Indique cuál/es son las funciones de cada elemento dentro del ciclo de vida de los objetos:

	Asignar espacio en memoria.	Inicializar los datos del objeto.	Liberar memoria ocupada por los objetos.	Ninguna.	Otra.	
Constructores	☐	☑	☐	☐	☐	✓
Operador new	☑	☐	☐	☐	☐	✓
Garbage Collector	☐	☐	☑	☐	☐	✓

✗ Marque cuál/es de las siguientes afirmaciones representan propósitos de la herencia:

- ☑ Reutilizar y organizar mejor el código. ✓
- ☑ Proteger el acceso a la implementación de la clase base. ✗
- ☐ Ninguna respuesta.
- ☑ Extender la estructura y el comportamiento de una clase existente. ✓
- ☐ Crear clases más generales a partir de otras más específicas.

Correct answer

- ☑ Extender la estructura y el comportamiento de una clase existente.
- ☑ Reutilizar y organizar mejor el código.

Feedback

La herencia nos permite crear nuevas clases reutilizando, extendiendo o modificando la estructura y el comportamiento de la clase padre. De esta manera podemos crear clases más específicas a partir de otras más generales.

Su propósito es agrupar atributos y comportamiento en común de un conjunto de clases que componen una determinada realidad, reutilizando y organizando mejor el código.

** El mecanismo de herencia por si mismo no impide el acceso a los detalles de la implementación. Eso tiene más que ver con encapsulamiento.*
** A través de la herencia se pueden crear clases más específicas a partir de otras más generales. Y no al revés.*

✓ Marque la/s afirmaciones verdaderas:

- [x] Las colecciones de tipo SortedList están indexadas por el número correspondiente a la posición de sus elementos y por clave / key. ✓
- [ ] Las colecciones de tipo Queue están indexadas por el número correspondiente a la posición de sus elementos.
- [ ] Las colecciones de tipo Stack están indexadas por el número correspondiente a la posición de sus elementos.
- [ ] Las colecciones de tipo Queue están indexadas por clave / key.
- [ ] Ninguna respuesta.

Feedback

Efectivamente, las colecciones SortedList están conformadas por pares clave-valor que se pueden acceder en base a la posición o por su clave.

Las colecciones tipo Queue y Stack NO están indexadas. Se acceden respectando el orden FIFO y LIFO respectivamente.

✓ Marque la/s afirmaciones verdaderas:

- [ ] Si existe una conversión explícita los tipos se convertirán sin intervención del programador.
- [x] Las conversiones implícitas no deberían implicar pérdida de información. ✓
- [ ] Ninguna respuesta.
- [ ] Las conversiones explícitas no deberían implicar pérdida de información.
- [ ] Las conversiones implícitas exigen el uso del operador de casteo "(tipo)".

Feedback

** Las conversiones explícitas exigen el uso del operador de casteo ya que se suelen utilizar cuando la conversión podría implicar pérdida de información.*
** Las conversiones implícitas no exigen el uso del operador de casteo, se resuelven automáticamente. Esto es porque se suelen utilizar cuando la conversión no podría implicar pérdida de información.*

✗ Marque la/s afirmaciones correctas con respecto a los objetos:

- [ ] Los objetos son creados por el compilador.
- [ ] Los objetos poseen comportamiento (atributos) y estado (métodos).
- [x] Un objeto es una manifestación concreta / específica de una clase en tiempo de ejecución. Contendrá valores concretos en sus atributos que utilizará para operar dentro de sus métodos. ✓
- [x] Un objeto contiene una referencia al bloque de memoria donde está almacenada la clase. ✗
- [ ] Ninguna respuesta.

Correct answer

- [x] Un objeto es una manifestación concreta / específica de una clase en tiempo de ejecución. Contendrá valores concretos en sus atributos que utilizará para operar dentro de sus métodos.

Feedback

** El estado de un objeto son sus datos (atributos). El comportamiento se representa por implementaciones concretas de sus operaciones conocidas como métodos.*
** Es el objeto quien está almacenado en memoria, no la clase. La clase es un molde o plantilla a partir de la cual se instancian objetos. La referencia al objeto en memoria puede estar almacenada en una variable.*
** Los objetos se crean en tiempo de ejecución.*

✓ Marque la/s afirmaciones correctas con respecto a indexadores:

- [x] En la implementación de un indexador, la palabra clave "value" contendrá el valor a asignar en base al índice proporcionado. ✓
- [ ] Ninguna respuesta.
- [x] Indexar es ordenar una serie de datos de acuerdo a un criterio, para facilitar su consulta a través de un índice. ✓
- [ ] Las clases sólo se pueden indexar por un parámetro numérico.
- [ ] Los indexadores sólo se pueden utilizar si la clase tiene declarada una colección como atributo.
- [ ] Indexar es darle al objeto un espacio en memoria.

Feedback

** Darle al objeto un espacio en memoria es parte del proceso de instanciar, no tiene que ver con indexar.*
** Las clases se pueden indexar por múltiples parámetros de cualquier tipo. Depende de la necesidad y del criterio de indexación diseñado para la clase.*
** No es un requisito que la clase tenga una colección o array como atributo para poder ser indexada. De nuevo, depende de la necesidad y del criterio de indexación diseñado.*

✗ Marque la/s afirmaciones correctas con respecto a las clases en el contexto de la programación orientada a objetos:

- [ ] Una clase es una abstracción que representa un conjunto de características y comportamiento en común dentro de un grupo de objetos.
- [ ] Una clase es un bloque de memoria que se ha asignado y configurado para almacenar un conjunto de datos.
- [x] Una clase es un conjunto de objetos. ✗
- [x] Una clase define los atributos y el comportamiento que tendrán los objetos de ese tipo. ✓
- [ ] Ninguna respuesta.

Correct answer

- [x] Una clase es una abstracción que representa un conjunto de características y comportamiento en común dentro de un grupo de objetos.
- [x] Una clase define los atributos y el comportamiento que tendrán los objetos de ese tipo.

Feedback

Una clase es un tipo de clasificador que representa un conjunto de atributos y operaciones en común para un conjunto de objetos o entidades.

Funciona como plantilla o molde para la creación de instancias de objetos.

** Las clases definen los atributos y el comportamiento que tendrán los objetos, los cuales se crean a partir de ellas.*

** Lo que se almacena en un bloque de memoria es una instancia de la clase u objeto.*

** Un conjunto de objetos es una colección o array.*

✗ Marque cuál/es de las siguiente afirmaciones son verdaderas:

- [ ] Aplicar encapsulamiento es permitir acceso directo e irrestricto para consultar y modificar el estado de un objeto.
- [x] El modificador protected permitirá que sólo se tenga acceso a los miembros desde la clase derivada, no pudiendo acceder desde la clase donde están declarados. ✗
- [x] El encapsulamiento previene el acceso a los detalles de la implementación y protege el acceso y modificación de los datos del objeto. ✓
- [ ] Ninguna respuesta.
- [ ] El modificador internal permitirá que sólo se tenga acceso a los miembros desde la clase donde se declaran.

Correct answer

- [x] El encapsulamiento previene el acceso a los detalles de la implementación y protege el acceso y modificación de los datos del objeto.

Feedback

El pilar del encapsulamiento consiste en agrupar en un contenedor los datos del objeto junto con los métodos que operan sobre esos datos (Clases). Al mismo tiempo que se ocultan los detalles de la implementación o internos y se protege el acceso a datos. Exponiendo únicamente lo esencial / importante.

** El modificador internal permite acceso desde clases dentro de un mismo ensamblado (unidad de compilación / proyecto).*
** El modificador protected permite acceso desde la misma clase y derivadas.*
** El encapsulamiento protege el acceso a datos y oculta los detalles de la implementación.*

✗ Marque la/s afirmaciones verdaderas:

- [x] Los arrays pueden ser recorridos con un foreach ya que implementan la interfaz IEnumerable. ✓
- [ ] Ninguna respuesta.
- [x] El primer formulario de Windows Forms se instancia dentro de un método en la clase Form. ✗
- [x] Una de las características del tipo "string" es ser inmutable. Esto significa que no pueden cambiar una vez que fueron inicializados. Cuando modificamos un string en realidad estamos generando una nueva cadena. ✓
- [x] Los elementos de un array "Jagged" se inicializan siempre en null. ✓

Correct answer

- [x] Los elementos de un array "Jagged" se inicializan siempre en null.
- [x] Los arrays pueden ser recorridos con un foreach ya que implementan la interfaz IEnumerable.
- [x] Una de las características del tipo "string" es ser inmutable. Esto significa que no pueden cambiar una vez que fueron inicializados. Cuando modificamos un string en realidad estamos generando una nueva cadena.

Feedback

** El primer formulario se instancia en el método Main. El cuál es el entry-point de todos los programas ejecutables de .NET.*

** Los elementos de un array "Jagged" son otros arrays. Los arrays son reference type, por lo tanto se por defecto con null.*
** Los objetos de las clases que implemen IEnumerable podrán ser recorridos con un foreach.*
** Efectivamente, inmutable significa que los datos del objeto no podrán ser alterados una vez que fue instanciado. Es el caso de los strings.*

✓ Marque la/s afirmaciones correctas sobre operadores:

- [ ] Ninguna respuesta.
- [ ] No existen diferencias en la sintaxis de las sobrecargas de operadores binarios y unarios.
- [x] Un operador es un símbolo que especifica un algoritmo a ejecutar dados uno o más operandos de un tipo específico. ✓
- [ ] Si sobrecargo el operador + (suma) también tendré que sobrecargar el operador += (suma y asignación).
- [ ] Cuando sobrecargamos el operador de igualdad "==" también debemos sobrescribir el método Equals. De no hacerlo, resultará en un error en tiempo de compilación.
- [ ] Se dice que los operadores son binarios cuando sólo permiten retornar dos posibles valores.

Feedback

** El operador suma y asignación se sobrecarga automáticamente al sobrecargar el operador suma.*
** Los operadores se dicen binarios cuando operan con dos operandos.*
** En la declaración de la sobrecarga de un operador unario hay un solo parámetro de entrada (operando), mientras que los binarios tienen dos.*
** Si bien se recomienda sobrescribir el comportamiento del método Equals al sobrecargar el operador de igualdad, no hacerlo no resultará en un error en tiempo de compilación.*

** Para una mejor definición de operadores, consulte un diccionario. ¯_(ツ)_/¯*

- [x] Las propiedades permiten que una clase exponga una manera segura de exponer y modificar valores almacenados en sus atributos o calculados. ✓
- [x] Los enumerados sirven para acotar los valores posibles que se pueden asignar a una variable o parámetro. ✓
- [x] Un enumerado es un tipo de dato que define un conjunto de constantes numéricas con nombre. ✓
- [ ] Ninguna respuesta.
- [x] Un enumerado es un descriptor de acceso que define un conjunto de pares clave-valor de tipo <string, int>. ✗
- [ ] Debe haber una propiedad por cada atributo que tenga la clase, exponiendo siempre TODOS los datos del objeto.

Correct answer

- [x] Un enumerado es un tipo de dato que define un conjunto de constantes numéricas con nombre.
- [x] Los enumerados sirven para acotar los valores posibles que se pueden asignar a una variable o parámetro.
- [x] Las propiedades permiten que una clase exponga una manera segura de exponer y modificar valores almacenados en sus atributos o calculados.

Feedback

** Siguiendo el principio de encapsulamiento. Se deben exponer únicamente los datos esenciales para quien consume los objetos de la clase.*

Habrá muchos atributos que serán de uso interno y no requieren ser consultados externamente. Habrá otros que serán expuestos a través de un cálculo.

Esto favorece que la clase exponga una interfaz limpia y segura.

** Un enumerado no es un descriptor de acceso (esos son get y set dentro de una propiedad). Tampoco expone pares clave-valor, son constantes numéricas (enteros).*

✗ Marque cuál/es de las siguientes afirmaciones sobre herencia son verdaderas:

- [ ] Se heredan constructores, métodos, propiedades y atributos.
- [ ] Ninguna respuesta.
- [x] Si B hereda de A y C hereda de B, entonces C hereda transitivamente de A. ✓
- [ ] Si la clase derivada no hace una llamada explícita a un constructor de la clase base, el compilador de C# usará implícitamente un constructor de la forma ":base()", dando un error en tiempo de compilación si no existiera un constructor con esa firma.
- [ ] C# admite herencia múltiple ya que todas las clases heredan de la clase Object.
- [ ] La clase base será una especialización de la clase derivada.
- [ ] Los miembros privados NO se heredan.

Respuesta correcta

- [x] Si B hereda de A y C hereda de B, entonces C hereda transitivamente de A.
- [x] Si la clase derivada no hace una llamada explícita a un constructor de la clase base, el compilador de C# usará implícitamente un constructor de la forma ":base()", dando un error en tiempo de compilación si no existiera un constructor con esa firma.

Comentarios

** Los constructores no se heredan.*
** Los miembros privados (que no sean constructores) SÍ se heredan, pero no se pueden acceder.*
** La clase derivada es una especialización de la clase base, y no al revés.*
** C# no admite herencia múltiple. Object se hereda de forma transitiva.*

✗ Marque la/s afirmaciones verdaderas:

- [ ] Ninguna respuesta.
- [ ] Los métodos no-estáticos se instancian utilizando la palabra clave "new".
- [ ] Instanciar un objeto es declarar una variable del tipo de la clase que estamos instanciando.
- [ ] Un namespace es el nombre que recibe nuestro proyecto internamente. Representa una unidad de compilación.
- [ ] Los métodos estáticos requieren instanciar una clase para poder invocarlos.

Feedback

Ninguna es correcta.

** Un namespace es una agrupación lógica con fines de organizar nuestras clases y otros elementos. No tiene relación con el proyecto, el cual sí representa una unidad de compilación.*
** Declarar una variable no es lo mismo que instanciar. De hecho, no se necesita declarar una variable para instanciar un objeto.*
** Los métodos no se instancian.*
** Los métodos estáticos son de la clase y no se invocan desde una instancia específica. Por lo tanto, no requieren ninguna instancia.*

Indique TODAS la/s afirmaciones correctas de acuerdo a cada modificador de accesibilidad:

El miembro con dicho modificador se podrá acceder desde...

	La clase que lo declara.	Una clase derivada declarada en el mismo proyecto.	Cualquier clase dentro del mismo proyecto.	Cualquier clase.	Ninguna.	
Private	☑	☐	☐	☐	☐	✓
Public	☐	☐	☑	☐	☐	✗
Internal	☐	☐	☐	☑	☐	✗
Protected	☐	☑	☐	☐	☐	✗

Correct answers

	La clase que lo declara.	Una clase derivada declarada en el mismo proyecto.	Cualquier clase dentro del mismo proyecto.	Cualquier clase.	Ninguna.
Public	☑	☑	☑	☑	☐
Internal	☑	☑	☑	☐	☐
Protected	☑	☑	☐	☐	☐

✓ Marque la/s afirmaciones correctas sobre Windows Forms:

- [ ] El método ShowDialog() visualiza un formulario de forma modal, permitiendo interactuar con otras ventanas.
- [ ] Los formularios deben tener obligatoriamente un constructor público sin parámetros.
- [ ] Los formularios son tipos especiales que no requieren ser instanciados.
- [x] Ninguna respuesta. ✓
- [ ] El modificador "partial" permite declarar dos clases distintas con el mismo nombre.

Feedback

Ninguna es correcta.

** Los formularios son instancias de clases que heredan de Form. Son objetos como cualquier otro.*
** No requieren tener un constructor sin parámetros.*
** El modificador partial permite que la declaración de una misma clase esté separada en dos o más archivos.*
** Efectivamente ShowDialog() es para mostrar el formulario de forma modal, pero las ventanas modales no permiten interactuar con otras ventanas.*

✓ Marque la/s afirmaciones verdaderas:

- [ ] Dos objetos del mismo tipo pueden tener distintos valores en una misma constante.
- [ ] Ninguna respuesta.
- [ ] Los objetos son almacenados en el segmento de memoria conocido como "Stack"
- [ ] Los objetos se crean en tiempo de compilación.
- [x] El valor de las constantes es asignado en tiempo de compilación. ✓

Feedback

** El valor de las constantes no puede cambiar durante la ejecución del programa, por lo tanto es asignado en tiempo de compilación.*

** Los objetos se crean en tiempo de ejecución.*
** Las constantes se comportan como atributos estáticos, por lo tanto hay un sólo valor para todos los objetos de esa clase. Además, no pueden ser modificadas en tiempo de ejecución, ¿cómo cambiarías su valor desde un constructor de un objeto?.*
** Los objetos se almacenan en el segmento Heap, no en el Stack.*

- [x] No todos los objetos de C# son polimórficos. ✗
- [ ] Es la propiedad que permite que objetos de diferentes tipos puedan ser accedidos a través de la misma interfaz, proveyendo una implementación específica dependiendo del tipo en tiempo de ejecución.
- [ ] Ninguna respuesta.
- [x] Se aplicar usando el operador new (entre otros). ✗
- [x] Es la propiedad que tienen los objetos de tomar distintas formas (tipos). ✓
- [ ] El polimorfismo se resuelve en tiempo de compilación.

Correct answer

- [x] Es la propiedad que tienen los objetos de tomar distintas formas (tipos).
- [x] Es la propiedad que permite que objetos de diferentes tipos puedan ser accedidos a través de la misma interfaz, proveyendo una implementación específica dependiendo del tipo en tiempo de ejecución.

Feedback

El polimorfismo describe el concepto de que objetos de diferentes tipos pueden ser accedidos a través de la misma interfaz.

Cada tipo puede proveer su propia implementación de la interfaz.

En otras palabras, en relaciones de herencia, llamando a una misma firma de un método (mismo nombre, parámetros, todo igual), se ejecutará una implementación distinta según el tipo del objeto en tiempo de ejecución.

Es también la propiedad de los objetos de tomar distintas formas (tipos).

** Todos los objetos de C# son polimórficos ya que todos pueden tomar la forma de object (clase de la cual heredan todas las clases de forma implícita).*
** El polimorfismo se resuelve en tiempo de ejecución, ya que los objetos son creados en tiempo de ejecución.*
** El operador new permite declarar un método con el mismo nombre que uno heredado, invalidando el heredado. Pero no permite que se ejecute la implementación correspondiente al tipo del objeto en tiempo de ejecución, sino que se ejecutará la implementación del tipo de la referencia.*

✓ Marque la/s afirmaciones verdaderas:

- [x] Ninguna respuesta. ✓
- [ ] Las colecciones de tipo Queue se procesan con el orden conocido como LIFO.
- [ ] Las colecciones genéricas están compuestas por elementos de tipo object, no contando con seguridad de tipos.
- [ ] La diferencia entre colecciones y matrices es que las primeras son fuertemente tipadas mientras que las segundas no.
- [ ] Las colecciones de tipo Stack se procesan con el orden conocido como FIFO.

Feedback

Ninguna es correcta.

** Las colecciones de tipo Stack se procesan con orden LIFO.*
** Las colecciones de tipo Queue se procesan con orden FIFO.*
** Las colecciones genéricas se caracterizan por ser fuertemente tipadas, se componen de elementos de un tipo concreto.*
** La diferencia entre colecciones y matrices es que las primeras son de tamaño dinámico mientras que las segundas son de tamaño fijo.*

✗ Marque la/s afirmaciones verdaderas:

- [ ] El entry point para los programas en C# es el método "Program".
- [x] El CTS es una especificación que define un conjunto común de tipos de datos que deben ser implementados por todos los lenguajes de la plataforma .NET. ✓
- [x] Las variables no-escalares contienen sólo un valor o dato atómico y unidimensional. ✗
- [ ] Ninguna respuesta.
- [ ] C# es un lenguaje interpretado y su intérprete es el runtime conocido como Visual Studio.

Correct answer

- [x] El CTS es una especificación que define un conjunto común de tipos de datos que deben ser implementados por todos los lenguajes de la plataforma .NET.

Feedback

El CTS o Common Type System es un conjunto de especificaciones sobre tipos de datos que deben implementar todos los lenguajes de .NET.

Es decir, establece la base de tipos primitivos para todos los lenguajes permitiendo la colaboración entre distintos lenguajes de la plataforma.

** El entry point de los programas en C# es el método Main.*
** C# es un lenguaje compilado. El Visual Studio es un IDE.*
** Las variables no-escalares son estructuras que almacenan más de un valor.*

✓ Marque la/s afirmaciones verdaderas:

- [ ] Las colecciones del tipo Dictionary están indexadas por el número correspondiente a la posición de sus elementos.
- [ ] Las matrices, a diferencia de los arrays, son dinámicas. Es decir, incrementan y decrementan su tamaño según la demanda en tiempo de ejecución.
- [ ] Ninguna respuesta.
- [ ] La diferencia entre colecciones no-genéricas y colecciones genéricas, es que las primeras tienen un tamaño fijo.
- [x] Las colecciones de tipo SortedList están compuestas por pares clave-valor. ✓

Feedback

** Tanto las colecciones genéricas como las no-genéricas son dinámicas, su tamaño crece y decrece de demanda.*
** Las matrices tienen tamaño fijo. Los arrays son matrices de una sola dimensión.*
** Los diccionarios están compuestos por pares clave-valor y están indexados por la clave o key.*

✗ Marque la/s afirmaciones correctas sobre el proceso de compilación de C#:

- [x] Ninguna respuesta. ✗
- [ ] El compilador de C# traducirá el código fuente a código intermedio en tiempo de compilación, necesitando la intervención de un intérprete en tiempo de ejecución.
- [ ] El código en lenguaje C# será interpretado por el runtime en tiempo de ejecución.
- [ ] El compilador de C# traducirá el código fuente a lenguaje máquina, permitiendo su ejecución sin intermediarios.
- [ ] El Visual Studio se encargará de interpretar el código intermedio generado por el entorno de ejecución.

Correct answer

- [x] El compilador de C# traducirá el código fuente a código intermedio en tiempo de compilación, necesitando la intervención de un intérprete en tiempo de ejecución.

Feedback

** Lo que interpreta el runtime en tiempo de ejecución es código intermedio o bytecode, no C#.*
** El Visual Studio es un IDE, no un interprete. El código intermedio es generado por el compilador, no por el runtime.*
** El compilador de C# traduce el código fuente a código intermedio, el cual deberá ser interpretado en tiempo de ejecución por el runtime.*

✓ Marque la/s afirmaciones correctas sobre Windows Forms:

- [x] Los formularios son objetos, por lo tanto requieren ser instanciados. El primer formulario será instanciado en el método Main. ✓
- [ ] Un formulario MDI no permite interactuar con otras ventanas mientras se encuentre abierto.
- [ ] Ninguna respuesta.
- [ ] El método Show() visualiza un formulario de forma modal, permitiendo interactuar con otras ventanas.
- [x] El modificador partial permite separar la declaración de una misma clase en dos o más archivos. ✓

Feedback

** Esa definición es para los formularios MODALES, no tiene nada que ver con los formularios MDI.*
** El método Show() muestra un formulario de forma NO-MODAL, permitiendo interactuar con otras ventanas.*

✓ Marque la/s afirmaciones correctas con respecto a los constructores:

- [ ] El constructor por defecto será invocado por el runtime en el momento de la primera interacción con la clase.
- [ ] Los constructores estáticos son los encargados de inicializar los atributos del objeto.
- [ ] Los constructores estáticos sólo pueden ser invocados desde métodos estáticos.
- [ ] Los constructores son los encargados de reservar memoria cuando se instancia un objeto.
- [x] Si no se declaró de forma explícita un constructor, existirá un constructor público y sin parámetros que será invocado cada vez que se instancie un objeto de esa clase. ✓
- [ ] Ninguna respuesta.

Feedback

Si no se declara de forma explícita un constructor, existirá un constructor por defecto. El constructor por defecto es público y no tiene parámetros de entrada.

** Los constructores inicializan los atributos del objeto. No reservan memoria, esa tarea es del operador new.*
** El constructor por defecto - si no fue reemplazado - se invoca al instanciar un nuevo objeto de esa clase.*
** Los constructores estáticos no pueden ser invocados. La invocación la realiza el runtime en la primera interacción con la clase.*
** Los constructores estáticos son de la clase, no de una instancia específica. Por lo tanto, sólo pueden inicializar atributos de clase (estáticos).*

✗ Indique cuál/es de las siguientes son funciones del runtime (entorno de ejecución) de .NET:

- [ ] Ninguna respuesta.
- [x] Liberar, a través del Garbage Collector, memoria almacenada en la región conocida como "Stack". ✗
- [ ] Compilar código fuente escrito en lenguaje C#.
- [x] Administración de la memoria utilizada por la aplicación. ✓
- [x] Compilar código intermedio a código nativo (máquina) cuando se ejecuta la aplicación. ✓

Correct answer

- [x] Administración de la memoria utilizada por la aplicación.
- [x] Compilar código intermedio a código nativo (máquina) cuando se ejecuta la aplicación.

Feedback

Las funciones del runtime son:
** Manejar la administración y alocación de memoria.*
** Ejecutar las aplicaciones al compilar a lenguaje nativo / máquina el lenguaje intermedio al que se compila el código fuente escrito con los lenguajes soportados por .NET.*
** Generar una capa de abstracción del hardware en el que corren las aplicaciones. Nuestro código es agnóstico del hardware, el que depende es el runtime.*

** El runtime no compila código. Intepreta código intermedio en tiempo de ejecución.*
** El garbage collector libera la memoria almacenada en el segmento Heap, no en el Stack. El segmento Stack es liberado automáticamente.*

✗ Marque la/s opciones correctas con respecto a sobrecarga de constructores:

- [ ] Los constructores privados no se pueden sobrecargar.
- [x] El operador :this() definirá a qué sobrecarga invocar en base a los nombres de los parámetros. ✗
- [x] El operador :this() me permitirá invocar a cualquier tipo de constructor dentro de la clase. ✗
- [x] Los constructores estáticos no se pueden sobrecargar. ✓
- [ ] Ninguna respuesta.
- [ ] El operador :this() me permitirá invocar distintas sobrecargas de constructores en la clase base.

Correct answer

- [x] Los constructores estáticos no se pueden sobrecargar.

Feedback

** Los constructores estáticos no tienen parámetros de entrada, por lo tanto no se pueden sobrecargar.*

** No se puede invocar constructores estáticos con el operador :this().*
** Los constructores privados SÍ se pueden sobrecargar.*
** Los constructores de la clase base se invocan con el operador :base(), no con el :this().*
** Las sobrecargas no se resuelven por el nombre o identificador de los parámetros entrada. Lo que analiza el compilador es el número y orden de los TIPOS de los parámetros.*

✗ Marque cuál/es de las siguientes afirmaciones es verdadera:

- [x] La principal función de las clases abstractas es ser la base de una jerarquía de herencia, definiendo un marco de atributos y comportamiento para todas las clases que deriven de ella. ✓
- [ ] Las clases abstractas no pueden ser instanciadas ni heredadas.
- [ ] Las clases estáticas no pueden ser instanciadas, pero pueden heredar.
- [ ] Ninguna respuesta.
- [ ] Las clases selladas (sealed) no pueden heredar de otras clases.
- [x] Los métodos virtuales definidos en una clase abstracta deberán ser sobrescritos obligatoriamente por la clase derivada. ✗

Correct answer

- [x] La principal función de las clases abstractas es ser la base de una jerarquía de herencia, definiendo un marco de atributos y comportamiento para todas las clases que deriven de ella.

Feedback

** Las clases abstractas no pueden ser instanciadas, pero sí pueden ser heredadas. Están pensadas para ser la base de una jerarquía de herencia.*
** Las clases selladas no pueden ser heredadas (ser base), pero sí pueden heredar (ser derivadas).*
** Las clases estáticas no pueden heredar, ni ser heredadas, ni instanciarse.*
** Los métodos virtuales ya tienen una implementación por defecto, por lo tanto no es obligatoria su invalidación en la clase derivada. Los abstractos, por otro lado, no tienen implementación por defecto y deben ser sobrescritos por las clases derivadas.*

✗ Marque cuál/es de las siguientes afirmaciones sobre herencia son verdaderas:

- [ ] La clase base será una especialización de la clase derivada.
- [x] Si B hereda de A y C hereda de B, entonces C hereda transitivamente de A. ✓
- [ ] Se heredan constructores, métodos, propiedades y atributos.
- [x] C# admite herencia múltiple ya que todas las clases heredan de la clase Object. ✗
- [ ] Ninguna respuesta.
- [ ] Si la clase derivada no hace una llamada explícita a un constructor de la clase base, el compilador de C# usará implícitamente un constructor de la forma ":base()", dando un error en tiempo de compilación si no existiera un constructor con esa firma.
- [ ] Los miembros privados NO se heredan.

Correct answer

- [x] Si B hereda de A y C hereda de B, entonces C hereda transitivamente de A.
- [x] Si la clase derivada no hace una llamada explícita a un constructor de la clase base, el compilador de C# usará implícitamente un constructor de la forma ":base()", dando un error en tiempo de compilación si no existiera un constructor con esa firma.

Feedback

** Los constructores no se heredan.*
** Los miembros privados (que no sean constructores) SÍ se heredan, pero no se pueden acceder.*
** La clase derivada es una especialización de la clase base, y no al revés.*
** C# no admite herencia múltiple. Object se hereda de forma transitiva.*

```
public class Numero
{
    private int numero;

    private Numero(int numero)
    {
        this.numero = numero;
    }

    public int NumeroLoco
    {
        get { return this.numero; }
        set { this.numero = value; }
    }

    public static implicit operator Numero(int numero)
    {
        return new Numero(numero);
    }
}
```

```
static void Main(string[] args)
{
    int numeroUno = 1;
    int numeroDos = 2;
    Numero numeroTres = 3;

    MetodoLoco(numeroUno);
    MetodoLoco(out numeroDos);
    MetodoLoco(numeroTres);

    Console.WriteLine("{0} - {1} - {2}", numeroUno, numeroDos, numeroTres.NumeroLoco);
    Console.ReadKey();
}

public static void MetodoLoco(int numero)
{
    numero = 14;
}

public static void MetodoLoco(out int numero)
{
    numero = 14;
}

public static void MetodoLoco(Numero numero)
{
    numero.NumeroLoco = 14;
}
```

- ○ 14 - 2 - 14
- ○ 14 - 14 - 3
- ○ 14 - 2 - 3
- ◉ 1 - 14 - 14 ✓
- ○ Ninguna respuesta.
- ○ 1 - 14 - 3
- ○ 14 - 14 - 14
- ○ 1 - 2 - 14

Feedback

** El tipo int es un value type, al pasarse como argumento a un método se realizará una copia del valor y cualquier modificación al mismo dentro del método NO se verá reflejada en el ámbito de llamada.*
** Los value type que se pasen a un parámetro con el modificador "out" se comportarán como argumentos de tipo reference, lo que tiene como consecuencia que cualquier modificación al mismo dentro del método se verá reflejada dentro del ámbito de llamada.*
** Los objetos son reference type, cualquier modificación de los mismos dentro de un método se verá reflejada en el ámbito de llamada.*

✗ El siguiente código corresponde con un patrón de diseño llamado Singleton. Interprete el código e indique cuál/es de las siguientes afirmaciones es correcta:

```
public class Singleton
{
    private static Singleton instance;

    private Singleton() { }

    public static Singleton Instance
    {
        get
        {
            if (instance is null)
            {
                instance = new Singleton();
            }
            return instance;
        }
    }
}
```

- ☐ Para crear un objeto de la clase "Singleton" desde otra clase deberemos usar el operador "new" junto con el constructor.
- ☑ Desde otras clases sólo se podrá instanciar un objeto de la clase Singleton. ✓
- ☑ El campo estático "instance" se instanciará solamente cuando se llame a la propiedad "Instance" la primera vez. ✓
- ☐ El campo estático "instance" se inicializará la primera vez que cualquier miembro de la clase sea referenciado.
- ☐ Ninguna respuesta.
- ☐ Error en tiempo de ejecución.
- ☐ Error en tiempo de compilación.

- [ ] Error en tiempo de compilación.
- [x] La propiedad "Instance" es de solo lectura. ✓
- [x] Desde otras clases, se podrán instanciar múltiples objetos de la clase Singleton. ✗
- [ ] El constructor privado no se está utilizando.

Correct answer

- [x] Desde otras clases sólo se podrá instanciar un objeto de la clase Singleton.
- [x] El campo estático "instance" se instanciará solamente cuando se llame a la propiedad "Instance" la primera vez.
- [x] La propiedad "Instance" es de solo lectura.

Feedback

Explicación del patrón singleton en el siguiente enlace:

Singleton

Es una relación entre clases en la cual una clase comparte la estructura y comportamiento definido en otra clase.	◉	○	○	○	
Consiste en ocultar los detalles de la implementación y proteger el acceso a datos.	○	○	○	◉	
Permite crear clases más especializadas a partir de otras más generales.	○	◉	○	○	
Es la propiedad que tienen los objetos de permitir invocar genéricamente un comportamiento (método) cuya implementación será delegada al objeto	○	○	○	○	
Implica la definición de métodos en una clase base y sobrescribirlos con nuevas implementaciones en clases derivadas.	○	○	○	○	
Consiste en seleccionar las características relevantes/importantes y comportamiento/operaciones en común dentro de un conjunto de objetos, definiendo nuevos tipos de entidades.	○	○	◉	○	

	Polimorfismo	Herencia	Abstracción	Encapsulamiento	Virtual	Privat
Es una relación entre clases en la cual una clase comparte la estructura y comportamiento definido en otra clase.	○	◉	○	○	○	○
Es la propiedad que tienen los objetos de permitir invocar genéricamente un comportamiento (método) cuya implementación será delegada al objeto	◉	○	○	○	○	○
Implica la definición de métodos en una clase base y sobrescribirlos con nuevas implementaciones en clases derivadas.	◉	○	○	○	○	○