de la instancia en tiempo de ejecución | Texto de la salida por consola |
| persona | Persona | Persona | "Una persona puede enseñar." |
| otraPersona | Persona | Profesor | "Una persona puede enseñar." |
| profesor | Profesor | Profesor | "Una persona puede enseñar." |

**Clases y miembros abstractos**

1. ¿Qué modificador debo utilizar si quiero declarar un método que **pueda** ser sobrescrito en las clases derivadas?

Virtual

1. ¿Qué modificador debo utilizar si quiero declarar un método que **deba** ser sobrescrito en las clases derivadas?

abstract

1. ¿Qué es una clase abstracta? ¿Cuál es su función?

Una clase abstracta es un tipo de clase que nos da la posibilidad de hacer una definición particular que nos genere un modelo para una jerarquía de herencia

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

Si, las clases derivadas deben realizar la implementación del método que se encuentra en la clase base(abstracta) debido a que la clase abstracta

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

No hace falta

1. ¿Se pueden declarar miembros abstractos en clases **no-abstractas**? ¿Por qué?

No se puede

1. ¿Para sobrescribir un método se debe heredar de una clase abstracta? ¿Por qué?

Si, se debe heredar de una clase abstracta para poder sobrescribir un metodo

1. Marque los campos de la siguiente tabla con SÍ o NO según si la afirmación es verdadera para el tipo de clase:

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de clase | Puede heredar de otras clases (ser derivada) | Puede heredarse de ella (ser base) | Puede instanciarse |
| Clase normal (sin modificadores) | Si | Si | Si |
| Clase abstracta (abstract) | Si | Si | No |
| Clase sellada (sealed) | Si | No | Si |
| Clase estatica (static) | No | No | No |

**Herencia**

 Qué es la herencia en el contexto de la programación orientada a objetos? ¿cuál es su propósito?

Es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase

 ¿Qué nombre recibe la clase que hereda y qué nombre recibe la clase que es heredada?

Clase base, clase derivada

 Explique el principio de sustitución de Liskov.

Liskov fue una científica que propuso el concepto de que cada clase que hereda de otra, debe poder usarse como su padre sin necesidad de conocer la diferencia entre las dos

 ¿Qué significa que la herencia es transitiva?

Significa que si nosotros tenemos una clase C, la cual hereda de otra clase B, y al mismo tiempo, B hereda de otra clase A, C hereda de A

 ¿Se heredan los constructores?

No se pueden heredar constructores, pero si puedo llamar los costructores de la clase base, en las clase hija. Ejemplo

public Persona(string nombre, int edad)

{

This.nombre = nombre;

This.edad = edad;

}

Public Profesor(string nombre, int edad, double salario) :base(nombre,edad)

{

This.salario = salario;

}

El :base() llamo al constructor de la clase padre

 ¿Se heredan los miembros private de la clase base?

No, no se puede.

 ¿Qué es herencia múltiple? ¿es posible en C#? ¿en qué se diferencia de la herencia simple?

Se dice herencia múltiple cuando una clase derivada puede tener más de una clase base, adquiriendo los miembros de todos sus padres. Esto no es posible en c#. La diferencia principal es que la herencia simple, una clase derivada hereda de una sola clase base.

 ¿Una clase pública puede heredar de una clase privada?

No se puede, debido a su nivel de proteccion

 ¿Qué es una clase sellada (sealed)?

Una clase sellada (Sealed), implica que esa misma clase no podrá ser heredada (no pueda ser base)

 ¿Una clase sellada puede heredar de otras clases (ser clase derivada)?

Si, si puede

 ¿Cómo actúa el modificador “protected” en los miembros de la clase base para una clase derivada y cómo para una clase no-derivada? Relacionar la respuesta con los modificadores “public” y “private”.

 ¿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?

Genera un error

 La clase Alumno hereda de Persona. ¿Una instancia de Alumno es también de tipo Persona? Justifique.

Si

 La clase Alumno hereda de Persona. ¿Se puede hacer Persona persona = new Alumno()? ¿por qué?

Si se puede hacer

 La clase Alumno hereda de Persona. ¿Se puede hacer Alumno alumno = new Persona()? ¿por qué?

No se puede